Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

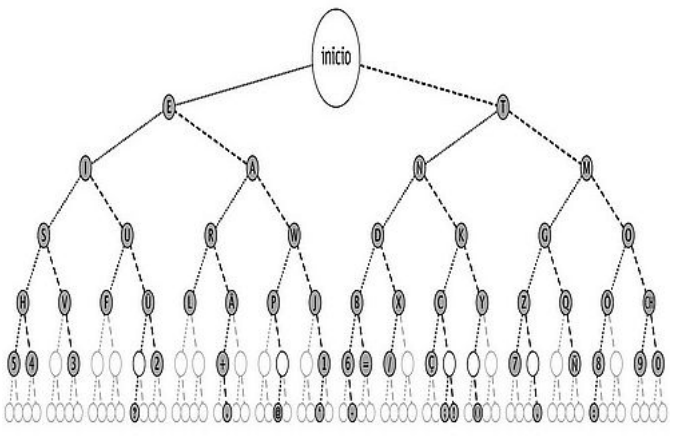
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

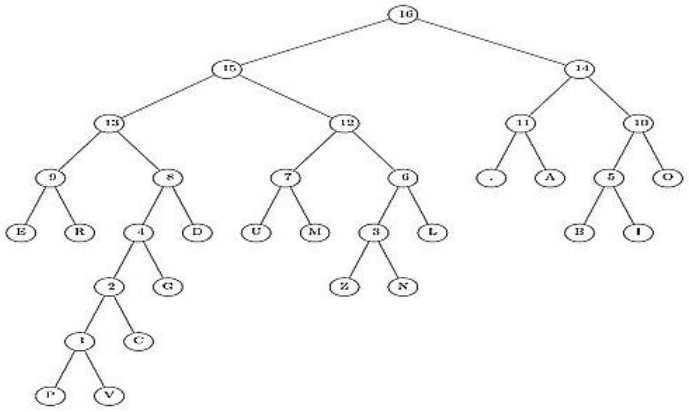
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
-.-.1....1.-1.--.1.1..-12
2)
--.1.-1...-1..1.-1---12
3)
-.-.1.-1--1..1...1.-12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
A4 42 65 26 B6 80                    41 bits significativos
5)
CE F6 56 23 C0                       36 bits significativos
6)
2A 0E CC EB 20                       37 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76506

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76663 DANILO DEPETRIS SOARES
24CC2406 Exercício : 2 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

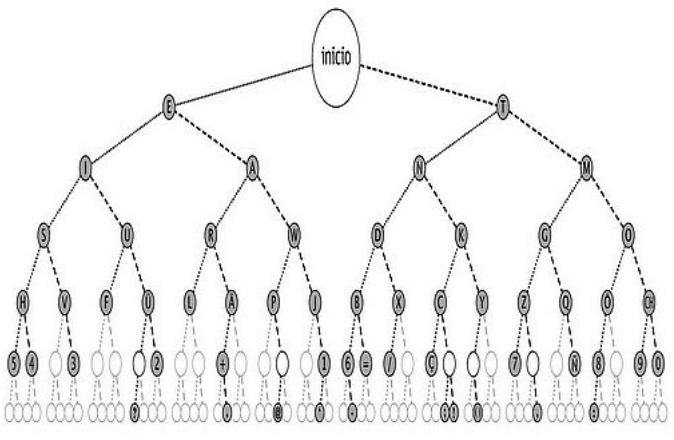
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

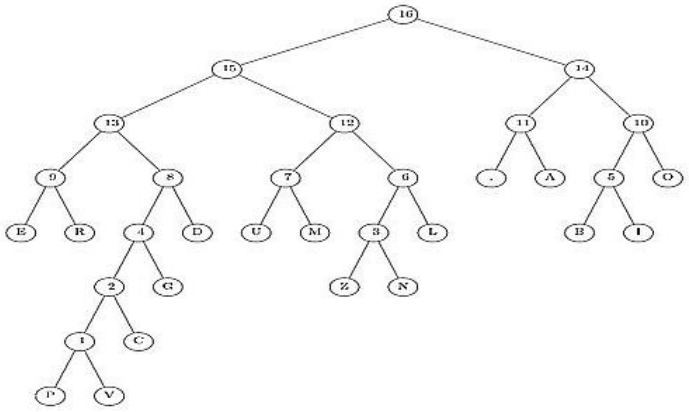
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O   .     C    A    O   .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.-..1.1-1.-.1.-1...12
2)
.1...1-.-.1---1.-..1.-12
3)
-..1..1...1-.-.1---1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
26 8D 95 80 7C                          38 bits significativos
5)
20 10 40 FB 4F 80                       41 bits significativos
6)
CE F6 56 23 C0                          36 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76663

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76513 EDUARDO CAVALHEIRO DIAS
24CC2406 Exercício : 3 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

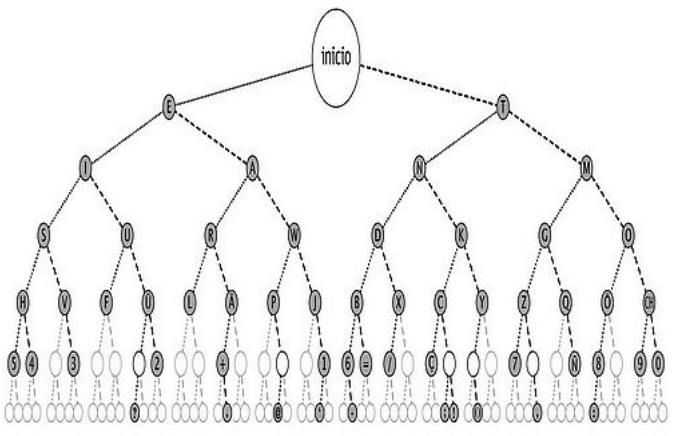
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

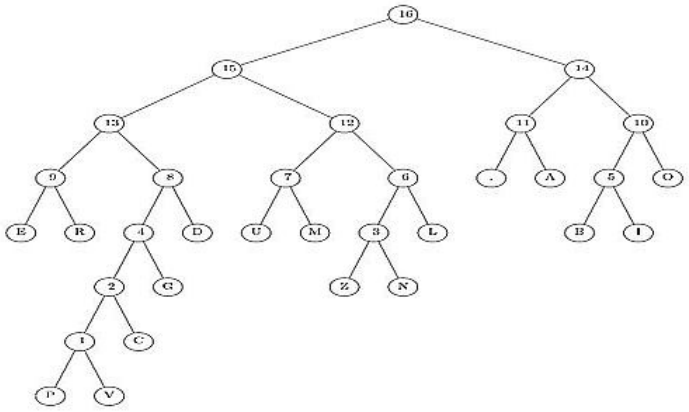
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
---.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
--.1.-1...-1..1.-1---12
2)
-.-.1.-1.-..1-.-1.-1...12
3)
.1...1-.-.1---1.-..1.-12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
20 06 3E 67 90                          38 bits significativos
5)
CE F6 56 23 C0                          36 bits significativos
6)
CE F6 72 35 E0                          37 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76513

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76520 ERIC CORDEIRO
24CC2406 Exercício : 4 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

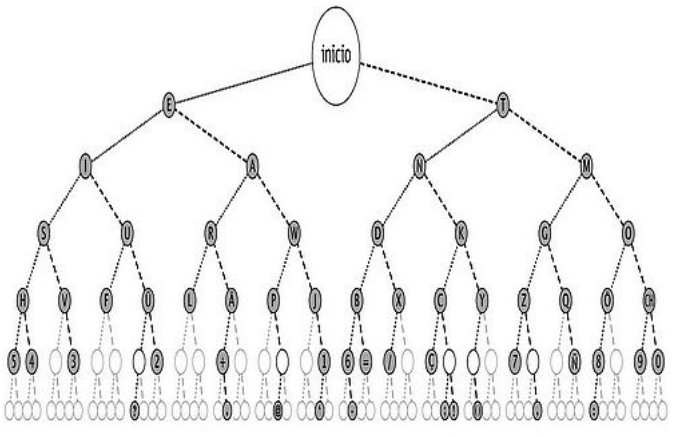
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

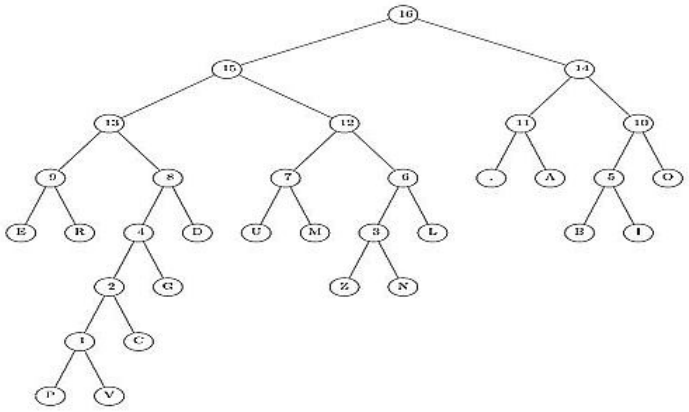
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.1...1-.-.1---1.-..1.-12
2)
-.-.1---1-...1.-.1.-1...12
3)
-.-.1.-1.-..1-.-.1.-1---12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
E4 7C 21 CC 42 00                    41 bits significativos
5)
2A 0E CC EB 20                       37 bits significativos
6)
A4 42 65 26 B6 80                    41 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76520

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76537 FELIPE EIJI KANO
24CC2406 Exercício : 5 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

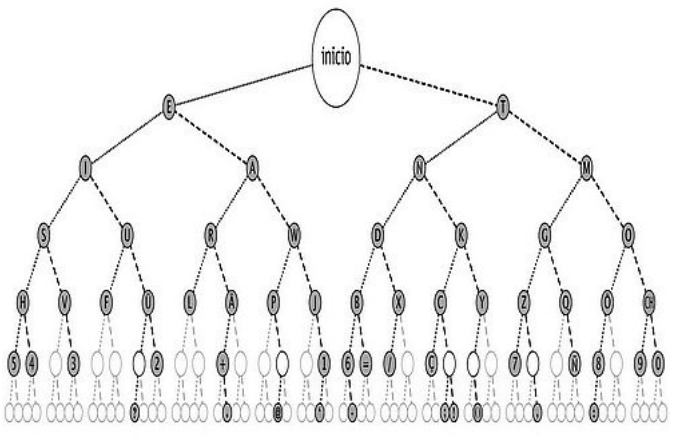
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

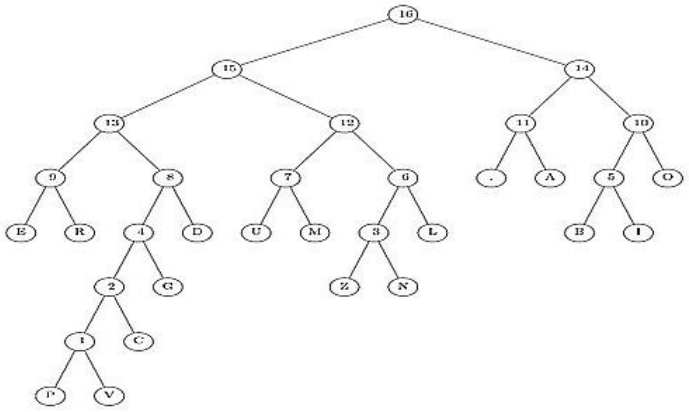
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O   .     C    A    O   .     B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
-.-.1.-1.-..1-.-.1.-1...12
2)
--.1.-1...-1..1.-1---12
3)
-.-.1..-1.-.1...1---1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
26 8D 95 80 7C                          38 bits significativos
5)
2A 0E D8 B6 40                          36 bits significativos
6)
26 8D 95 80 74                          38 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76537

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76544 FELIPE JESMIEL LEITE
24CC2406 Exercício : 6 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

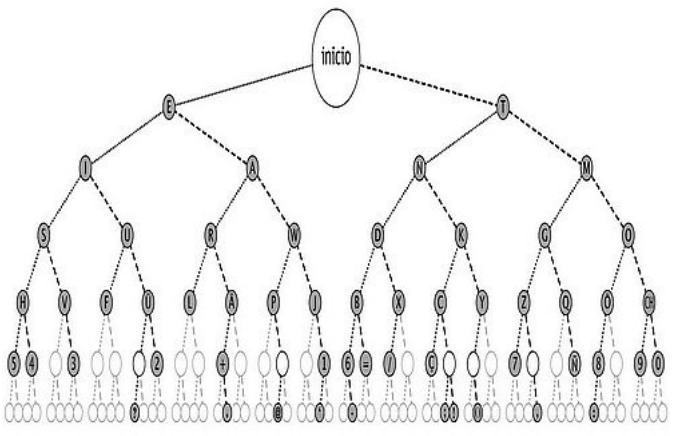
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

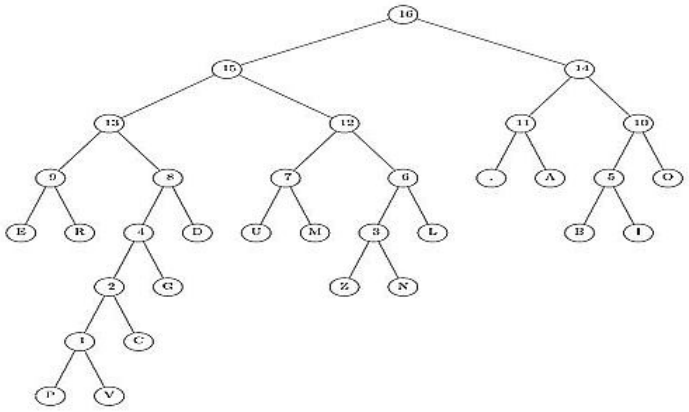
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
-.-.1.-1-.1.1-1.-12
2)
-.-.1.-1--1..1...1.-12
3)
.--.1.-1...1-1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
AC 47 84 82 48                          39 bits significativos
5)
5A 42 C1 4D 58                          39 bits significativos
6)
20 10 BC 4F C8                          39 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76544

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76551 GABRIELA BERNARDO PASQUALOTTO
24CC2406 Exercício : 7 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

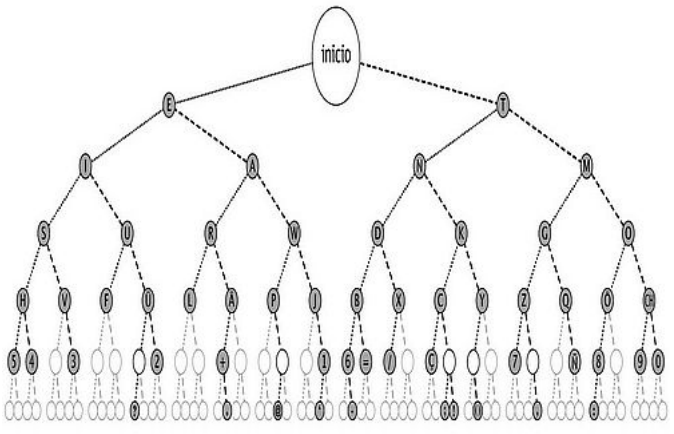
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

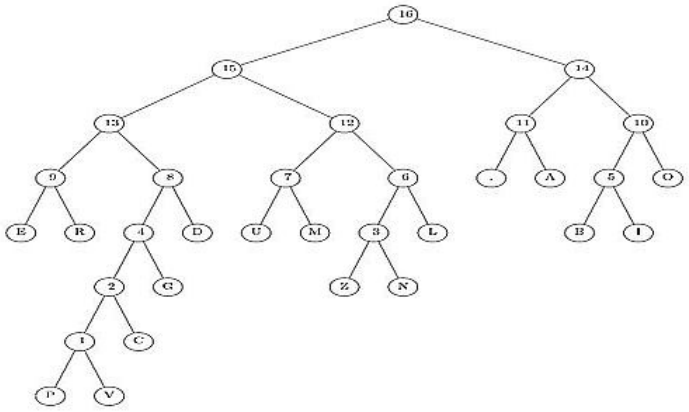
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
...1.1--1.-1-.1.-12
2)
-.-.1---1-...1.-.1.-1...12
3)
-1.1-.-.1.-..1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
3E 32 95 0E C0                          36 bits significativos
5)
6A 28 0F 88 B0                          38 bits significativos
6)
A5 E3 61 1B B0                          38 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76551

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76737 GRAZIELLY A. L. VALICOSKI
24CC2406 Exercício : 8 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

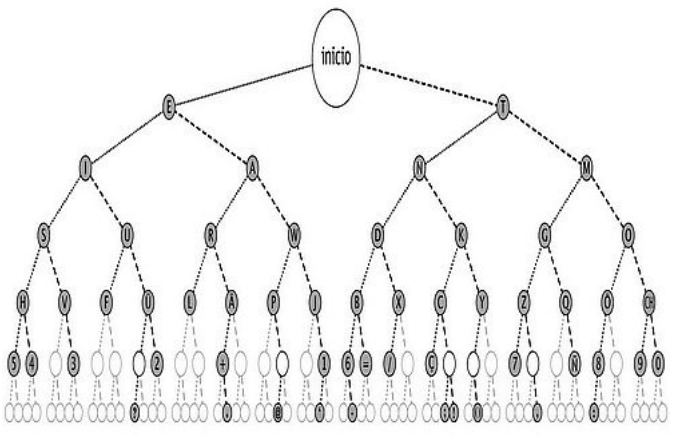
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

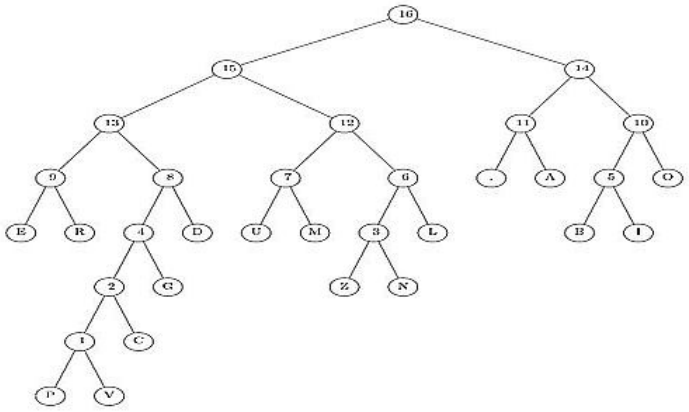
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.1...1-.-.1---1.-..1.-12
2)
--.1.-.1.-1--1.--.1---12
3)
-.-.1.-1.-..1-.-.1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
24 E6 52 6B 6C                          40 bits significativos
5)
3E 32 95 0E C0                          36 bits significativos
6)
E4 7C 21 CC 42 00                       41 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76737

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76568 GUILHERME VITOR DA SILVA
24CC2406 Exercício : 9 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

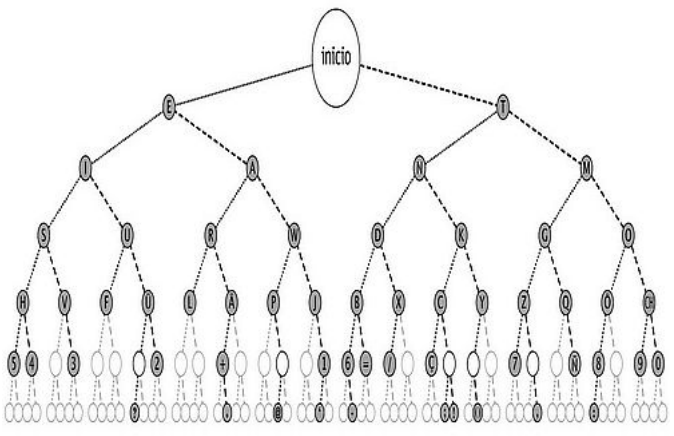
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

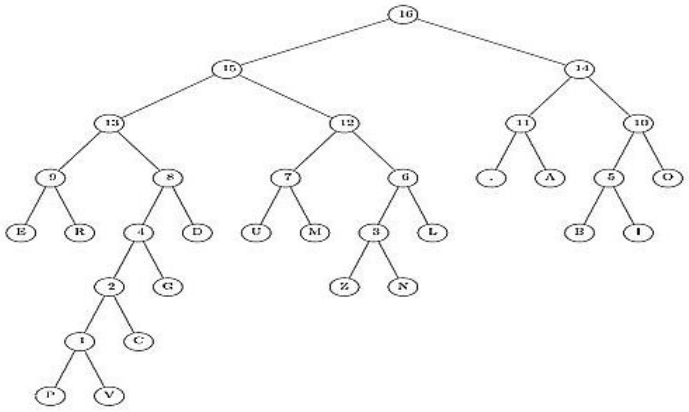
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
-.-.1....1.-1.--.1.1..-12
2)
.1...1-.-.1---1.-..1.-12
3)
-..1..1...1-.-.1---1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
20 06 3E 67 90                          38 bits significativos
5)
A5 E3 61 1B B0                          38 bits significativos
6)
3F 04 22 EC 80                          35 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76568

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76706 KAWAN DAIKI HASEGAWA
24CC2406 Exercício : 10 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

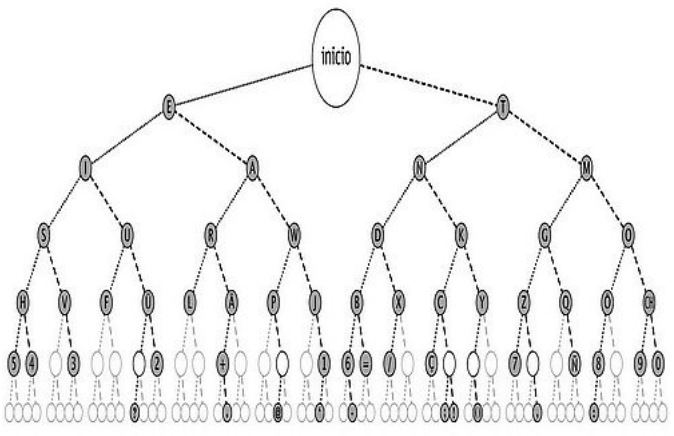
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

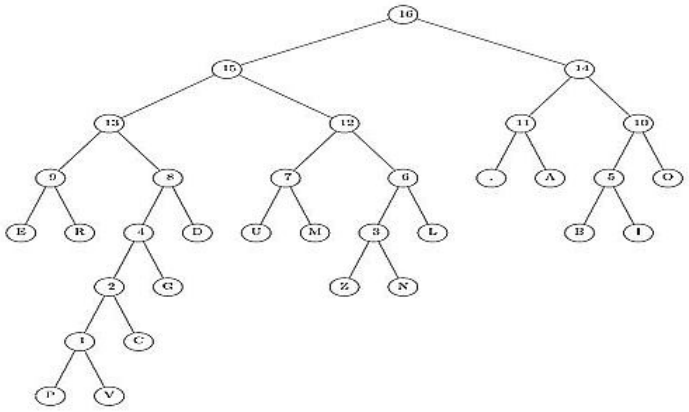
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
-.-.1.-1.-..1-.-.1.-1---12
2)
.-1--.1---1...1-1---12
3)
-..1..1...1-.-.1---1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
2A 0E CC EB 20                          37 bits significativos
5)
20 10 BC 4F C8                          39 bits significativos
6)
CE F6 72 35 E0                          37 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76706

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

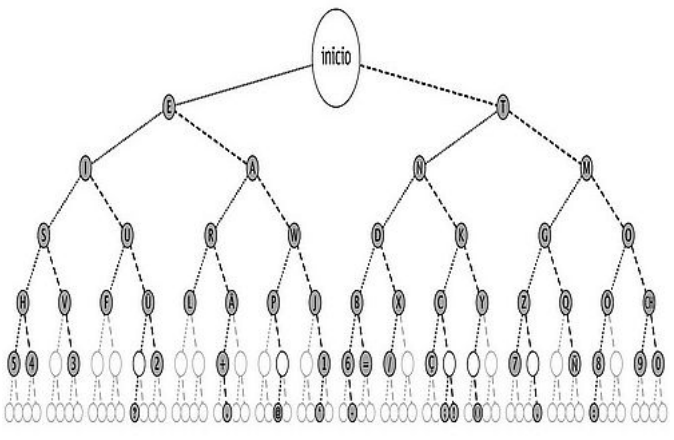
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

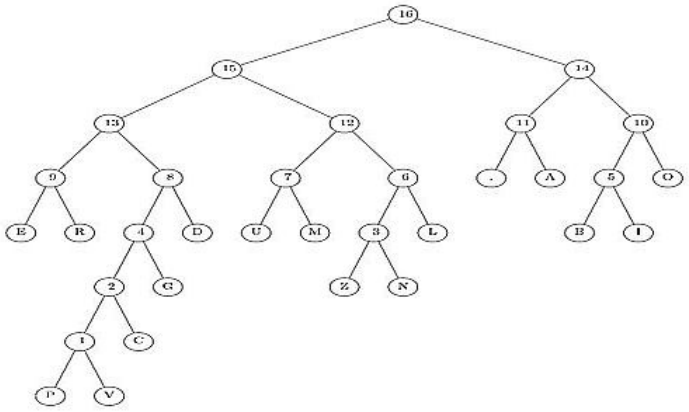
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
-.-.1..-1.-.1...1---1...12
2)
.-1--.1---1...1-1---12
3)
-.-.1---1-...1.-.1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
2A 0E D8 B6 40                          36 bits significativos
5)
CE F6 72 35 E0                          37 bits significativos
6)
5A 42 C1 4D 58                          39 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76575

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76694 LAURO HENRIK RUTZ
24CC2406 Exercício : 12 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

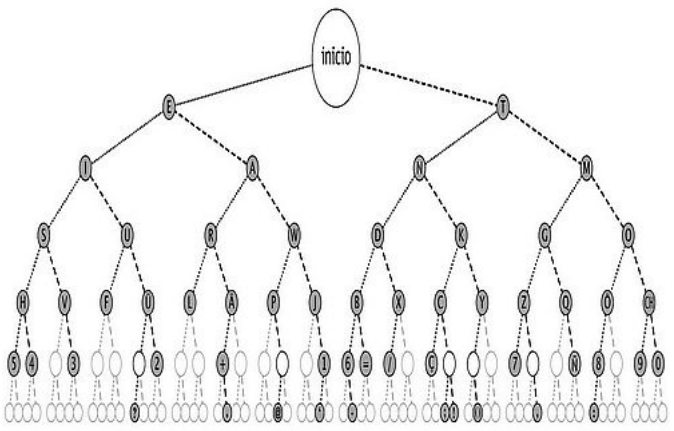
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

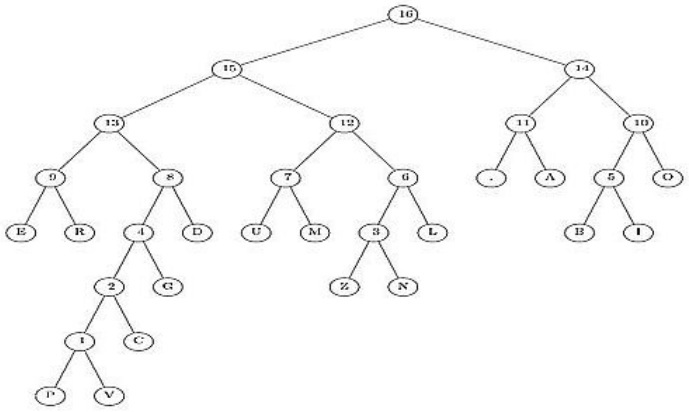
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.-..1..1...-1.-.1---1...12
2)
.--.1.-1...1-1.-1...12
3)
.-1--.1---1...1-1---12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
CE F6 11 01 30                          40 bits significativos
5)
3E 32 95 0E C0                          36 bits significativos
6)
A4 42 65 26 B6 80                       41 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76694

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

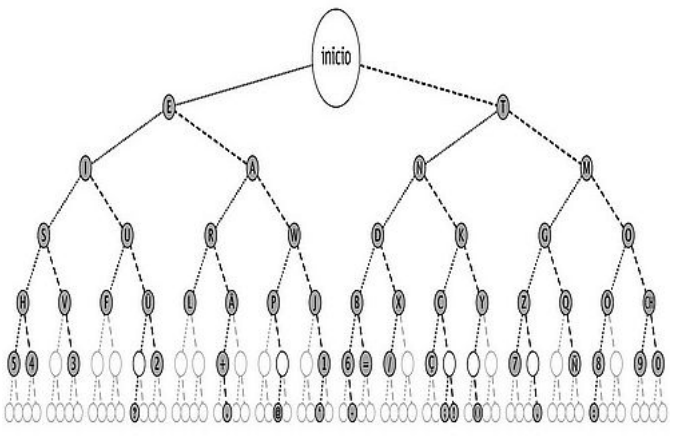
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

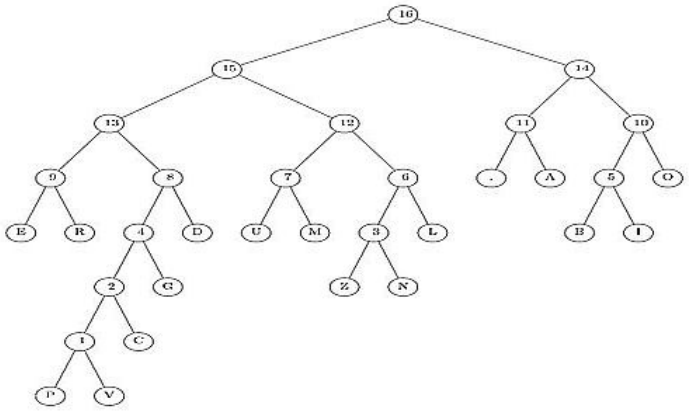
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
..-.1---1.-..1....1.-1...12
2)
-1.1-.-.1.-..1.-1...12
3)
...-1..1...1---1.1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
AC 47 84 82 48                          39 bits significativos
5)
20 06 3E 67 90                          38 bits significativos
6)
3F 04 22 EC 80                          35 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76582

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76599 NICOLE DA SILVA DE AZEVEDO
24CC2406 Exercício : 14 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

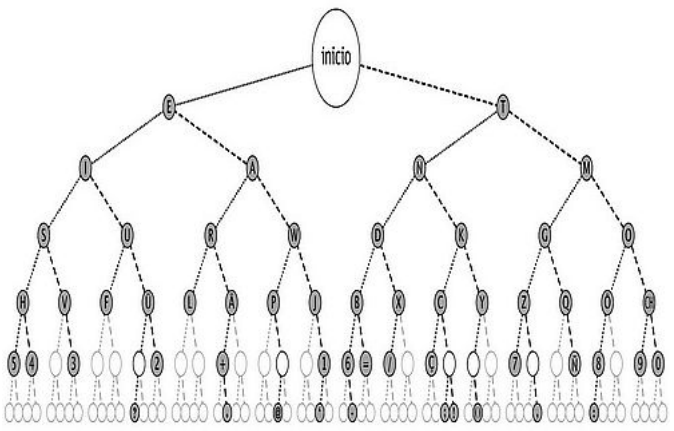
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

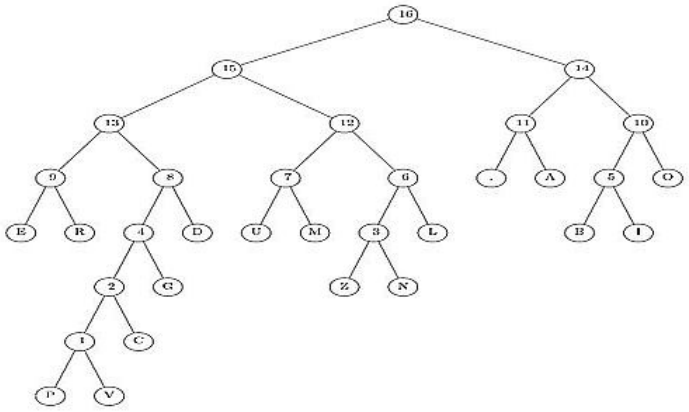
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.-..1.1-1.-.1.-1...12
2)
...1.1--1.-1-.1.-12
3)
.-1..-1.-.1---1.-.1.-12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
20 11 69 47 C8                          39 bits significativos
5)
3E 32 95 0E C0                          36 bits significativos
6)
AC 47 84 82 48                          39 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76599

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76601 PAULO JOSE FREIRE CORREA
24CC2406 Exercício : 15 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

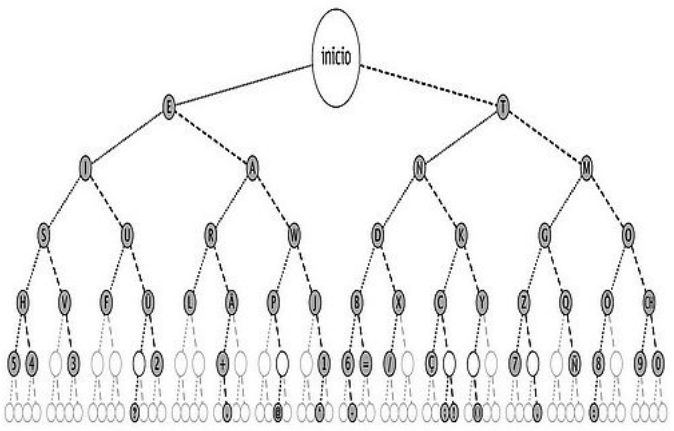
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

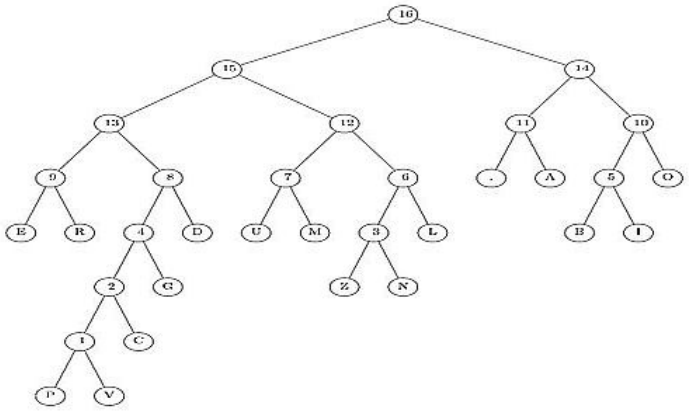
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.-..1..1...-1.-.1---1...12
2)
.-1--.1---1...1-1---12
3)
-.-.1.-1--1..1...1.-12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
5A 42 C1 4D 58                          39 bits significativos
5)
E4 7C 21 CC 42 00                       41 bits significativos
6)
20 10 40 FB 4F 80                       41 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76601

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76618 PEDRO HENRIQUE PIEKARSKI DE OL
24CC2406 Exercício : 16 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

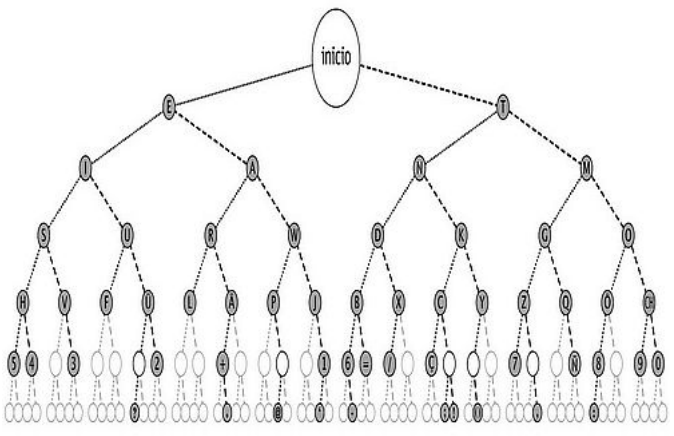
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

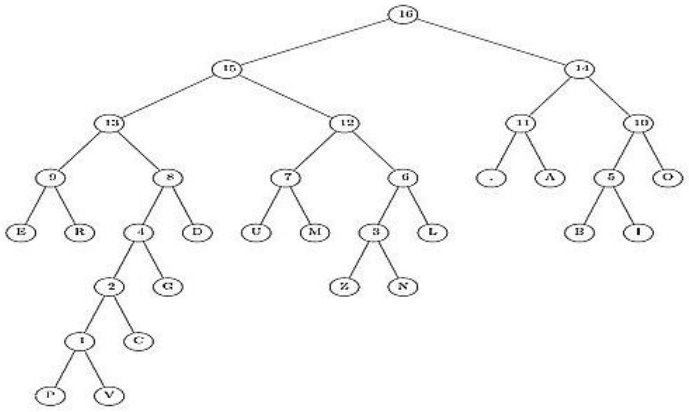
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O   .     C    A    O   .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
--.1.-.1.-1--1.--.1---12
2)
.1...1-.-.1---1.-..1.-12
3)
-1.1-.-.1.-..1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
26 8D 95 80 74                          38 bits significativos
5)
26 8D 95 80 7C                          38 bits significativos
6)
AC 47 84 82 48                          39 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76618

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

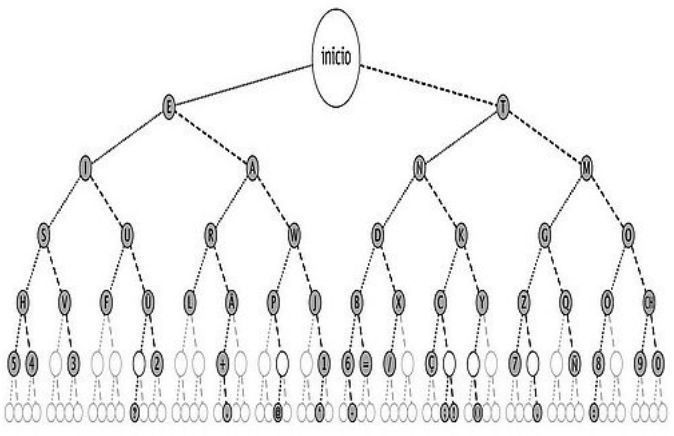
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

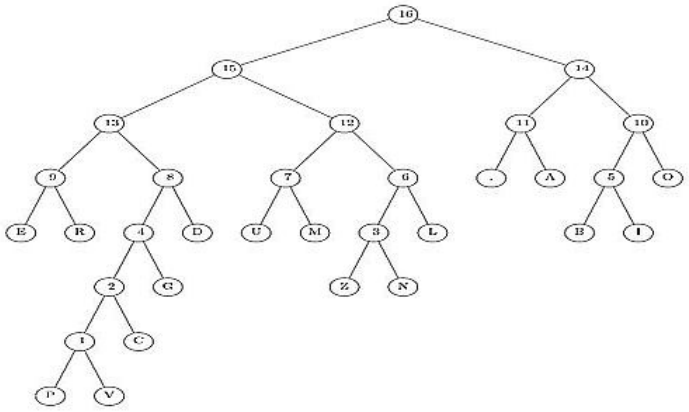
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.-1..-1.-.1---1.-.1.-12
2)
-.-.1.-1-.1.1-1.-12
3)
-1.1-.-.1.-..1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
6A 28 0F 88 B0                          38 bits significativos
5)
3F 04 22 EC 80                          35 bits significativos
6)
3E 32 95 0E C0                          36 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76625

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76632 VICTOR BRONDANI DOS SANTOS
24CC2406 Exercício : 18 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

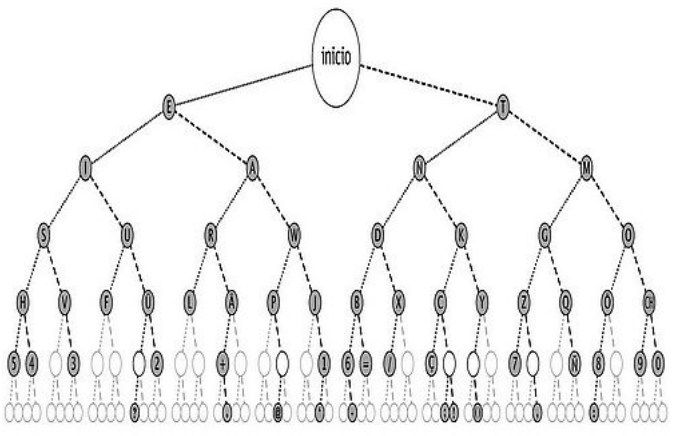
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

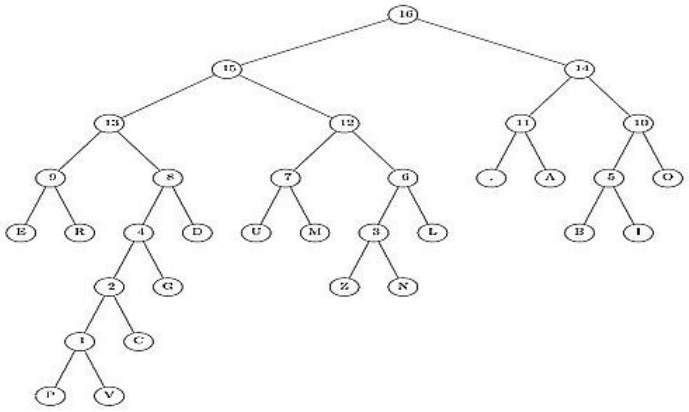
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
-.-.1....1.-1.--.1.1..-12
2)
-.-.1.-1.-..1-.-.1.-1...12
3)
.--.1.-1...1-1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
A4 42 65 26 B6 80                    41 bits significativos
5)
6A 28 0F 86 0C                       40 bits significativos
6)
CE F6 56 23 C0                       36 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76632

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76649 VICTOR HUGO DOS SANTOS DE CAMA
24CC2406 Exercício : 19 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

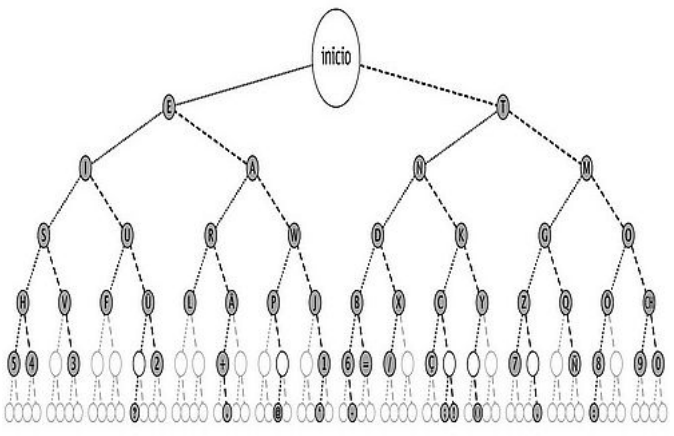
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

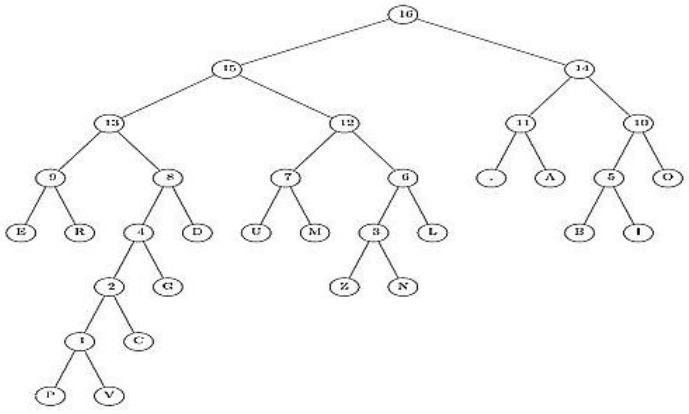
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O   .     C    A    O   .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.-1..-1.-.1---1.-.1.-12
2)
.-..1.1-1.-.1.-1...12
3)
.-..1..1...-1.-.1---1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
E4 7C 21 CC 42 00                    41 bits significativos
5)
CE F6 72 35 E0                       37 bits significativos
6)
20 06 3E 67 90                       38 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76649

UFPR-UP-UTFPR-PUC/Pr Sistemas de Informação 23/09/2024 - 09:27:11.7
Tópicos Avançados em SI Prof Dr P Kantek (pkantek@gmail.com)
Compressão de dados - VIVXq82a, V: 1.04
76656 VITOR EDUARDO DE GOES FONTES
24CC2406 Exercício : 20 limite entrega 10/10/24 _____ / _____ / _____

Compressão de dados

A compressão de dados visa diminuir a redundância de dados. Logo apenas conjuntos de dados que tenham redundância podem ser comprimidos com vantagem. A compressão pode se dar em camadas: a. aplicativo: o aplicativo controla. Exemplo: uma tabela de cidades; b. do ambiente: o sistema operacional aplica. Para o aplicativo nenhuma mudança surge. Exemplo: PKZIP; c. Física: o hardware comprime. Exemplo: compressão feita pelo modem.

Os benefícios esperados são: a. diminuição de espaço e de tráfego. b. criptografia; c. melhoria de uso da CPU (já que os aplicativos em geral são IO bound); d. ultrapassar limites de mídias (exemplo um DVD). Certos arquivos são altamente propícios à compressão, por causa do fenômeno de "localidade de referência". Exemplos: música digital, vídeo...

Conceitos: Razão de compressão: tamanho descomprimido ÷ tamanho comprimido.

Prática Vamos examinar um arquivo real:
Nome:`dira` (um diretório) . Tamanho: `29.009.582` Compressão rápida (`7z -mx1 a lixo dira`), Tamanho: `4.475.057`, Razão: 0.154 Compressão forte (`7z -mx9 a lixo dira`), Tamanho `3.093.879` (demora mais). Razão: 0.106

Agora uma imagem JPG: (Não se esqueça que o padrão JPG já faz uma compressão e tanto...)
Nome: `reca.jpg`. Tamanho: `130.913` Compressão normal (`7z a lixo reca.jpg`), Tamanho `115.352`. Razão: 0.8811 Compressão forte (`7z -mx9 s lixoa reca.jpg`), Tamanho `115.336`. Razão: 0.8810.

Código Morse Só aparece aqui por razões históricas. O código morse é um sistema de representação de letras, números e sinais de pontuação através de um sinal codificado enviado intermitentemente. Foi desenvolvido por Samuel Morse e Alfred Vail em 1835, criadores do telégrafo elétrico (importante meio de comunicação à distância), dispositivo que utiliza correntes elétricas para emissão ou recepção de sinais. Veio na continuação do telégrafo visual em uso na França desde o século XVIII.

Uma mensagem codificada em Morse pode ser transmitida de várias maneiras em pulsos (ou tons) curtos e longos:

- pulsos elétricos transmitidos em um cabo;
- ondas mecânicas (pertubações sonoras);
- sinais visuais (luzes acendendo e apagando);
- ondas eletromagnéticas (sinais de rádio);

Este sistema representa letras, números e sinais de pontuação apenas com uma seqüência de pontos, traços, e espaços.

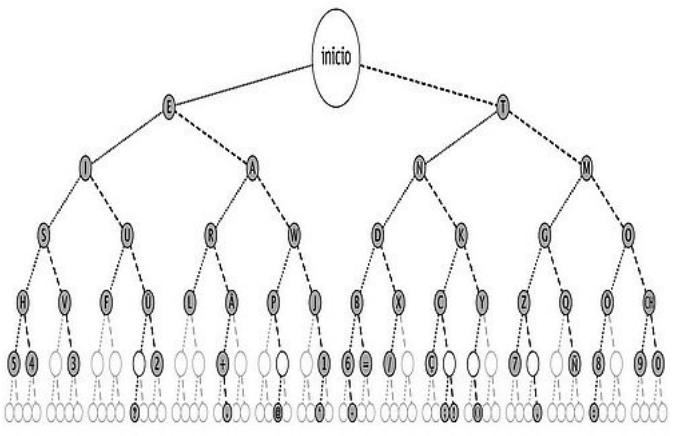
Com o desenvolvimento de tecnologias de comunicação mais avançadas, o uso do código morse é agora um pouco obsoleto, embora ainda seja empregado em algumas finalidades específicas, incluindo rádio faróis, e por CW (continous wave-ondas contínuas), operadores de radioamadorismo. Código morse é o único modo de modulação feito para ser facilmente compreendido por humanos sem ajuda de um computador, tornando-o apropriado para mandar dados digitais em canais de voz.

Os sinais são:

- Sinal curto, ponto ou 'dit' (.)
- Sinal longo, traço ou 'dah' (-)
- Intervalo entre caracteres (entre pontos e traços). Aqui é 1.
- Intervalo curto (entre letras). Aqui é 2.
- Intervalo médio (entre palavras)
- Intervalo longo (entre sentenças)

Como o comprimento variável de caracteres do código morse dificulta a adaptação à comunicação automatizada, o Morse foi amplamente substituído por formatos mais regulares, incluindo o Código Baudot e ASCII.

Para construir a tabela morse, usa-se a busca dicotômica na seqüência, com ponto à esquerda e traço à direita. E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,Ü,L,Ä,P, J,B,X,C,Y,Z,Q,... Acompanhe na imagem:



Exemplo Traduza para o português:
`...-1..1...-1.-12.1..-12.` Resposta: _____.
(VIVA EU)

Comprimento de fileira Escolhe-se um caracter pouco frequente (no nosso exemplo será Z). Sempre que um conjunto de mais de 3 caracteres se repetir, todos eles serão substituídos pelo conjunto Z+carater a transmitir +quantidade de repetição (com um único número).

No exemplo A.....DIVIDA.....E.DE.0000000033.REAIS, ficaria AZ.5DIVIDAZ.7E.DE.Z0833.REAIS

Observações: 1.naturalmente, se o caracter de contagem for um byte binário, o contador poderá chegar até 258, pois ele pode começar em 4, neste caso, reservando-se o '0' para a ocorrência simples do caracter.

Se por acaso, o caractere de controle existir no arquivo, ele pode aparecer na saída como duplicado, com tamanho 1.

Exemplo: A.....LUZ.E.BRANCA ficaria AZ.7LUZZ1.E.BRANCA

Exemplo Comprima usando esta técnica:
`AAABBBBBBB0000000574-REAIS`. Resposta: _____.
(AAAZB7Z07574-REAIS)

Substituição de padrões A substituição de padrões é –em geral – um método de compressão de aplicativo, já que é dependente dos padrões da aplicação. É muito usada em compressão de códigos fonte (os padrões são as palavras reservadas) ou em grandes sistemas de dados comerciais (nomes de pessoas, de ruas, de cidades, de conjuntos de zeros, CEPS, etc).

Nestes casos há uma tabela com os padrões mais freqüentes (256, 65536,...) que são convertidos para 1 ou 2 bytes. Se o arquivo for binário, não há este limite, pode-se usar qualquer tamanho de tabela de padrões ($2^9, 2^{10}, 2^{11}$...). Apenas há que se sinalizar a existência do padrão comprimido. Pode ser o sempre presente caracter de controle acima visto.

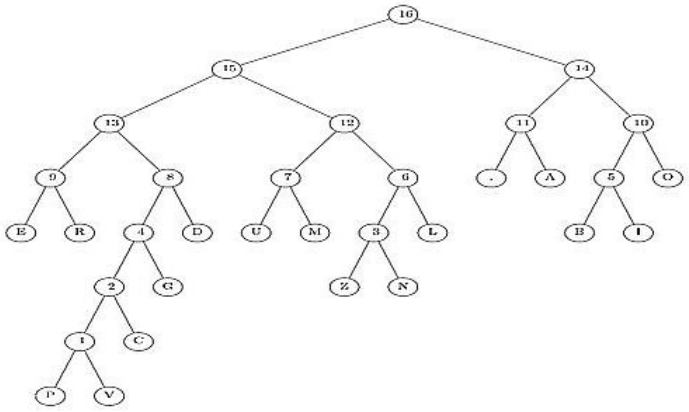
Huffman A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". As etapas do algoritmo são:

- Análise do objeto (arquivo, imagem, etc..) a comprimir
- A construção de um dicionário
- A construção de uma árvore binária, sempre a partir dos caracteres mais frequentes
- Compressão do objeto, percorrendo a árvore para cada caracter e recolhendo os bits no trajeto da árvore
- Construção da saída, a partir dos bits acima.

Supondo o dicionário:

```
P0010000                U0100
V0010001                M0101
C001001                 D0011
Z01100                  E0000
N01101                  R0001
G00101                  O111
B1100                   .100
I1101                   A101
L0111
```

Constrói-se a árvore



Exemplos Descomprima:
`26 98 68 D8 B4` (38 b.s.) Resposta: _____.

```
   2    6    9    8    6    8    D    8    B    4
0010.0110.1001.1000.0110.1000.1101.1000.1011.0100
     C    A    D    E    I    R    A    .    M   A//
```

Mais um
`F0 9B E6 6B A0` (37 b.s.) Resposta: _____.

```
   F    0    9    B    E    6    6    B    A    0
----.----.----.----.----.----.----.----.----.----
  O    .    C    A    O    .    B    I    L   U///
```

Mais um
`66 8E 95 34 D1 40` (42 b.s.)

A resposta e ZIRIGUIDUM.

☞ **Para você fazer** Traduza para português, as mensagens morse

```
1)
.--.1.-1.--.1.1..1...12
2)
.-1--.1---1...1-1---12
3)
.--.1.-1...1-1.-1...12
```

Descomprima as mensagens Huffman, usando o dicionário acima:

```
4)
AC 47 95 BA F0                          38 bits significativos
5)
2A 0E D8 B6 40                          36 bits significativos
6)
20 11 69 47 C8                          39 bits significativos
```

Codificação Relativa Este tipo de compressão se aplica apenas em casos particulares, mas quando se aplica ele é muito bom. Novamente é uma aplicação do conceito de localidade de referência. Enviado (ou gravado) o registro inicial, para os subseqüentes serão enviadas apenas as alterações em relação ao elemento anterior. Não se pode colocar aqui um esquema geral, fica apenas o conceito. Imagine-se por exemplo uma aplicação que faz captura remota de dados de telemetria. Este esquema é usado no compactador campeão que é da Intel e comprime imagens em movimento, alcançando razão de compressão de 350:1.

Compressão de Shannon-Fano Similarmente ao código Huffman, esta compressão gera cadeias de bits de comprimento variável para cada caracter. Para sua implementação devemos conhecer o arquivo a transmitir, e a partir dele, construir a tabela de freqüência de caracteres, em ordem decrescente.

Supondo, o seguinte conjunto FEEDDDCCCBBBBBAAAAAA. Isto geraria as seguintes freqüências:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Agora, o conjunto é divido em duas partes, tão logo a soma acumulada de probabilidades **ultrapassa** 50%. Ficaria

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%

Atribui-se a primeira metade, o dígito 0 e à segunda, o dígito 1. Fica:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1

Procede-se igualmente na segunda metade, e assim por diante, ficando:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
0	0	1	1	1	1
		0	0	1	1
				0	1

E temos então:

A = 00	B = 01	C = 100	D = 101	E = 110	F = 111

A tabela do processo Shannon-Fano pode ser ambígua, já que dificilmente teremos uma divisão em 50% exatos. Em alguns casos sua eficiência será igual a de Huffman (que é a eficiência ótima), podendo entretanto ser pior.

Exemplo Descomprima 85 71, com 16 bits significativos e universo='LLLLLAAAAOOOHH'. Resposta: _____. O dicionário: A=00, B=01, C=100, D=101, E=110, F=111.

Código de vírgula No código Huffman e no Shannon-Fano, temos uma dificuldade adicional, principalmente em transmissão remota de arquivos. Se um único bit for perdido no meio do string, perde-se toda a recuperação posterior do arquivo. Diz-se neste caso que foi-se o sincronismo entre o código e o processo. Para evitar este inconveniente, pode-se utilizar um terminador de caractere, (usualmente o bit 1 isoladamente). Naturalmente este código não seria fornecido a nenhum caracter, e com isto perde-se eficiência na compressão, que no entanto é compensada pela possibilidade de perda de bits sem a perda adicional. Agora, os caracteres são conhecidos pela quantidade de zeros, como por exemplo:

A=30%	B=25%	C=15%	D=15%	E=10%	F=5%
1	01	001	0001	00001	000001

Este código usa um conceito de separação similar ao do código Morse.

Exemplo Descomprima 48 C2 0C, com 22 bits significativos e universo = 'AAAAAAOOOOOLLLLHHH..C'. Resposta: _____.

Compressão auto-adaptável As técnicas até agora vistas pressupõe um conhecimento prévio sobre o universo de caracteres a comprimir. Mas como utilizar estas técnicas quando nada se sabe sobre o arquivo a comprimir ? Para dar resposta a esta questão surgiram as técnicas auto-adaptáveis, que garantindo um comprimento variável de bits por byte (eficiência) prescindem de uma análise anterior no arquivo.

Vamos imaginar uma tabela de caracteres a transmitir (pode ser uma tabela de todo o código ASCII, por exemplo)

A	B	C	D

Como nada sabemos sobre a ocorrência de caracteres, vamos atribuir códigos de tamanho variável, sem qualquer critério. Pode ficar:

caracter	A	B	C	D
codificação	1	01	001	0001

Vamos acrescentar a esta tabela um contador de caracteres processados,

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	0	0

e agora, vamos ver como esta técnica se comporta

Suponhamos que o primeiro caracter que chega para ser enviado é um caracter "C". O algoritmo consulta a tabela, manda o código "001", que é o que NO MOMENTO corresponde a "C", e em seguida incrementa o contador de "C", e a tabela fica

caracter	A	B	C	D
codificação	1	01	001	0001
contador	0	0	1	0

A seguir, a tabela é ordenada, e temos:

caracter	C	A	B	D
codificação	1	01	001	0001
contador	1	0	0	0

Digamos que agora chegue um caracter D. O mesmo é enviado como 0001, o contador é incrementado, e a tabela reordenada. Fica

caracter	C	D	A	B
codificação	1	01	001	0001
contador	1	1	0	0

Agora chega de novo um caracter "C". Ele é enviado como 1, pois agora o C ocupa outra posição na tabela. Seu contador, obviamente é incrementado.

E o processo flue assim por diante. Do lado do receptor, a tabela de início é igual a do transmissor. Cada caracter que chega é decodificado com a tabela atual, e um contador também é incrementado, seguindo-se a uma reordenação da tabela. Assim procedendo, as duas tabelas são reordenadas sincronamente, e com isso a compressão é feita com sucesso.

Note-se que neste exemplo, empregou-se um código de vírgula ao invés de um dicionário à la Huffman ou mesmo Shannon-Fano. Ao escolher a vírgula o algoritmo ficou mais fácil de explicar e entender. Entretanto ele é menos eficiente.

Algoritmo LZW As iniciais de seus autores: Lempel, Ziv e Watch este algoritmo agrupa caracteres que costumam aparecer juntos. Ao invés de usar 8 (ou menos) bits por caracter, ele usa 12, 14 bits para representar grupos de n caracteres que sejam comuns. Tem desempenho ótimo. É usado no padrão GIF de imagens.

Responda aqui

:

1
2
3
4
5
6

76656