Programação Dinâmica

Para muitos problemas quando a força bruta resulta inaceitável, a técnica da programação dinâmica pode ser uma alternativa. Esta técnica permite construir algoritmos que resolvem problemas computacionais, sobretudo na área de otimização combinatória. Tais problemas precisam ter 2 características para poderem usar a técnica:

- subestrutura ótima: quando a solução ótima contém em seu interior soluções ótimas para subproblemas e
- superposição de problemas: ocorre quando o algoritmo de força bruta reexamina o mesmo problema muitas vezes.

Exemplos de problemas que podem ser resolvidos usando programação dinâmica: multiplicação de séries de matrizes, problema da mochila, otimização combinatória, etc. Os problemas desta folha são os problemas 67, 81 e 82 do projeto Euler.

Maior soma em pirâmide

Suponha uma matriz como

```
   3
  7 4
 2 4 6
8 5 9 3
```

Precisa-se calcular a soma máxima começando no vértice superior e descendo sempre para os 2 números adjacentes. Neste caso é fácil ver que a soma máxima é 3+7+4+9=23. Se a matriz tiver 15 linhas (e 15 números na última linha), já serão 16384 possíveis caminhos: o computador ainda os percorre muito rápido. Já em uma matriz de 100 linhas, se seu programa testar um trilhão de caminhos por segundo, serão necessários vinte bilhões de anos para testar todas as possibilidades.

Começando no alto do triângulo acima (no 3), deve-se escolher ir para a direita ou a esquerda. Para decidir isto, precisa-se resolver 2 problemas que são:

```
  7                4
 2 4       e       4 6
8 5 9             5 9 3
```

Pode-se ir quebrando a série de problemas até a última linha, cuja resposta é evidente: é o número que está lá. Resolvido, pode-se retornar para cima, usando a regra

```
 a
b c
```

$a + max(b, c)$. Entretanto, ao quebrar o problema original em 2, e cada um deste em outros 2, fica-se com 4 problemas. (As pirâmides: 7,2,4; 4,4,6 se quebram em 2,8,5; 4,5,9 e em 4,5,9; 6,9,3). Note que há problemas que se repetem (4,5,9 no exemplo). Ora, eles vão ter o mesmo resultado não importando como se chegou a eles. Logo, basta calcular tais problemas uma vez, e aqui a programação dinâmica mostra seu valor.

o algoritmo Começando de baixo para cima e aplicando a regra $a + max(b, c)$, fica

```
     3
    7 4
   2 4 6
  8 5 9 3

fica

       3
     7   4
  10  13   15

e depois

    3
  20 19
```

E finalmente 23, que é a resposta correta.

o algoritmo Neste caso o algoritmo é (origem=1):

```
1: leia(mat)
2: i ← número de linhas da mat
3: enquanto (i > 0)
4:    j ← 1
5:    enquanto (j ≤ i)
6:       mat[i][j]← mat[i][j]+max(mat[i+1][j],
         mat[i+1][j+1])
7:       j++
8:    fim{enquanto}
9:    i- -
10: fim{enquanto}
11: imprima mat[1][1]
```

Menor soma em matriz

Suponha a matriz

```
131  673  234  103   18
201   96  342  965  150
630  803  746  422  111
537  699  497  121  956
805  732  524   37  331
```

Começando no alto à esquerda (no 131) e podendo apenas descer 1 linha ou andar à direita uma coluna, qual a menor soma que se pode obter ao chegar no último elemento à direita (o 331) ? Neste problema é fácil ver que a mínima soma é 131+201+96+342+746+422+121+37+331=2427. Mas, numa matriz quadrada de ordem 80, fica impossível a força bruta (serão 9×10^{46} caminhos). Para usar a programação dinâmica, vai-se quebrar o problemão em probleminhas e depois juntar tudo. O ponto chave, é que se você estiver no final do problema, digamos: (o 188 é o número da última linha e última coluna da matriz)

```
270   312
416   188
```

Ao sair do 270 você deve escolher à direita ou para baixo: A decisão pode ser vista em (500=188+312 e 604=188+416)

```
270   500
604   188
```

Desta maneira a menor soma desde 270 até 188 é 270+min(500,604) que é 270+500=770. Usando esse raciocínio em toda a matriz, tem-se o algoritmo (origem=1):

```
1: leia mat
2: i ← ordem da matriz
3: enquanto (i > 0)
4:    mat[ord][i]← mat[ord][i]+mat[ord][i+1]
5:    mat[i][ord]← mat[i][ord]+mat[i+1][ord]
6:    i ← i - 1
7: fim{enquanto}
8: i ← ordem - 1
9: enquanto (i > 0)
10:    j ← ordem - 1
11:    enquanto (j > 0)
12:       mat[i][j]← mat[i][j]+min(mat[i+1][j],
          mat[i][j+1])
13:       j ← j - 1
14:    fim{enquanto}
15:    i ← i - 1
16: fim{enquanto}
17: escreva mat[1][1]
```

A mínima soma na matriz

Saindo de qualquer elemento da primeira coluna e chegando em qualquer elemento da última coluna, qual a soma mínima, podendo subir, descer ou andar à direita ? Por exemplo, em

```
131  673  234  103   18
201   96  342  965  150
630  803  746  422  111
537  699  497  121  956
805  732  524   37  331
```

é fácil ver que a menor soma é 201+96+342+234+103+18=994. No entanto, em uma matriz de ordem 80, não é mais tão fácil achar a resposta. A solução parte da premissa de que como não é permitido andar à esquerda, pode-se quebrar a matriz em colunas e resolver cada uma. Assim, o modelo da programação dinâmica continua válido.

o algoritmo (origem=1):

```
1: leia mat
2: int sol[ord]
3: sol ← última coluna de mat
4: i ← ord - 1
5: enquanto (i > 0)
6:    sol[1]← sol[1]+mat[1][i]
7:    j← 2
8:    enquanto (j ≤ ord)
9:       sol[j]← min(sol[j-1]+mat[j][i],
         sol[j]+mat[j][i])
10:      j ← j + 1
11:   fim{enquanto}
12:   j ← ord - 1
13:   enquanto (j > 0)
14:      sol[j]← min(sol[j], sol[j+1]+mat[j][i])
15:      j ← j - 1
16:   fim{enquanto}
17:   i ← i - 1
18: fim{enquanto}
19: imprima o menor valor de sol
```

Exemplos

Depois que você criar e testar os algoritmos com as matrizes acima, e antes de executar esta folha às veras, teste-os com 3 exemplos cujos nomes são EXEF00E1, EXEF00E2 e EXEF00E3 cujas respostas esperadas são: 73218, 42729 e 27535. A propósito, as respostas corretas da sua instância darão números parecidos a estes. Peça estes arquivos para o professor.

☞ Para você fazer

Os dados referentes aos problemas estão em arquivos magnéticos que deverão estar publicados no portal na aula referente ao dia em que a folha for distribuída, ou deverão ser disponibilizados aos alunos em qualquer outro meio magnético.

1. Exercício da máxima soma em pirâmide. Use o arquivo

   `XXXF01E1`

   que é uma matriz de ordem 100 parcialmente zerada (para simular uma pirâmide) e que contém em cada linha, 100 números menores do que 1000 e separados por um ou mais brancos. A linha ao final contém os caracteres CR+LF.

2. Exercício da mínima soma em matriz da esquerda superior até a direita inferior. Use o arquivo

   `XXXF01E2`

   que é uma matriz de ordem 80 que contém em cada linha, 80 números menores do que 1000 e separados por um ou mais brancos. A linha ao final contém os caracteres CR+LF.

3. Exercício da mínima soma em matriz da esquerda até a direita. Use o arquivo

   `XXXF01E3`

   que é uma matriz de ordem 80 que contém em cada linha, 80 números menores do que 1000 e separados por um ou mais brancos. A linha ao final contém os caracteres CR+LF.

1	2	3