A Escola do Futuro

P. Kantek

17 de maio de 2020

Antes de tudo

```
      (+/a)÷⍴a←⍞
⍞:
      1 2 3 4
2.5
```

A idéia

No dia 10 de abril de 2020, a emissora pública inglesa, BBC (no seu site bbc.com [1]) publicou um longo artigo intitulado *Should we teach children about quantum computing ?* no qual a reporter Kathryn Kyte descreve a *The Knowledge Society* (TKS), uma escola para adolescentes que dá a eles a chance de aprenderem o que não é visto na escola tradicional. Há um depoimento de um rapaz de 15 anos, Jack McDonald, que diz: *Na minha escola regular não se fala sobre criptografia ou computação quântica, isto não faz parte do curriculo, assim durante muito tempo tive que investir para aprender tais coisas por mim mesmo.* O site da TKS é `https://theksociety.com/`

A escola TKS foi declarada como uma das Escolas do Futuro pelo think tank *World Economic Forum* e ela oferece educação e treinamento para alunos entre 13 e 17 anos interessados em inteligência artificial e outros assuntos em nichos de tecnologia que raramente são tratados na escola regular.

As classes na TKS tem cerca de 40 alunos e são dadas em 2 dias por semana (aos fins de semana) em sessões de 3 horas cada dia. O programa de 10 meses não é barato e custa entre 5 e 8 mil dólares para o ano acadêmico 2020-21 dependendo em que cidade ele é fornecido. Atualmente o TKS é oferecido em diversas cidades da América do Norte incluindo Boston, Seattle, Los Angeles e Toronto e é esperada sua expansão para Londres e América Latina em 2021, sendo que os cursos no momento estão sendo dados online por conta da interrupção do coronavirus.

O programa da TKS 2020-21 abrange 40 áreas de foco que inclui aprender sobre impressão 3D, biônica ou eletricidade sem fio, com o programa completo demorando 3 anos.

Ao ler sobre este tema, pensei na transposição imediata dessa escola para a nossa realidade. A sensação de que a escola tradicional deixa a desejar é mais ou menos comum entre nós. Certamente não quando ela ensina matemática (aprender a pensar) e linguagem (aprender a se comunicar), nem quando habilita as pessoas a trabalhar em grupo e de forma complementar. Mas, mesmo tendo estes fatos em mente, salta aos olhos que a nossa escola tradicional arrisca a produzir um profissional para os anos passados e não para os anos futuros.

As mudanças no mercado de trabalho, nos contratos e nas formas de remuneração, na interação da TI, nas exigências de expertise são evidentes e importantes. Ainda agora, no episódio do coronavirus, há uma sensação de tais mudanças acima apontadas vão vir ainda mais depressa e de maneira avassaladora.

Então a minha proposta: vamos montar uma escola baseada na TKS entre nós ? Na falta de nome melhor chamei-a de **Escola do Futuro** pois é isso que ela é. Sem eliminar a escola tradicional, ela agrega um novo curriculo. Não é para todos, pois exige muito em troca: comprometimento, tempo, suor, horas cadeira de estudo e recursos financeiros.

Escola do Futuro

Uma instituição que reune poucos alunos interessados em desenvolver habilidades que serão necessárias no século XXI e além. Não substitui a escola tradicional, mas é complementar a ela. Tem uma carga horária presencial de 3h/semana, mas exige dos participantes pelo menos o dobro disso em casa. Possui uma aderência importante ao computador e à programação, já que parte do pressuposto que a habilidade mais importante a desenvolver é a intimidade com essa tecnologia, associado ao desenvolvimento do saber pensar, planejar e construir para essa plataforma.

Precisa um local confortável com acesso a internet em larga capacidade e no qual cada aluno tenha a sua plataforma de trabalho, mas idealmente, cada aluno deve trabalhar em seu próprio computador, que o acompanhe na escola e em casa.

As aulas presenciais são espaços de construção. Ao final de cada aula (ou vá lá, de um ciclo de aulas) o aluno terá o seu próprio engenho, construído e mantido por ele. Aqui dá-se a palavra a Aristóteles, em seu livro Ética: *o que precisamos aprender, aprendemos... fazendo.* Este ponto é inegociável: Cada aluno ao chegar em casa, depois de uma aula, deverá poder mostrar a si e aos pais o que aprendeu naquele dia. Esta é outra diferença importante em relação à escola tradicional. A nossa Escola não tem espaço para enganação em nenhum dos 2 sentidos: nem do aluno em relação ao professor, mas tampouco do professor em relação ao aluno. Sejamos honestos quanto a isto.

É óbvio que o trabalho em grupo é recomendado, até para que interesses e habilidades complementares se somem, mas é comportamento esperado de todos os alunos uma certa independência e uma capacidade de entrega de tarefas e projetos: a Escola trabalha e encoraja esta capacidade.

[1] para ver o artigo original, vá ao site da BBC e peça na máquina de busca: "quantum computy". O artigo deve aparecer nas primeiras posições.

A Escola é exigente: as tarefas precisam ser entregues e são avaliadas sem paternalismo. Não há pressa, os prazos são negociáveis, mas atenção: qualquer espaço de enrolação ou de encostamento é rapidamente cortado, sob pena de a Escola ser apenas mais um caça-níqueis de tantos que existem por aí. Se – ao agir assim – ficarmos sem alunos, paciência, terá sido uma boa iniciativa.

Público

Crianças, adolescentes e jovens, que tenham

- Bom histórico escolar, sobretudo em matemática e linguagem e que estejam cursando ao menos a 7^a série.
- Interesse em aprender a programar ou se desenvolver nisso, acesso a plataforma computacional.
- Disposição para trabalho (intelectual) pesado. Para efeito de divulgação, a escola pode pretender ocupar 3 tardes por semana do interessado. 1 na escola e 2 em casa – no mínimo.
- Curiosidade e sede de saber.

A escola deve enfatizar que não existe desenvolvimento sem esforço. Atenção que este discurso é um pouco remar contra a maré. Nos dias de hoje dissemina-se o discurso do crescimento sem investimento e do resultado quase de graça. Nada mais falso, e este recado precisa ser mostrado (e praticado) a todo instante.

Até pela faixa etária do público, não há pré-requisitos formais. Isto ajuda a ampliar a abrangência da iniciativa, mas acrescenta complexidade à abordagem: há que se falar no estado-da-arte em tecnologia, sem exigir nenhuma condição prévia.

Operação

Precisamos formatar a iniciativa, buscando patrocinador, local, administração, e recursos mínimos de infraestrutura. Insisto na pequena escala (1 turma no máximo, com até 40 alunos), até para apreendermos como a coisa funciona. O grupo de professores precisa ser escolhido bem: as aulas têm que ser 100% cativantes. Não se pode esquecer que se a coisa ficar chata, os alunos debandarão e a iniciativa morrerá. Pensei em convidar o Leandro Henrique, nele eu ví este fogo nos olhos ao falar da tecnologia. Pensei também na Isabelle, idem idem. Modestamente, eu me incluo nesta lista. A dificuldade em achar os professores é óbvia: precisamos de gente que pense à frente e não atrás, que é como as nossas escolas de mestrado e doutorado ensinam hoje.

Apenas para efeito de lançamento da Escola, pensei em uma palestra, convite livre, sem custos, abordando a história de Alan Turing e da máquina ENIGMA, com um desafio (o que está escrito na mensagem YXRHWLNWKHLZGWOPHZIJZI ?). Já dei essa palestra muitas vezes e o resultado em geral é impactante.

Na sequência deste texto, listei os assuntos dos quais posso falar. É uma amostra do que a Escola pode apresentar. Outros temas, aderentes ao time de instrutores podem e devem ser incluídos. Note-se como a lista é dependente das pessoas que vão conduzir as reuniões. Acho que – de início – deveríamos elaborar uma programação para o primeiro ano da Escola, na forma de 32 encontros. Há bastante material para trabalho. Lembrando que para cada tema, não pode haver nenhum pré-requisito formal, estamos falando de adolescentes de 15 anos. (Talvez a única exceção aqui seja a habilidade de programar: mas essa é tranquila para o nosso aluno padrão).

Cada tema precisa ter como apoio um (bom) material escrito, em papel ou em PDF, tanto faz. Não uma palestra PPT, mas sim um texto em bom português, de autoria do instrutor. Ainda que se apropriem materiais na Internet, (estamos no século XXI), a base tem que ser um bom material de autoria.

Na lista a seguir, há muitas referências a material de minha autoria do pacote Algoritmos Vivos, que pode ser integralmente visitado em `algoritmovivo.com`.

Lista de assuntos a tratar

Programando como respiramos Desenvolver a habilidade de programar como pré-requisito para todas as demais atividades. Os alunos são levados a construir programas, começando com exigências crescentes e escalonadas. A linguagem é Python, na carona do que se faz no resto do mundo. O Google não me deixa mentir: ao projetar a ferramenta de aprendizado na qual ele vai botar todas as suas fichas, a linguagem escolhida é esta (estou falando do TensorFlow). O Python pode ser aprendido em poucas horas e em mais algumas horas permite a construção de engenhos robustos e que não são apenas do tipo *toy domain* ou seja, que geram resultados aproveitáveis e capazes de entrar em produção. Das 3 horas de cada encontro, esta atividade absorve partes decrescentes. Explico melhor: nos primeiros 4 encontros, esta atividade exige 100%. Depois 2 horas, e depois 1 hora em cada encontro, mas ela não tem fim. Treinar a programação é uma tarefa interminável. Esta atividade gera uma exigência semanal e ela precisa ser entregue pelo aluno e avaliada pela escola. O conjunto de programas que o aluno vai construindo ao longo das semanas vai formando o porfólio individual e este é um resultado e tanto do esforço do aluno. É importante salientar isso, já que a produção neste caso é abstrata (um programa é uma abstração, mas é uma senhora abstração: os

 ©75-20, Pedro Kantek

donos do Google não me deixam mentir: construíram uma das maiores fortunas do mundo sobre o Page Rank: um algoritmo antes e um programa depois). Uma das características da escola é concretizar o abstrato, este abstrato. Vamos *bater bumbo* sobre este tema. Um bom programa, que funcione e tenha existência independente é um fato a ser comemorado e disseminado.

```
atividade prática é constituído por dezenas de exercícios práticos que podem ser vistos em
algoritmovivo.com. É o que mais tem lá.
```

Computação evolutiva e solução de problemas que não sabemos resolver A atividade começa estudando e caracterizando problemas que não sabemos resolver. Note-se a abertura que esta abordagem permite. Qualquer outra atividade de programação começa caracterizando a solução – portanto restringindo a mesma. Aqui ao contrário contamos com a nossa ignorância e via de consequência, abrimos o campo de atuação.

As características da computação evolutiva são trabalhadas agora: em número de 4:

1. a solução como um cromossomo
2. o papel da natureza e da disputa por recursos
3. a passagem das gerações: a seleção natural (ou artificial, neste caso)
4. recombinação e mutação, como a evolução aprende

Cada uma delas é mapeada para um poblema e uma plataforma em particular. Começa-se com um problema conceitualmente simples e com ênfase na operacionalização do conceito. Prepara-se o terreno para implementar as idéias depois em um caso mais real e por consequência mais complexo. Eis alguns exemplos possivelmente reais:

1. melhor gingle de uma campanha comercial
2. receitas culinárias inesperadas e sobretudo gostosas

```
atividade prática é VIVO809: definições da IA
vivo942: algoritmos genéticos
vivo943: um problema de PO usando computação evolutiva
vivo945: programação genética
vivo947: pacote TUDEL
```

Certificação digital A digitalização da vida moderna e seus desafios: um deles é garantir que seu interlocutor que você não conhece e nunca vai conhecer seja quem diz ser. Até há poucos anos, ia-se a um cartório e fazia-se uma declaração, carimbada e cheia de selos. Isto certificava algo como verdadeiro, mas a característica fundamental era o necessário *vis-a-vis*. Como fazer o mesmo se o seu parceiro de negócios está na China ou no Paquistão ? Agora, compra-se um certificado digital.

Os livros e os especialistas têm dito que isso é seguro, que funciona, mas a pergunta é: *como é mesmo que isso tudo se encaixa e garante uma identificação 100% segura ?*. Para que esta explicação seja aceita, todas as idéias e tecnologias precisam ser conhecidas e aceites.

Se começa pelo conceito da criptografia e seu uso ao longo dos séculos principalmente na guerra (e no amor). Depois a escalada trazida ao assunto pelo surgimento da eletrônica digital e do computador.

A evolução da criptografia simétrica em direção à criptografia assimétrica e as grandes vantagens desta, ao prescindirem da necessidade de troca prévia de chaves.

Vão-se estudar os principais algoritmos assimétricos, enfocando principalmente o RSA (Rivest, Shamir e Adleman) e o seu subproduto bem importante, a assinatura digital. Lembrando que este algoritmo e seus derivados é que garantem boa parte da segurança comercial da Internet nos dias de hoje.

A partir deste ponto quais foram as construções de protocolos, arquiteturas e tecnologias que permitiram expandir a assinatura digital através de uma hierarquia certificadora.

Não se pode tratar este tema sem uma passada pelo conceito de cartório virtual, gratuito, 100% seguro representado pela tecnologia de *blockchaining*.

Antes da atividade, uma caminhada pela certificação temporal e sua tecnologia associada que é a certificação de objetos binários: o uso de funções *hash* e o que elas garantem.

Este tema assume importância porque as instituições brasileiras (como por exemplo o poder judiciário) já usam esta tecnologia, as pessoas assinam digitalmente coisas, tudo baseado em leis aprovadas, como manda o figurino, mas com bem poucas pessoas realmente sabendo como funciona e em que se baseia esta estratégia.

Aqui vai se desenvolver uma atividade prática de certificação envolvendo todos (ou a maior parte) dos participantes. O instrutor é a autoridade certificadora raiz e todos poderão ser certificados, estabelecendo-se a cadeia de hierarquia até a raiz.

Inteligência artificial Dartmouth College em 1956 e o nascimento da Inteligência Artificial: Seguramente há 2500 anos, na Grécia antiga começaram a surgir as teorias do raciocínio e da aprendizagem. Mais precisamente durante algumas décadas por volta de 420 aC, um trio andou produzindo as bases de toda a cultura ocidental. Fala-se de Sócrates, seu aluno Platão e o aluno deste, Aristóteles. Estes 3 abordaram a política, a matemática, a física, astronomia e muita, muita filosofia.

Posteriormente, há cerca de 400 anos, foi a vez dos matemáticos assumirem. Tudo começou com René Descartes que por volta de 1630 estabeleceu a teoria do dualismo, afirmando que o homem é constituído de corpo e mente. A diferença para os animais é de que estes têm apenas corpo. A resposta veio através de Leibniz, que em 1700 aproximadamente, afirma o contrário: Tudo é matéria, e toda a matéria está sujeita às leis do Universo, ainda que estas leis sejam desconhecidas. O nome desta doutrina é, a propósito, *materialismo*.

No fim das contas, segundo Leibniz, a mente é uma máquina que funciona como um dispositivo físico que raciocina manejando o conhecimento até ali depositado.

Nos primórdios da IA, a preocupação, era a construção de programas inteligentes, capazes de perceber um estado de coisas e sugerir o melhor curso de ação. À medida em que o tempo foi passando, surgiu, já nos anos 1990, uma abordagem mais moderna de IA. Segundo esta, *IA é construir agentes inteligentes*. Assim, precisa-se definir o que seja um agente inteligente: Trata-se de algo capaz de perceber e atuar. Assim, IA é o estudo e a construção de agentes racionais. A visão antiga da IA (fazer inferências corretas) é, ou pode ser, apenas uma parte da visão mais moderna.

Aristóteles já dizia "o que nos preocupa é o fim". Daí, deve-se ir aos meios que vão permitir chegar ao fim desejado. (Este curso de raciocínio recebeu o nome de *ética Nicomáquea, Aristóteles*). Este enfoque foi implementado por dois pesquisadores, Newel e Simon, num programa chamado GSP, cerca de 2300 anos depois de Aristóteles. Aqui, explora-se a diferença entre o que se tem e o que se quer.

A matemática assumiu o bastão depois da idade média. Para passar da filosofia para uma ciência formal como a matemática, necessitou-se desenvolvimentos em 3 áreas: computação, lógica e probabilidade. Alguns expoentes importantes: Al Jwarizmi, Cardano (1501-1576), Fermat, Pascal, Bernouilli, Boole, Frege, Hilbert, Gödel, Bayes, Turing, entre outros.

A psicologia, também foi importante, tendo como marco o livro *Handbook of physiological optics* publicado em 1879 por Helmholtz. Ele inaugurou a tradição existente até hoje de mesclar a IA e a ciência cognitiva. Vale referência também ao papel de Freud, ao descrever o processo de introspecção que levou ao inconsciente.

A linguística teve a sua importância a partir dos trabalhos de Skinner (1957) e a posterior formalização da linguística por Chomski. A importância desta contribuição se faz presente num capítulo importante da IA, chamado PNL (processamento em linguagem natural), uma das primeiras iniciativas da IA.

Finalmente, a computação teve papel fundamental na IA, já que para que esta exista, deve-se ter inteligência+artefato (este, o computador).

McCullogh e Pitts propuseram em 1943 uma Rede Neural Artificial. Foi uma construção intelectual impressionante, resultado de no mínimo 3 *inputs* a saber:
⋆ A fisiologia de um neurônio natural. Conhecida a partir dos trabalhos do médico espanhol Ramon y Cajal, no início do século XX.
⋆ Lógica Proposicional formulada e sistematizada por Russell
⋆ Teoria da Computação, a partir dos trabalhos de Turing

Shannon em 1950 e Turing em 1953 chegaram a escrever programas de computador para jogar xadrez.

Em 1956, organizou-se um seminário de verão, com duração de 2 meses no Dartmouth College, com a finalidade de conectar pesquisadores e de avaliar o estado da arte. Atenderam ao seminário 10 pessoas. Duas foram as consequências importantes: Na primeira, os pesquisadores se conheceram e tiveram seus trabalhos interfecundados. Na segunda, foi acordado o nome para este ramo novo de conhecimento: **Inteligência Artificial**.

O programa TL (Teórico Lógico) construído por Newell e Simon na Carnegie Tech (hoje Universidade Carnegie Mellon), deu uma demonstração de um dos teoremas do Principia Mathematica (de Bertrand Russel) que era mais curta que a original do autor. Este fato entretanto não sensibilizou os editores do *Journal of Symbolic Logic* que recusaram um artigo de Newell, Simon e TL descrevendo o fato.

No começo hove grandes progressos, entretanto já nessa época os céticos diziam que o computador jamais poderia fazer X. Os pesquisadores respondiam a isso implementando X. Essa época é conhecida como *olhe mamãe, agora sem as mãos*.

John McCarthy (que já fora o autor do nome IA) fêz 3 colaborações de peso:
⋆ a construção da linguagem LISP (LISt Processor), verdadeiro paradigma na programação de IA até os nossos dias. É a segunda linguagem de programação mais antiga ainda em uso, só perdendo para o FORTRAN.
⋆ a tecnologia de tempo compartilhado. Essa idéia acabou implementada e os responsáveis construiram a Digital

 ©75-20, Pedro Kantek

Co, durante muito tempo a segunda maior fabricante de computadores, e que implementava em toda a linha o tempo compartilhado.
⋆ A escrita do artigo *Programs with common sense*. Embora com quase 50 anos este artigo continua atual. Ele propõe que programas usem raciocínio mais algum conhecimento do mundo, e esta segunda parcela foi a novidade apresentada.

Em 1958, Marvin Minsky foi para o MIT e junto com McCarthy começou a definir o campo de estudo. A partir daqui surgiram os micromundos:

- Slagle, 1963, SAINT, que fazia cálculo de integrais
- Evans, 1968, ANALOGY, respondia testes de QI
- Bobrow, 1967, STUDENT, resolvia problemas de álgebra do colegial
- Winograd, 1972, programa que compreendia linguagem natural
- Huffman, 1971, programa que manipulava blocos sobre uma mesa
- Waltz, 1975, programa que olhava e entendia os blocos de Huffman
- Rosenblat, 1962, Perceptron, programa que aprendia (RNA)
- Weizenbaum, 1965, ELIZA, uma brincadeira que foi levada a sério.

Nos anos 70, viu-se que tinha havido otimismo de sobra. De fato, a primeira aplicação "comercial" de IA era um tradutor russo-inglês, motivado pelo receio dos EUA de perderem a guerra (fria) com a URSS, na esteira do lançamento do Sputnik em 1957. Este programa fora um fracasso, traduzindo por exemplo, "O espírito é forte, a carne é fraca" por "a vodka é boa, mas a carne está podre".

Um relatório do governo americano em 1966 afirmou que nenhuma tradução havia sido gerada e com isso contratos e subvenções foram cancelados.

Antes da teoria NP (devida a Cook, 1971) se achava que para resolver problemas reais bastava agregar mais hardware e mais memória. Em 1973, através do relatório Lighthill, o Governo Britânico retirou a ajuda generalizada na pesquisa de IA mantendo-a apenas em 2 universidades. Este relatório citou a incapacidade dos programas em lidar com a explosão combinatória. Hoje existem importantes desenvolvimentos, sobretudo no que tem sido chamado de aprendizado de máquina.

```
atividade prática é VIVO809: definições da IA
vivo810: diagramas PAGE para sistemas de IA
vivo814: o mundo dos aspiradores de pó inteligentes
vivo815: sistemas de produção
vivo817 e vivo819: sistemas especialistas
```

Redes neurais e aprendizado automático Ou Tensorflow: a rede neural do Google. Este produto só roda em máquinas 64 bits (lembrem-se, ele não economiza recursos) e estando nesse ambiente, a instalação é simples, basta fazer `pip install tensorflow`.

Depois de instalar, para verificar se deu tudo certo faça:

```
>>> import tensorflow as tf
>>> hello = tf.constant('Oi, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
```

Deve aparecer a mensagem `Oi, TensorFlow!` indicando que a instalação foi bem sucedida.

O ponto central em TF é o tensor. Trata-se de um array de valores primitivos configurados em uma matriz (Python). O ranking de um tensor é o número de dimensões da matriz que o contém. Eis alguns exemplos:

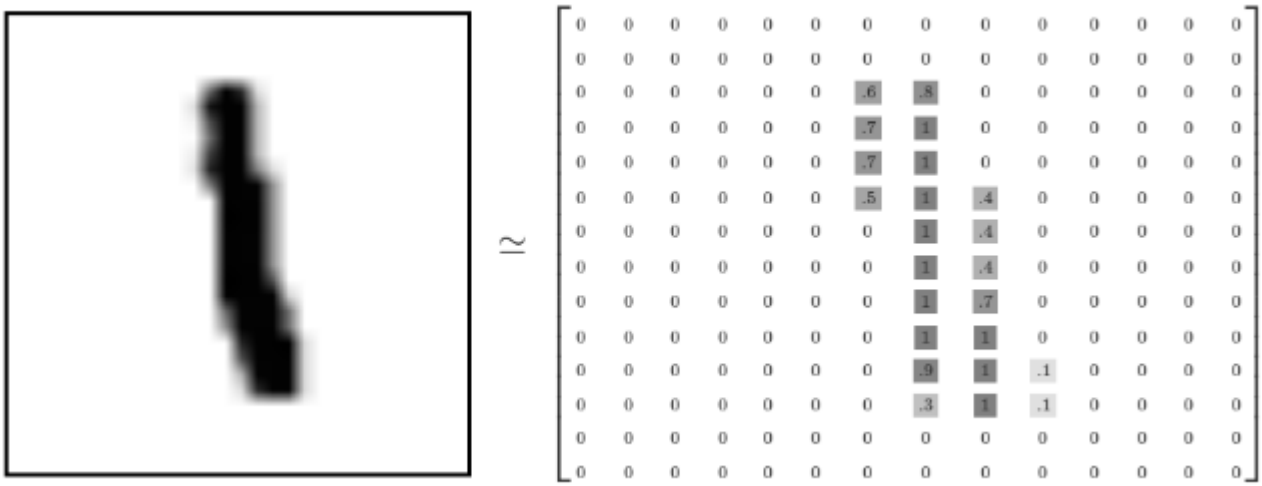
```
3 # tensor de rank 0; escalar com forma []
[1. ,2., 3.] # tensor de rank 1;vetor forma [3]
[[1., 2., 3.], [4., 5., 6.]] # tensor rank 2;
#      matriz com forma [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]]
# tensor rank 3; tensor com forma [2, 1, 3]
```

Uma rede neural é uma série de operações TensorFlow dispostas em um grafo de nodos. Cada nodo tem zero ou mais tensores de entrada e produz um tensor de saída.

Um exemplo completo está a seguir. Ele é um interpretador de dígitos escritos à mão, a partir dos dados do NIST.

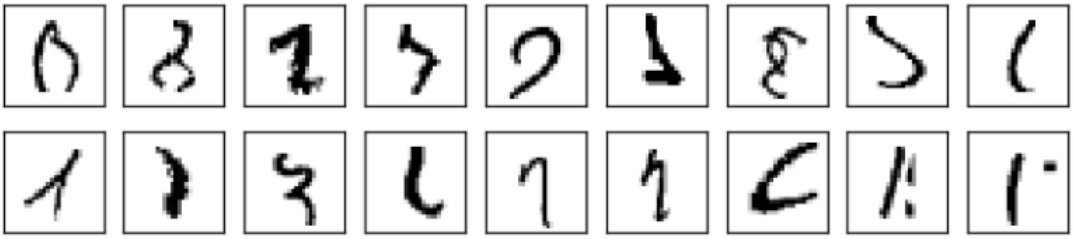
É um conjunto de 70.000 imagens de dígitos manuscritos capturadas como imagens de 28×28 pixels ($28 \times 28 = 784$) associados a um vetor [0..9] que indica qual o número certo. Os dados estão divididos em 55.000 (treinamento) 10.000 (teste) e 5.000 validação. Lembre-se que quem aprova não pode ser o que treinou.

Veja um exemplo de um número 1



Se você rodar o programa abaixo terá uma acurácia de 92% aproximadamente. Melhorando os operadores da rede neural chega-se facilmente a 97% e os melhores modelos chegam a 99.7 %.

Eis algumas imagens difíceis do conjunto



Uma explicação completa deste exemplo está em

```
\verb+https://www.tensorflow.org/get_started/mnist/beginners+
O módulo abaixo é mnist_softmax.py...

atividade prática é vivo841
```

Matemática (aprender a pensar) Esta atividade deve ser a queridinha dos alunos: de fato, nossos alunos devem amar a matemática, ou pelo menos devem poder ser levados a amá-la. Qualquer rechaço aqui deve poder ser contornado ou senão implica desligamento: um programa de computador é um teorema e depurar e certificar um programa é igual a eventualmente demonstrar um teorema.

A palestra começa definindo um fusca: nele entrarão os 5 maiores matemáticos deste planeta: Gauss, Newton, Euler, Fermat e Turing.

Esta apresentação é um passeio pela matemática moderna. Começa em meados do século XVII com Newton e sua história de vida. A infância difícil e a entrada na universidade que foi interrompida por uma peste que determinou a volta a casa de todos os alunos. Com alguns meses em casa, sem nada para fazer exceto pensar, foi o que ele fêz. Na época o movimento dos planetas estava sujeito a inúmeras teorias e explicações com um (ou os dois) pés no ocultismo e no obscurantismo. A sacada de Newton foi intuir que a força que atraía uma maça ao solo (na cabeça ?) era a mesma que mantinha Terra e Lua unidos. Aliás, mantinha todo o universo unido. Bingo. Estava posta a lei universal da gravitação $F = G.\frac{m_1 m_2}{d^2}$. A seguir, começou a estudar as variações. Mais um pouco e estavam dados os fundamentos do cálculo diferencial e integral. Historicamente a integração veio antes no bojo das tentativas de medir área e volumes. Newton estipulou que integração e diferenciação são funções inversas. Há aqui uma disputa com Leibnitz que mais ou menos na mesma época publicou suas idéias sobre o cálculo. Ambos falam da mesma coisa, mas a notação empregada por Leibnitz é melhor do que a usada por Newton. Aliás é bastante próxima da que usamos hoje. Também descobriu a decomposição da luz branca em uma mistura de diversas luzes de cores (frequências) distintas. É dele a famosa experiência da decomposição da luz em seus componentes ao atravessar um prisma de vidro, devido à refração. Já com a reputação solidamente estabelecida, Newton foi convidado pela coroa a assumir a Casa da Moeda Real. Aqui não deu folga a falsificadores, pendurando alguns na forca.

Depois vem Gauss, um alemão do século XVIII. Estudou probabilidade (a distribuição de Gauss), teoria das equações (teorema fundamental da Análise), lançou as bases da matemática complexa. Números aqui são representados no

 ©75-20, Pedro Kantek

sistema de eixos de Argand-Gauss. Trabalhou também com a aritmética modular, base de boa parte da matemática atual. Ele dizia que uma construção matemática é como uma catedral medieval. Antes de inaugurá-la é preciso tirar os andaimes usados na construção. Ao eliminar todas as construções intermediárias, seu trabalho ficava ainda mais brilhante. Seu dístico: *pauca sed madura!*

Depois vamos para Euler, o mais prolífico de todos. Teve uma vida longa com os últimos 20 anos acompanhado de uma cegueira primeiro parcial e depois total, fruto de infecção ocular. Sua declaração a respeito: *"tanto melhor, agora posso me dedicar mais a pensar"*. Foi o primeiro programador profissional da história, ganhou 100 libras do almirantado britânico por ter proposto uma maneira fácil de determinar a posição de um navio no mar. É dele a teoria dos grafos, a equação $e^{i\pi} + 1 = 0$ que junta os 5 símbolos mais importantes da matemática. Trabalhou em otimização e em dinâmica de fluidos.

Agora, Pierre de Fermat. Um advogado de profissão, trabalhou a vida inteira em Toulouse. Cedo virou magistrado e decidiu que não estabeleceria relações de amizade na cidade: poderia vir a ter que julgar essas pessoas. Tanto melhor para a matemática que recebeu seus cuidados. Não publicou nada de matemática (afinal, era apenas um hobbie) e sabemos de suas idéias apenas pelas cartas que escreveu e que sobreviveram. Trabalhou em probabiblidade, lançou as bases da geometria analítica, deixou grande contribuição na área de matemática inteira. Seu teorema de números primos é a base da moderna segurança na internet. Sem falar no último teorema de Fermat, provado apenas em 1993/94 por Wiles.

Finalmente Turing, o mais novo de todos. 4 trabalhos dele podem ser listados aqui, em ordem cronológica. Primeiro ele ajuda e muito os ingleses a sobreviver ao ano de 40/41 e muitos anos depois ganhar a segunda grande guerra. Depois ele constrói um computador. Mais tarde, estabelece as bases filosóficas da inteligência artificial (é desta idéia o Teste de Turing) e finalmente começa a estudar as bases matemáticas da herança biológica: como uma máquina pode criar uma cópia de si.

```
a atividade prática neste tema é uma tradução criativa de (partes) da
obra magnífica "Disquisitiones Arithmeticae" do matemático Carl Gauss.
```

Estado da arte em edição de textos: LATEX Em 1968 o professor D. Knuth começou a escrever a coleção seminal da programação de computadores, "The Art of Computer Programming". Embora tais livros tenham mais de 50 anos, continuam atualíssimos. De que outros livros de informática com essa idade dá para dizer a mesma coisa ?

O Knuth ficava incomodado com o visual pobre que os livros dele tinham ao ser impressos, e começou a buscar uma ferramenta que lhe permitisse usar o próprio computador para compor os livros e os textos que escrevia. Como tal ferramenta não havia, ele botou mãos à obra em 1977. Assim, em 1982 nasceu o TEX. Com alguns melhoramentos introduzidos alguns anos depois, ainda é o mesmo programa. A propósito, vale lembrar que os softwares são como os bons vinhos: os melhores são os que envelhecem bem. A versão dele começou com 3, e ela tende a se aproximar de π. Atualmente está na versão 3.14159.

Em meados dos anos 80, Leslie Lamport desenvolveu uma série de macros que automatizavam o uso do TEX. Nascia o LATEX, que é usado até hoje e que está sendo usado para compor este material.

O ambiente LATEX é composto por:

linguagem LATEX Um conjunto de comandos que são introduzidos em um texto ASCII comum.

editor ASCII Um programa capaz de manusear arquivos. Pode ser o EDIT (do DOS), o NOTEPAD (do windows), o WORD e similares. Para o WORD os dados devem ser gravados "puros" sem aquelas marcações todas que o word coloca. A terminação deste arquivo deve ser "tex".

compilador LATEX Um programa que leia o texto que contém os comandos LATEX e gere um arquivo com o material já composto. Este arquivo tem o mesmo nome do arquivo original e a terminação dvi (que significa "independente de dispositivo").

Visualizador Um programa que seja capaz de ler (e imprimir) o arquivo "dvi" e gerar uma visualização na tela ou uma impressão.

extras Programas para gerar versões PDF, PS entre outros.

Um documento LATEX é formado por duas partes, o preâmbulo e o corpo. O primeiro vai desde o início do arquivo até o comando `\begin{document}`. O corpo começa depois deste comando e vai até o comando `\end{document}`. Tudo que vier depois disso é desprezado pelo LATEX.

As informações que são colocadas no preâmbulo se aplicam diretamente ao texto e valem para todo o arquivo. As informações do corpo são o texto que vai ser composto misturado a comandos de editoração que são lá colocados.

Este ambiente não é WYSIWYG (*what you see is what you get*). Alguns acham isso ruim, mas eu acho excelente: Quando se escreve um texto maior que um bilhete, há duas tarefas bem separadas: escrever o texto e compor o texto. Misturá-las (como o WYSIWYG faz) não ajuda e é fonte de erros e dificuldades.

```
a atividade prática é vivor40: LATEX
```

Probabilidade e Teorema de Bayes Thomas Bayes foi um pastor protestante provavelmente nascido em 1701. Integrante da igreja inconformista não pode entrar em Oxford ou Cambridge precisando ir estudar na universidade de Edimburgo. Escreveu *An Essay toward Solving a Problem in the Doctrine of Chances (Tentativa de elucidar um problema na doutrina do acaso)*. O livro só foi publicado após a sua morte por seu amigo Richard Price. Na introdução, Price sugere um primeiro homem: Adão, talvez recém criado ou recém saído da caverna de Platão, conforme maravilhosamente descrito no livro VII d'A República. Ele vê o sol nascer pela primeira vez e fica maravilhado. Nesse dia ele não sabe se é um fenômeno típico ou ao contrário, insólito. Mas, a cada dia que sobrevive e vê o sol nascer aumenta sua confiança de que se trata de uma característica periódica da natureza. Aos poucos, baseado nessa dedução estatística a probabilidade que ele aplica à previsão de que o sol vai nascer no dia seguinte se aproxima de 100%, embora nunca chegue a esse ponto. Já a expressão matemática mais comum do que hoje é reconhecido como o Teorema de Bayes foi desenvolvida pelo ateu, matemático e astrônomo francês Pierre-Simon Laplace. O teorema associa quatro variáveis sendo 3 conhecidas ou supostas e a quarta desconhecida. Ele se concentra na probabilidade condicionada. Ou seja, na probabilidade de uma hipótese ser verdadeira SE tiver acontecido determinado coisa.

Vamos usar o Teorema de Bayes no atentado às Torres Gêmeas em NY no dia 11 de setembro de 2001. Antes desse dia, a probabilidade de um atentado terrorista era muito baixa, próxima a 0%. Vamos calcular o que aconteceu depois do primeiro avião e sobretudo depois do segundo avião. O P(A) é a chance de um atentado terrorista no qual um avião é atirado contra um prédio. Peritos americanos em segurança avaliavam previamente (antes do 11/set) esta probabilidade em 0.00005 ou 0.005%. Agora P(B) é um choque de avião em Manhattan. P(B|A) é 100%. P(B|~A) é um choque de avião por acidente. Essa é fácil: nos 25.000 últimos dias ocorreram 2 eventos: em 1945 um avião bateu no Empire State Building e em 1946 um avião caiu no número 40 de Wall Street. Então P(A) = 2 em 25.000 ou x = 0.00008 ou x = 0.008%. Calculando tudo certinho, acha-se P(A|B) = 0.38. Agora a grande vantagem do Teorema de Bayes que é usar os mesmos dados numa segunda vez. Nossa segunda vez é usar o teorema após o primeiro avião ter sido jogado contra o prédio. Será que o segundo avião poderia ter sido um acidente ? Vamos lá. Agora P(A) = 0.38 e os demais dados permanecem inalterados. Acha-se P(A|B) = 0.9998720081 ou seja, a probabilidade de um atentado terrorista dado que o segundo avião colidiu é de 99,99%, nada mau, para uma teoria que nasceu no século XVIII, não é ?

```
atividade prática é vivo911
vivo912: prática em probabilidade com uso de simulação
vivo913: Redes Bayesianas de Crença
```

Espaços de busca imensos: o xadrez, o rubik, por exemplos As velocidades de um computador e de um cérebro humano são muito diferentes: se nós tivessemos a velocidade dele, a entrada de dados durante a digitação de um texto, equivaleria a receber um ditado à razão de uma letra por semana.

Essa distinção nos permite usar o computador para resolver problemas de uma maneira bastante simples: para achar a melhor alternativa de solução, podemos testar todas as possibilidades e ficar com a melhor. Esta estratégia, só usando um computador, ninguém a usaria para resolver o problema à mão.

Há problemas que impedem esta estratégia: por mais rápido que um computador seja há ocasiões em que testar todas as possibilidades demoraria bilhões de anos. Para estes problemas há outras maneiras (como o algoritmo A*), mas quando testar cada possibilidade demora 1 milésimo de segundo e há 2000 possibilidades, o computador demora 2 segundos: perfeitamente aceitável.

Denomina-se esta estratégia de *busca exaustiva* ou também de *força bruta*. Algumas questões chave aparecem aqui. A primeira é o conceito de recursividade. *(Um programa de computador chama a si mesmo para resolver uma nova instância do problema)*. A recursividade não é obrigatória, mas seu uso torna as coisas muito mais fáceis. A segunda idéia é chamada backtracking. Imagine você percorrendo um laberinto: a cada bifurcação, fixa-se um barbante e ele vai sendo desenrolado. Ao chegar a um beco sem saída, retorna-se enrolando o barbante e ao chegar onde ele foi fixado, marca-se a entrada anterior como já vista e vai-se para a próxima.

Vamos estudar estas estratégias em 2 problemas simples: o primeiro é o chamado passeio do cavalo. Relembrando o jogo de xadrez e o movimento da peça cavalo. Colocando um cavalo em uma posição qualquer do tabuleiro qual seriam os movimentos para que ele pudesse passar em todas as casas do tabuleiro sem repetir nenhuma casa e sem deixar nenhuma sem ser visitada ?

O outro problema é o do quebra-cabeça conhecido como SUDOKU. Lembrando, sua origem é recente no tempo: menos de 30 anos e ele parece ter surgido no Japão pela dificuldade local óbvia de propor palavras cruzadas. Fica muito mais fácil lá usar números do que letras. Há uma grade de 9×9 casas e nela há 9 sub-grades de 3×3. Há portanto, 9 linhas, 9 colunas e 9 sub-grades estas de 3×3 casas. Em cada linha, coluna e sub-grade, devem-se colocar os dígitos de 1 a 9 sem nenhuma repetição. O problema já vem com algumas casas preenchidas e é a partir destes valores que o problema é resolvido.

Para um humano fazer isto, é preciso *quebrar a cabeça*. Para um computador usando busca exaustiva é mais simples: basta testar todas as possibilidades até chegar na solução.

 ©75-20, Pedro Kantek

Um outro exemplo é o conhecido como casamento perfeito. Finalmente, um contra exemplo é dado pelo problema do Caixeiro Viajante, conhecido na literatura como TSP.

```
atividade prática é vivo253: passeio do cavalo
vivo256: casamento estável
vivo257: sudoku resolvido por computador
vivo259: problema do caixeiro viajante
```

Banco de dados e Modelo de Date e Álgebra Relacional Falamos de sistemas nos anos 70: cada aplicação tinha seus próprios arquivos. Máquinas tinham 64Kb de memória e discos de 5MB eram raros porque muito caros. Os dados residiam em fitas. Os primeiros bancos de dados conviviam com estas restrições e eram de uma complexidade interna bem grande. Até que um analista da IBM (C. Date) bem à frente de seu tempo, lançou as bases da álgebra relacional e pontuou que arquivos deveriam ser vistos como tabelas e ato contínuo definiu operações possíveis sobre essas tabelas. Que as máquinas dessa época fossem incapazes de realizar tais operações era mero detalhe. Nascia o modelo relacional que até hoje modela nossos sistemas de informação. Aqui um arquivo é composto por colunas (atributos) e linhas (registros). Sobre essas relações, Date definiu 3 operações: restrição (extração de linhas particulares da relação com base em alguma condição), a projeção (extração de algumas colunas específicas) e junção (a união de 2 relações com base em um valor de atributo comum). As 3 operações recebem uma ou mais relações e devolvem outra relação, o que caracteriza a álgebra relacional envolvida. Na álgebra esta propriedade é conhecida como *fechamento* e a consequência imediata é que a saída de uma operação pode ser a entrada de outra. Para que as relações possam descrever adequadamente o universo de um sistema de informações, elas precisam passar por um processo chamado normalização que é um conjunto de projeções com vistas a diminuir a redundância e garantir a integridade dos dados. Antes, precisa-se definir a chave de uma relação, como a informação que identifica cada linha. A literatura registra inúmeras formas de normalização, mas vamos listar as principais:

primeira forma normal, ou ÑN → 1FN separar as colunas multivaloradas da relação

segunda, ou 1FN → 2FN separar as dependências funcionais da chave principal

terceira, ou 2FN → 3FN descoberta de dependências funcionais de atributos não chave

O objetivo da normalização é facilitar a garantia de integridade referencial. Esta, é a propriedade de um banco de dados de garantir a qualquer momento a inexistência de qualquer informação que esteja em desacordo com as regras de negócio que o banco de dados implementa.

```
atividade prática é vivod60: introdução ao modelo relacional
vivod62: normalização de relações
vivod63: integridade referencial e
vivo64: restrições de integridade
```

Idiomas do planeta terra e os computadores A informática convive – desde seu início – com uma dificuldade e tanto: como padronizar a utilização de códigos binários. O objetivo é nobre: permitir a troca de objetos binários entre indivíduos, empresas, países e culturas diferentes e a sua correta interpretação por parte de todos os parceiros em uma comunicação qualquer. A necessidade é nova, a informática só apareceu na face da terra na década de 50 do século passado. No início, cada fabricante usava seus códigos. A IBM, por exemplo, construiu toda a sua família de equipamentos usando o que ela chamou de código EBCDIC. Isto evoluiu para o código ASCII, oficializado na comunidade americana pelo órgão normatizador associado a grandes fabricantes, usuários e universidades (início de 1960, comitê ANSI-X3.2) que acabou sendo adotado como padrão. Ele privilegiou o idioma inglês, e padronizou seus caracteres nas primeiras 128 posições do código ASCII. Aos demais idiomas do planeta ficou disponível a segunda parte do código ASCII (mais 128 posições). É aqui que o código BRASCII (NBR-9614:1986 e NBR-9611:1991 da ABNT) colocou os caracteres acentuados do português. Assim como o Brasil, muitos outros países fizeram o mesmo. Isto apresentou pelo menos 2 problemas:

- Um arquivo composto no Brasil, quando impresso numa impressora fabricada na França, por exemplo, aparecia como uma série de caracteres incorretos.
- Idiomas com muitos caracteres (katakana, hiragana, hangul, tagalog,...) simplesmente não tinham espaço para ser representados no código ASCII. Não havia como.

Em meados da década de 80, a academia e a indústria começaram a discutir propostas de resolver os 2 problemas acima. Diversos padrões foram sugeridos e eles começaram a ser usados, na base do "cada um por si". Uma primeira proposta sugeria usar 16 bits (2 bytes) o que permitiria $2^{16} = 65536$ caracteres. Parecia muito, mas vale citar Peter Norton, que disse há mais de 40 anos: *na informática, não importa quanto você tem. Não é o suficiente.*

Em 1991, criou-se o Consórcio Unicode e em outubro de 1991 o primeiro volume do padrão foi publicado. Em 1996, introduziu-se um mecanismo de codificação e com ele, a barreira dos 2 bytes foi ultrapassada. Agora, o

espaço de endereçamento do Unicode ultrapassou o milhão de caracteres, o que permitiu incluir idiomas históricos, como hieróglifos egípcios, ou mesmo idiomas muito pouco usados, como Linear B por exemplo, além de algumas brincadeiras como o idioma Klingom. Tipicamente estamos diante de uma piada nerd.

O padrão atual consagra 1.114.112 códigos de ponto (*codepoints*), correspondendo ao intervalo hexadecimal de 00.00.00 a 10.FF.FF. Note que nem todo codepoint indica um caracter, já que alguns deles são reservados a outras finalidades.

Há alguns codepoints privados, que não têm significação no Unicode. Eles ficam para uma negociação entre remetente e destinatário e podem ser usados para qualquer coisa.

O Consórcio Unicode trabalha junto com a ISO (International Organization for Standardization) já que o padrão ISO/IEC 10646 é o próprio código Unicode. Um conceito importante é o de script. Trata-se de um conjunto de letras, sinais e regras de escrita que agregam partes do Unicode e que são usados em conjuntos de *writing systems*. Na versão 7 do Unicode, são 123 os scripts, e há uma lista de candidatos a serem incluídos.

Diversos mecanismos foram sugeridos para implementar o código Unicode, com as preocupações sempre presentes de eficiência e de compatibilidade reversa, isto é: um novo esquema de codificação deveria preservar – tanto quanto possível – o legado infindável de dados já coletados e guardados. Criou-se o conceito de UTF ou *Unicode Transformation Formats*. Foram propostos os UTF-1 (nunca chegou a ser muito usado, pois tinha problemas de performance), UTF-16 (usava 1 ou 2 palavras de 16 bits) , UTF-32 (cada caracter usava exatos 32 bits), mas todas elas acabaram migrando para a UTF-8 que se firmou como padrão de fato, principalmente porque muito cedo foi adotado como padrão pelo consórcio IMC (Internet Mail Consortium) e pelo W3C (a Internet). O Google reporta que desde 2008, a maioria dos conteúdos HTML trocados na web usa a codificação UTF-8. O esquema todo foi escrito durante um jantar em 2 de setembro de 1992 por Ken Thompson e Rob Pike. [2] Suas grandes qualidades são:
⋆ Compatibilidade reversa, já que caracteres da primeira parte do ASCII não sofrem modificação (1^o bit=0). Desta maneira qualquer arquivo ASCII que não use a segunda parte já é um arquivo UTF-8.
⋆ Clara distinção entre caracteres de byte único (1^o bit=0) e caracteres multi-bytes. Estes contém um primeiro byte heading e um ou mais bytes de continuação. O heading tem 2 ou mais bits 1 seguidos por um zero enquanto os bytes de continuação sempre começam por 10.
⋆ Auto-sincronização. Bytes únicos, header e continuação não compartilham valores, indicando que o início do caracteres está a no máximo 3 bytes para trás.
⋆ Indicação do comprimento: O número de uns na alta ordem do byte heading indica a quantidade de bytes do caractere, sem ser necessário examinar os seus bytes.

Dito assim, parece bem complexo mas a lindeza do esquema está aqui: uma vez visto estudado e entendido o esquema se revela bem simples, embora trabalhoso. Só que o trabalho fica por conta das máquinas, a nós resta apenas adimirar a beleza da coisa.

`atividade prática é vivo480`

Processamento Fonético Enquanto os códigos são números (com seus correspondentes dígitos verificadores) é fácil aos sistemas computacionais localizar referências e conteúdos armazenados. Entretanto, nem sempre é possível construir os sistemas baseados em códigos numéricos. O ser humano ao nascer recebe um nome e não um número. Inúmeras referências no mundo são a nomes. Este fato complica as coisas: nomes dependem de idiomas, ou seja a cada pedacinho do mundo são escritos e falados de maneira diferente. Pior ainda, nomes escritos diferentes são ditos de mesma maneira e também o contrário: nomes ditos diferentemente são escritos iguais. Outra dificuldade é que a transmissão de nomes é pela fala, e o uso do sistema falante, e do consequente sistema ouvinte introduz nova fonte de ruídos. Na tentativa de amenizar, nunca resolver, este problema, surgiram há muitos anos os sistemas de processamento fonético. Tais sistemas, derivados da fonologia e da antropologia tentam registrar primeiro as falas a partir de seus fonemas e depois mediante algumas substituições e/ou processamentos locais obter chaves alfanuméricas que possam ser usadas na busca de conteúdos dentro de computadores.

O algoritmo SOUNDEX foi projetado em 1918 por dois pesquisadores da área de fonética. Ele transforma um nome em uma de seis classificações fonéticas: bilabial, labio- dental, dental, alveolar, velar e glotal. Esta classificação está baseada na configuração da boca, lábios e língua ao formar os sons. Naturalmente, um problema deste algoritmo é sua configuração ao idioma (aos fonemas) da língua inglesa, mas como a máquina produtora de sons é a mesma para qualquer pessoa em qualquer lugar, é possível aproveitar o algoritmo. Para cada palavra do texto de entrada, o SOUNDEX gera um código de 4 caracteres (1 letra + 3 dígitos) e a esperança é que palavras foneticamente iguais ou parecidas tenham o mesmo soundex.

As estapas do algoritmo: Primeiro, transforme todas as letras em maiúsculo. Como os fonemas são os mesmos (tanto para minúsculo como para maiúsculo) essa conversão pode ser feita sem perda de substância.

Depois, retire todos os caracteres estranhos, deixando apenas letras. Uma especial atenção deve ser dada aos números, que em geral são retirados também, a menos que eles sozinhos constituam uma palavra.

[2] Não consigo deixar de pensar que a humanidade deveria recuperar a receita do que esses dois caras beberam nessa noite.

Em seguida, individualize cada uma das palavras do nome. Para cada palavra: Retire e guarde a primeira letra da palavra. Ela é o primeiro caractere do código SOUNDEX.

Transforme as ocorrências de

```
A, E, I, O, U, H, W, Y  →  0
B, F, P, V  →  1
C, G, J, K, Q, S, X, Z  →  2
D, T  →  3
L  →  4
M, N  →  5
R  →  6
```

Deve-se remover agora pares de dígitos iguais, um ao lado do outro. Depois, deve-se remover todos os zeros do código Finalmente, deve-se devolver 4 caracteres apenas (se sobrarem, deve-se truncar. Se faltarem, completar com zeros). Este é o código SOUNDEX.

Eis um exemplo deste algoritmo aplicado ao belo poema de Manoel Bandeira, Vou me embora pra Pasárgada

```
VOU ME EMBORA PRA PASARGADA →    V000 M000 E516 P600 P262
LA SOU AMIGO DO REI              L000 S000 A520 D000 R000
LA TENHO A MULHER QUE EU QUERO   L000 T500 A000 M460 Q000 E000 Q600
NA CAMA QUE ESCOLHEREI           N000 C500 Q000 E246
```

```
atividade prática é vivo024: dígitos verificadores
vivo790: processamento fonético
```

Redes de Petri As Redes de Petri são um tipo de ferramenta para estudo do comportamento de sistemas. Embora nada exista nelas que implique sua utilização em sistemas de informação, é nessa área que elas tem sido largamente usadas. Fruto do trabalho original do Dr. C. A. Petri, que a partir de sua tese de doutorado ("*Kommunikation mit Automaten*", Bonn, 1962) desenvolveu o início da teoria, passaram a encontrar aceitação em outros centros de pesquisa e universidade e acabaram alcançando um patamar de conhecimento e disseminação significativo. E, isso ocorreu, graças a potência da ferramenta, sua generalidade e facilidade de entendimento e representação. As vantagens das redes de Petri, podem ser assim discriminadas:
⋆ Permitem o estudo do comportamento dinâmico dos sistemas
⋆ Tratam os conceitos de paralelismo e de sincronismo
⋆ Oferecem ferramentas matemáticas de análise, consequentemente permitem simular o desempenho dos sistemas através de programas de computador
⋆ Finalmente, dão condições de tratar a complexidade crescente dos sistemas de informação.

Dependendo de como são interpretados os fenômenos estudados dentro de uma rede, podemos ter os seguintes tipos de redes

- Redes tipo Condição/Evento, onde as condições são os estados possíveis para o sistema, e os eventos são os estágios pelos quais o sistema passa durante a sua vida. Uma condição pode ter 2 valores possíveis, que são "verdadeiro"e "falso".
- Redes tipo Lugar/Transição, onde o conceito de condição (V ou F) cede espaço ao conceito de lugar. Aqui, os valores possíveis para o lugar é o conjunto dos números inteiros positivos. É uma generalização do caso anterior, sendo que o valor 0 deve ser entendido como falso e os demais valores como V.
- Redes tipo Predicado/Transição, onde o lugar passa a ser um predicado e as transições passam a ter fórmulas de habilitação. Novamente é uma generalização do caso anterior, na qual os números inteiros cedem lugar a t-uplas de números.

Para a utilização das Redes de Petri do tipo mais simples (C/E), precisamos conhecer quais os elementos que serão modelados, isto é, como traduzir os sistemas em estudo em entidades próprias das redes. Uma Rede de Petri do tipo Evento/Condição é constituída de 4 elementos, a saber:
⋆ Um conjunto de condições, que são estados possíveis do sistema em estudo.
⋆ Um conjunto de eventos, que ocorrem dentro do sistema (e que podem modificar os estados do sistema).
⋆ Uma função de entrada que relaciona condições e eventos, mapeando para cada condição, um conjunto de eventos.
⋆ Uma função de saída que também relaciona os eventos e as condições e que para cada evento mapeia um conjunto de condições.

Matematicamente falando, portanto, uma Rede de Petri R é uma quádrupla $R = (C, V, E, S)$, onde C é o conjunto de condições, V o de eventos, E a função de entrada e S a função de saída.

Diz-se que uma determinada condição C_k é uma condição de entrada para um dado evento V_j, se $C_k \in E(V_j)$. Identicamente, diz-se que uma condição C_m é um condição de saída para um dado evento V_n, se $C_m \in S(V_n)$.

Trocando isso em miúdos, uma Rede de Petri será um conjunto de condições e eventos, sendo que a cada evento estarão associados dois conjuntos de condições. O de entrada e o de saída.

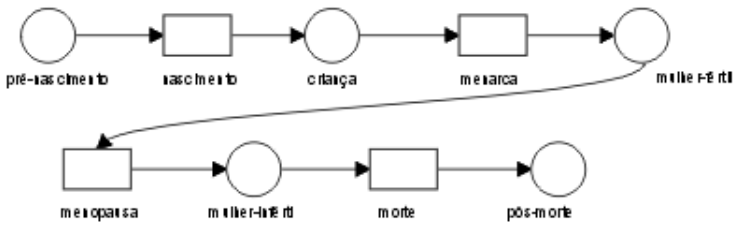
Vamos representar através de uma rede de petri tipo C/E, o modelo de vida de uma mulher. Chamemos a esse modelo de VM. Nesse Caso, RV = (CM, VM, EM, SM), onde

CM = {pré-nascimento, criança, mulher-fértil, mulher-infértil, pós-morte}
VM = {nascimento, menarca[3], menopausa, morte}
EM = {{E(nascimento) = pré-nascimento}, {E(menarca) = criança}, {E(menopausa) = mulher-fértil}, {E(morte) = mulher-infertil}}
SM = {{S(nascimento) = criança}, {S(menarca) = mulher-fértil}, {S(menopausa) = mulher-infértil}, {S(morte) = pós-morte}}.

Obviamente, como em qualquer modelagem, perdem-se aspectos essenciais do problema, que não estão sendo objeto de estudo agora (como por exemplo a utilização de uma mesma condição - criança -, para todo o tempo decorrido desde o nascimento até a menarca), enquanto se introduzem artificialismos para o funcionamento do modelo, como por exemplo, pré-nascimento e pós-morte. Mas essa é a sina da modelagem. É a sua grande qualidade, já que permite tratar coisas complexas, vendo uma coisa de cada vez, e ao mesmo tempo seu grande defeito, uma vez que simplificações significativas precisam ocorrer, para que a modelagem seja possível e tratável.

O grande charme das redes de petri, está em que, além de um tratamento matemático formal e rigoroso, admitem também uma representação gráfica, através de um grafo, que é muito ergonômico de ser usada por seres humanos. Para a criação desse grafo, usaremos as seguintes convenções:

As condições C, serão representados por círculos; Os eventos V, por meio de retângulos; Os mapeamentos da função de entrada serão representados por meio de arcos orientados saindo das condições e entrando nos eventos e Os mapeamentos da função de saída, por meio de arcos orientados saindo dos eventos e entrando nas condições.



atividade prática é vivo763b: redes de petri
vivo763c: um semi-somador em funcionamento
vivo763d: modelo de uma ATM funcionando

Simulação Simular é construir uma miniatura do sistema que ser quer estudar (e onde está inserido o problema que se quer resolver) e submeter essa miniatura a fenômenos, eventos ou pressões supostamente similares aos que atingiriam o sistema maior para analisar o comportamento do modelo e consequentemente inferir como se comportará o fenômeno maior.

Schriber, um dos construtores da linguagem de simulação GPSS, em 1974 definiu a simulação como *a modelagem de um processo ou sistema de tal forma que o modelo imite as respostas do sistema real numa sucessão de eventos que ocorrem ao longo do tempo.*

Esse modelo pode ser físico ou lógico. Ele é físico (ou analógico) quando é uma miniatura controlada do sistema maior. São exemplos desta abordagem as miniaturas de barragens (há uma imensa no Laboratório do Hidrologia do Centro Politécnico da UFPR em Curitiba), os túneis de vento, ou as provas de veículos lunares feitos na Antártida ou no Deserto de Atacama no Chile.

Os modelos lógicos usam a imensa capacidade de trabalho dos computadores digitais. Aqui o sistema é modelado como *um programa de computador cujas variáveis apresentam o mesmo comportamento dinâmico e estocástico do sistema real que representa* (Robert Shannon, 1975).

A resposta à pergunta *porque simular ?* pode ser uma das seguintes opções, ou uma combinação linear delas.
⋆ O sistema real ainda não existe. Entretanto partes precisam ser planejadas, encomendadas, produzidas e entregues em processos físicos muito demorados. Não se pode esperar para ir tomando as decisões etapa a etapa.
⋆ Conduzir experimentos no sistema real é custoso ou inadequado. Experimentar um tipo novo de asa em um avião pode conduzí-lo ao solo em rota de colisão. Experimentos de disseminação de uma toxina letal em uma população não podem ser conduzidos em populações reais
⋆ Inúmeras variáveis podem ser controladas e variadas durante a simulação, gerando riqueza de detalhes e inúmeras

[3] a primeira menstruação

 ©75-20, Pedro Kantek

opções para o processo decisório. Comparativos de custo benefício podem ser obtidos com facilidade.
⋆ A simulação pode se modificar ao longo do tempo (tal como os sistemas que estão sendo modelados) com realismo e facilidade, gerando resultados ricos

Embora a simulação se aplique virtualmente em qualquer sistema que possa ser modelado em computador, a simulação tem sido usada em: Construção one-way, barragens de hidroelétricas, protótipos de aviões, satélites, pontes, estruturas treliçadas, obras civis, sistemas de manufaturas, sistemas de manufatura flexível (job shop), movimentação de peças, estoques, sistemas just in time, áreas de armazenagem e de espera, tempos, sistemas computacionais, redes de computadores: desempenho, tempos máximos e médios, servidores (de processamento, de BDs, de memória, de comunicação), desempenho de sistemas operacionais e de software de uso geral, modelos econométricos: inflação, desempenho do PIB, previdência, desempenho de operadores de crédito, quaisquer interfaces com o público (bancos, SM, agências de governo), restaurantes, aeroportos, estacionamentos, call centers, guerra (jogos de guerra), operações logísticas, etc.

atividade prática é
vivo775: uso do ARENA
vivo782: porta da esperança

Teoria das Filas Desenvolvida no início do século XX pelo engenheiro nórdico Agner Krarup Erlang ela se aplica a sistemas onde há escassez de recursos e consequentemente há formação de filas. De maneira singela, um sistema de filas pode ser definido como *clientes chegando, sendo atendidos ou entrando em fila, esperando, sendo atendidos, e saindo do sistema*. O termo cliente é usado independentemente de representar um ser humano: poderiam ser aviões chegando, jobs em uma CPU, chamadas telefônicas em uma central, etc.

O objetivo da teoria é prever o comportamento do sistema a partir de alguns dados de entrada previamente conhecidos.

Para caracterizar processos de filas, seis coisas precisam ser determinadas:

Padrão de chegadas dos clientes Como os clientes vão chegar no sistema. Usualmente, há uma coleta de dados em sistemas similares (e/ou aplicações de estimativas "à esmo"). Para efeito da simulação, a seguir, é necessária a busca de uma distribuição teórica capaz de reproduzir os dados aleatórios de chegada a fim de garantir aderência à realidade ao modelo.

Padrão de atendimento dos servidores Valem as mesmas considerações com relação ao padrão de chegada. Tem-se uma média de atendimento, e o processo não guarda memória.

Disciplina das filas A mais simples disciplina é a FIFO (*first in, first out*). Como disciplinas adicionais, pode-se ter:

fila com prioridade Cada cliente na fila tem associado a si uma determinada prioridade. Quando vaga um servidor, a fila é questionada e o cliente que tiver maior prioridade é atendido. Se houver empate entre mais de um cliente, usa-se – por exemplo – a disciplina FIFO.

fila com preempção Até aqui, quando um cliente começa a ser atendido, ele prossegue até o final. Nos sistemas com preempção (interrupção), quando surge um cliente altamente prioritário, ele pode interromper o serviço de um cliente menos prioritário. Quando o atendimento ao interruptor acabar, o interrompido volta a ser atendido.

mais de uma fila Ao invés de organizar uma fila única, podem-se criar várias delas, segundo os mais diversos critérios. O atendimento também pode ser estabelecido em função de quaisquer critérios.

Capacidade do sistema Em alguns processos, pode haver uma limitação no tamanho máximo da fila. Quando isto ocorre, e este número máximo é atendido, nenhum cliente novo pode entrar no sistema, enquanto não se abrir espaço.

Número de canais de serviço Indica o número de servidores em paralelo atendendo a clientes.

Número de estágios no serviço Indica a quantidade de etapas sequenciais que o cliente deve sofrer dentro do sistema.

A notação empregada em sistemas de filas é devida a Kendall, e foi proposta por ele em 1953. Trata-se de um conjunto de 5 símbolos, separados por barras, como em A/B/c/d/E, onde A indica a distribuição de interchegada dos clientes, B o padrão de atendimento de acordo com uma distribuição de probabilidades para o tempo de serviço, c o número de canais paralelos de atendimento, d a capacidade do sistema e E a disciplina das filas.

Os valores de A e B mais comuns são: M=distribuição exponencial, D=determinístico, E_k=tipo Erlang com índice k, G=geral, etc.

Assim, por exemplo, um sistema pode ser descrito como M/D/2/∞/FCFS. FCFS é o acrônimo de *first come, first served*. Note-se que quando os dois últimos símbolos são ∞ e FCFS, eles podem ser omitidos, então o exemplo anterior também poderia ser escrito como M/D/2.

A origem de M para representar uma distribuição exponencial é que, ao usar E poderia haver confusão com a distribuição de Erlang. Assim usou-se M originado da propriedade sem memória, ou Markoviana da distribuição exponencial.

Deve-se notar que se o sistema tem múltiplos servidores, cada um com a sua fila própria e que se os clientes uma vez entrando em uma fila NÃO trocam de fila, então o sistema pode ser estudado como se fossem vários sistemas independentes. A análise matemática parte da divisão da taxa de chegada pelo número de servidores e a análise segue o visto acima.

Por exemplo, num sistema de 3 servidores com filas independentes e imutáveis, com taxa de chegada de 48 clientes/hora, então, deve-se dividir esta taxa (48 cl/hr) pelo número de servidores (3) e prosseguir a análise com um sistema individual com taxa de 16 chegadas/hora.

`atividade prática é vivo772: simulações`

Verificação e correção de códigos Na década de 70, os EUA mandaram a Marte 2 sondas, denominadas VIKING. Elas enviaram milhares de fotografias do planeta vizinho (inclusive uma que deu o que falar: parecia uma face humanóide). Após capturados e codificados, os pixels viajavam cerca de 15 minutos à velocidade da luz (300.000 Km/s) e eram recebidos, decodificados e gravados na Terra, no ritmo de 500 pixels/seg. A distância entre Marte e a Terra era de 268.200.000 Km (o dobro da Terra/Sol).

Cydonia é uma região de Marte, no hemisfério do norte do planeta, em uma zona transitiva entre as regiões fortemente cobertas de crateras do sul e planícies relativamente lisas ao norte. Algum planetologistas acreditam que as planícies do norte podem ter sido fundos do oceano e Cydonia pode ter sido uma zona litorânea (embora isto ainda seja incerto). Cydonia está coberta em numerosos planaltos escarpados alguns dos quais atraíram a atenção de cientistas e curiosos.

A Face em Marte é uma grande característica da superfície do planeta Marte localizada na região de Cydonia, 10 graus a norte do equador marciano. Mede aproximadamente 3 km de comprimento e 1,5 km de largura. Foi fotografada a 25 de Julho de 1976 pela sonda Viking 1 que orbitava o planeta na altura.

A maioria das interpretações da fotografia sugeria que seria uma formação natural, uma das muitas "mesas" da região de Cydonia. Nesta visão, a aparência da face tem origem numa combinação do ângulo de luz (com o Sol baixo no horizonte marciano na altura em que a fotografia foi tirada), a baixa resolução da fotografia que suavizou as irregularidades da superfície e a tendência do cérebro humano em reconhecer padrões familiares, especialmente caras (pareidolia). Finalmente, um buraco nos dados enviados pela Viking 1 criaram um ponto negro no local exato onde se localizaria uma narina na face humana, muitos outros destes pontos negros são visíveis na imagem.

Outra interpretação da foto é que representaria um monumento artificial criado por antigos marcianos ou outros extraterrestres visitantes do sistema solar num passado muito antigo. O livro "Message of Cydonia"(Mensagem de Cydonia) de Richard Hoagland vai mais longe e interpreta o local como sendo uma cidade arruinada com pirâmides construídas artificialmente. Uma destas pirâmides perto da face é trilateral, lisa e com uma cratera perto da base que a maioria dos cientistas crêem que são de origem natural e produzidas por milhões de anos de erosão causada pelas tempestades de areia. O local seria uma cidade e um forte em ruínas, e que a Face estava alinhada apontando para o local em que o Sol se levantava há meio milhão de anos atrás, época em que se acreditava que a Face tinha sido construída.

A interpretação cientifica ganhou fôlego com as imagens da Mars Global Surveyor em 1998 e a Mars Odyssey em 2002, que mostraram a região com uma luminosidade diferente e com uma melhor resolução e o aspecto de face quase que desaparece, o que levou a que os que suportam teorias da conspiração afirmassem que as imagens foram propositadamente alteradas.

 ©75-20, Pedro Kantek

Em 21 de Setembro de 2006 a Agência Espacial Europeia publicou novas fotografias da região de Cydonia tiradas pela Mars Express. As novas imagens têm uma resolução de menos de 14 m/pixel. O resultado ? Adeus cara.

Qualquer erro introduzido na viagem era fatal: não havia como repetir a visada. Tendo em vista esta restrição, usou-se um método forte de detecção e correção de erros, conhecido como Código Reed-Muller. Aqui, cada pixel era codificado em 32 níveis de cinza e, ao invés de serem enviados 5 bits, eram enviados 32. Isto permitia detectar e corrigir até 7 erros cometidos no envio dos 32 bits. Ou seja, o sistema se aguentava firme com até 21% (7/32) de taxa de erro, o que é altíssimo sob qualquer ponto de vista, mesmo sob um ponto de vista marciano.

O Codigo Reed-Muller usa as matrizes de Hadamard que por sua vez codificam os bits usando os códigos Gray, que implementam o conceito de distância de Hamming entre eles.

As matrizes de Hadamard são obtidas a partir da seguinte equação:
$\mathbf{H_1} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ e $H_{n+1} = H_n \bigotimes H_1$
onde $\bigotimes$ = produto cartesiano. O produto cartesiano é simples: troque cada 1 em H_n por H_1 e cada -1 por $-H_1$. Na missão Mariner, construiu-se e usou-se a matriz H_5 (32,5). Esta matriz tinha 32 x 32 bits (a distância de Hamming entre linhas vizinhas sempre era 16). Por característica matemática, nesta codificação, pode-se detectar e corrigir até $\lfloor((d-1)/2)$ erros, onde d é igual a distância de Hamming entre o código recebido e o existente na tabela.

Quando a Mariner queria enviar um certo nivel de cinza (digamos 28) ela enviava os 32 bits da linha (no caso a 28. linha) Na recuperação na Terra o string recebido era comparado com as 32 linhas possíveis. Se nenhum erro houvesse haveria um batimento completo com alguma linha: pronto, o número da linha seria o código esperado. Se 1 erro aparecesse, não haveria mais o batimento completo, mas daí o receptor deveria procurar qual a linha que tinha a menor distância de Hamming em relação ao string recebido.

Como 1 erro ocorreu, haveria uma linha com d=1. O número desta linha era o código correto. Isto funciona com $d = 0, 1, 2, 3, 4, 5, 6$ e até 7. Com 8 erros já apareceria uma ambiguidade (2 linhas seriam candidatas...)

atividade prática é vivo701: Reed-Muller-detecção e correção de erros

Bioinformática Segundo a analogia de Ridley, o genoma (dê-se como exemplo o humano) é uma espécie de livro de tamanho gigantesco: tem "23 capítulos, chamados cromossomas; cada capítulo contém vários milhares de histórias, chamadas genes; cada história é feita de parágrafos, chamados exons que são interrompidos por anúncios chamados introns; cada parágrafo é feito de palavras, chamadas codons; e cada palavra é escrita com letras, chamadas bases." Todos os seres vivos, quer se tratem de plantas, animais ou insetos, partilham código genético, o que revela um passado comum entre todas as espécies de seres vivos.

Resumo do livro Bioinformática, de Arthur M. Lesk, 2.edição: A biologia foi, até o ano 2000 uma ciência observativa e portanto subjetiva. Havia graus variáveis de precisão, sendo alguns reconhecidamente altos. Em nossos dias os dados se tornaram mais quantitativos e precisos. No caso de sequências de nucleotídeos e de aminoácidos, além disso, eles se tornaram discretos. Hoje é possível determinar a sequência genômica de um organismo de maneira completa e exata. Infelizmente, a biologia ainda não é dedutiva. Embora a vida obedeça à química e à física, ela ainda é muito complexa (e desconhecida) para que suas propriedades possam ser deduzidas a partir dos princípios básicos. Se tal característica enriquece ou empobrece a biologia, é uma questão em aberto. Os dados da bioinformática são volumosos. Em abril de 2007, os bancos de sequências de nucleotídeos já ultrapassam os 100 bilhões de pares de bases (10^{12}). O tamanho aproximado do genoma humano é de 3×10^9 letras e esta unidade tem sido chamada de human genome equivalents ou *huges*, um nome apropriado. A base de nucletídeos em 04/07 é de 330 huges. Em fevereiro de 2020, o tamanho do GenBank é de 399 376 854 872 bases e 216 214 215 sequências. (veja https://www.ncbi.nlm.nih.gov/genbank/statistics/)

Para entender esta unidade, 1 huge equivaleria a todos os caracteres impressos em 6 anos completos de New York Times. A definição de vida: "um organismo biológico é um dispositivo de ocorrência natural, que se auto-reproduz e é capaz de manipular de forma controlada matéria, energia e informação". Mais amplamente a vida na terra é um sistema auto-replicativo, complexo, distribuído no tempo e no espaço, composto por organismos distintos e finitos.

Em visões sucessivamente ampliadas, tem-se: ... → Universo → Via Láctea → Sistema Solar → Gaia → ecossistemas → organismos → tecidos → células → organelas → moléculas → átomos → ...

O genótipo de um organismo corresponde a suas informações genéticas (a sequência de seu genoma). O fenótipo corresponde 'as características (macroscópicas e bioquímicas) que se observam no organismo.

O genótipo é herdado dos pais e está sujeito a modificações por mutação ou por transferência lateral. O fenótipo depende do genótipo e é influenciado pelo meio ambiente.

A assimetria entre genótipo e fenótipo é o motor da evolução. Alterações no genótipo são hereditárias. Alterações no fenótipo (melhor nutrição levando a aumento de massa corporal ou efeito debilitante de uma lesão) não são hereditários, pelo menos não diretamente. O genótipo limita o fenótipo, mas este não é influenciado por aquele.

Alelos são diferentes sequências de um mesmo gene. Um organismo pode ter 2 cópias iguais de um gene (sendo homozigoto) ou duas cópias diferentes do gene (sendo heterozigoto). Homozigotos e heterozigotos tem genótipos

diferentes, mas se um alelo for dominante e se o gene apresentar controle exclusivo sobre uma característica, ambos terão o mesmo fenótipo.

Em uma população, organismos sexualmente reprodutivos cruzam entre si. Os indivíduos competem por recursos. Indivíduos com diferentes fenótipos apresentam sucesso diferenciado na reprodução. Com isso a nova geração tem uma distribuição alterada de genótipos e fenótipos. Isto é a seleção natural, que conta com os seguintes mecanismos:

mutações substituições pontuais, inserções e deleções, além de transposições. A taxa de mutação está estimada em 10^{-11} por par de bases por geração.

recombinação podendo criar novos alelos ou alterar a relação entre genes e elementos regulatórios.

O DNA é o material genético, o plano de desenvolvimento e das atividades em potencial dos organismos. Moléculas de DNA são cadeias lineares longas contendo sequências de 4 letras. Implícitos no DNA estão os mecanismos de auto-replicação e tradução de genes em proteinas.

A dupla hélice e sua autocomplementariedade proporcionam o mecanismo básico da herança e sua replicação quase perfeita são essenciais para a estabiliade da herdabilidade.

A sequência de aminoácidos de uma proteína determina sua estrutura tridimensional. Para cada sequência há um único estado nativo estável, o qual é adotado espontaneamente. Se uma proteína é aquecida ela desenovela (é por isso que a febre mata). A conformação tridimensional correta é importante para as proteínas desempenharem suas funções. Assim, a sequência seguinte é um paradigma:

- A sequência de DNA determina a sequência da proteína
- A sequência da proteina determina a estrutura da proteína
- A estrutura da proteína determina a sua função
- Mecanismos reguladores fornecem as quantidades corretas de funções corretas nos momentos e locais corretos.

A Bioinformática analisa os dados referentes a estes processos. Note-se que ainda falta muito (por exemplo, como uma célula se diferencia para formar diferentes tecidos). Falta ainda estabelecer a causa e o efeito em boa parte dos processos de um ser vivo. Mas, está aberto o caminho para estudar as coisas e chegar lá.

Os bancos de dados biológicos contém inúmeras informações (sequências de DNA, sequências de proteínas, genomas de organismos, mutações, bibliografias, sites web, etc), mas o principal é o de sequências de DNA, cujas instâncias são:

1. GenBank situado no NCBI (US National Center for Biotechnology Information) em Bethesda, Maryland
2. EMBL Nucleotide Sequence Database, em Hinxton, UK
3. The Center of Information Biology e DNA Databak em Mishima, Japão.

Os 3 trocam diariamente suas atualizações, a fim de manter as 3 cópias atualizadas. Além destes 3 existem inúmeros outros BDs específicos por espécies ou por aspectos específicos da Bioinformática. As consultas que se podem fazer incluem

- Dada uma sequência, encontrar sequências similares no Banco de Dados (seria um CTRL-F...)
- Dada uma proteína encontrar proteína ou parte de proteína similar
- Dada uma proteína desconhecida, achar proteínas que adotem estruturas 3D similares

atividade prática é vivo536b e 536c: tarefas em bioinformática e vivo538: expressões regulares

QRCODE O Código Reed-Solomon foi inventado em 1960 por Irving S. Reed e Gustave Solomon no laboratório Lincoln do MIT (Massachussets Institute of Technology) é um código de correção de erros. O artigo seminal é "Polynomial Codes over Certain Finite Fields." (Reed & Solomon 1960). Um algoritmo eficiente de codificação só foi aparecer em 1969. Em um processo de transmissão ou armazenamento de dados, ao se usar este código é possível recuperar algum nível de erros de transmissão. Obviamente como não existe almoço grátis só é possível ter esta possibilidade ao se adicionar algum nível de redundância na transmissão. Existem diversos códigos que fazem isso, mas o código RS (Reed-Solomon) é um bom compromisso entre eficiência e complexidade. Ele é usado em

- TV digital
- no padrão QRCODE
- na missão Voyager (1977)
- na gravação de CDs (a partir de 1982)

Ainda são muito usados, se bem que em alguns casos, estão sendo lentamente substituídos por um código mais moderno (surgido em 1993): os códigos turbo.

 ©75-20, Pedro Kantek

QR Code

Originalmente criados em 1994 por uma empresa japonesa de ar condicionado, a Denso Wave Inc para seus sistemas internos, foram padronizados como uma norma ISO (ISO/IEC 18004:2006), logo hoje são abertos e livres. A propósito, segundo seus inventores QR significa "Quick response". Trata-se de um código de barra (?) bidimensional, para ser capturado por dispositivos estáticos, dos quais talvez o mais comum seja o smartphone. Podem conter números, dados alfanuméricos, dados binários e até caracteres Kanji (afinal, os QR codes nasceram no Japão).

Eles variam na quantidade máxima de bits dependendo de sua versão. A versão 1 tem 21×21 pixels, a versão 2 tem 25×25 pixels e assim por diantes, até a versão 40 que tem 177×177 pixels (logo, a dimensão da versão $v = 21 + (4 \times v - 1)$. Existem 4 níveis de detecção e correção de erros. Em outras palavras, pode-se incluir redundância na codificação, a fim de que o QR Code seja entendido mesmo com alguma degradação como manchas, rasgões, desbotado etc. Para isso, o padrão usa os códigos de correção Reed Solomon (vivo570).

Os níveis de correção são 4: o **L**, que consegue recuperar até 7%, o **M** que recupera até 15%, o **Q** que recupera até 25% e finalmente o maior, que é o nível **H** que recupera até 30% de erros. Assim, a capacidade de um determinado código QR depende da versão, do nível de correção e do modo de dados que são codificados.

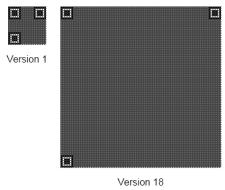
O modo de um QR Code indica os caracteres que podem ser armazenados. Se o modo é numérico, são aceitos os caracteres `0..9`. Se é alfanumérico são aceitos os dígitos `0..9`, as letras maiúsculas `A..Z` e os símbolos espaço, $ % * + - . / e :. No modo byte, aceitam-se as 256 configurações padronizadas em ISO-8859-1. Alguns scanners podem detectar UTF-8 se estes forem usados. No modo Kanji, usa-se o padrão SHIFT JIS que usa 2 bytes por caracter. Note que embora o UTF-8 permita codificar Kanji, este padrão não é usado por economia já que o UTF-8 usa 3 ou 4 bytes por caracter Kanji. Vale lembrar que é possível usar múltiplos tipos dentro do mesmo código QR. Também é possível escrever um mesmo conjunto de dados em até 16 QR Codes separados, mas ambas coisas não serão vistas aqui.

Como se sabe o QR-CODE usa os códigos de Reed Solomon, que por sua vez usam um campo de Galois-256. Portanto, além da mensagem já codificada é necessário calcular e incluir os campos de redundância, calculados pelo método RS.

Na sequência, é preenchido o total da mensagem (cujo limite foi determindo anteriormente em função da versão, modo e nível além de eventuais bits de stuffing para concluir com tamanho múltiplo de 8.

Agora, devem ser incluídos no desenho alguns padrões que ajudarão o leitor a interpretar o desenho. Começam pelos *find patterns* que são 3 blocos nos cantos superior direito, superior esquerdo e inferior esquerdo. Tais padrões consistem de uma linha preta 7×7, fora de uma linha branca de 5×5, que fica fora de um quadrado preto de 3×3. Veja o desenho:

Esses padrões são colocados nos seus lugares. Veja como ficam em um QR-Code na versão 1 e na versão 18



Agora são incluídos os separadores que são linhas brancas cercando os *find patterns*, a fim de que os mesmos possam ser isolados sem sombra de dúvida.

Agora, no canto sobrante (o direito inferior) pode ser colocado um padrão de alinhamento (chamado *alignment pattern*) que tem o seguinte jeitão

Nas versões menores ele não é colocado, nas médias é colocada 1 cópia e nas maiores, são colocados vários padrões de alinhamento.

Finalmente, os dados são incluídos, num modo extraordinariamente complexo, de maneira interlaceada, e com muitas informações de versão, modo e nivel. A exata maneira da colocação foge ao escopo desta apresentação, até porque não há espaço suficiente. Se tiver interesse procure na internet há inúmeros locais onde a informação está colocada. Um bom local é o site `http://www.thonky.com/qr-code-tutorial/` visitado em 31/01/2015. Ou `http://qrcode.meetheed.com/` na mesma data.

Para concluir, é posta uma zona de silêncio branca em volta do QR-Code. Eis a imagem final de um QR-Code 1-Q contendo "HELLO WORLD" codificado em modo alfanumérico

`atividade prática é vivo570: Reed Solomon`
`e vivo571: QRCODE`

Linguagens e compiladores Sabe-se que sempre é possível programar usando apenas 3 blocos básicos:

- A ordem simples (leia, imprima, atribua)
- A alternativa simples ou composta (se...então e se...então...senão...)
- A repetição (enquanto e repita)

A tese dos pesquisadores do início dos anos 70, Dijkstra – por exemplo, é que o desvio tendia (e tende) a gerar caminhos confusos e difíceis de seguir. Embora na época eles tenham sido muito questionados, o tempo acabou mostrando que eles tinham toda a razão do mundo: as linguagens mais modernas (como o Python) sequer têm o comando GOTO. Por esta razão todas as linguagens posteriores praticamente baniram o comando.

Um computador real O caso é que se formos olhar a linguagem de máquina dos processadores que existem por aí, arriscamos a tomar um susto. Qualquer processador tem as seguintes ordens:

na teoria	na linguagem objeto desta apresentação
Atribuição	<nome> ← expressão
Macros para fazer E/S, uma vez que do ponto de vista da máquina estas são as mais complexas operações possíveis	leia e escreva
Testes de condições. Mas, ao contrário do que estamos acostumados, a única resposta possível a um teste é um desvio	→(condição)/rotulo se a condição é verdadeira, ocorre o desvio para < rótulo >. Senão não.
Desvios incondicionais	→rotulo
	<rótulo>: estabelece um local de chegada do desvio

Um compilador de Programação Estruturada Aqui chega-se ao âmago da arte de programar. Tem-se uma camada superior que define regras apropriadas ao ser humano e que visam diminuir a chance dele (ser humano) errar. Uma camada inferior que obrigatoriamente tem que obedecer à arquitetura de hardware disponivel. Entre elas: um compilador que nós vamos construir.

O objetivo deste tema é pensar como funcionaria um compilador que recebesse programas escritos usando os blocos da Programação estruturada e os convertesse em uma implementação apropriada a rodar em um hardware real.

`atividade prática é vivo318: compiladores`
`vivo319: engenharia reversa de compiladores`

Tomografia Computadorizada Cerebral Vamos pensar em uma sessão transversal de um corpo humano. Imagine também um conjunto de emissores de raio-X, colocados paralelos em uma estrutura rígida que gira em um eixo estabelecido no centro do corpo humano. No lado oposto (ultrapassado o corpo) estão os detectores do raio-X.

À medida em que a máquina gira, inumeros conjuntos de Raios-X vão sendo tomados. Para uma determinada posição angular (digamos horizontal) tem-se um conjunto de x valores de atenuação. O valor x é a quantidade de emissores e detetores, e a atenuação é a diminuição da potência do Raio-X emitido numa ponta e recebido na outra após passar pelas estruturas do corpo que está sendo estudado.

Por exemplo, na horizontal, os raios inferiores, tem que atravessar a coluna (supondo que o paciente esteja deitado de costas), ao passo que os raios superiores ou não atravessam nada, ou apenas alguma gordura (isso para os mais gordinhos). Quando a estrutura estiver tirando raios X na vertical, a coluna do paciente atenuará os raios do meio do conjunto.

Os detectores não tiram fotografias, ao invés eles mandam sinais elétricos proporcionais a potência do Raio-X recebido diretamente para um computador. Este, de posse da informação referente ao ângulo de tomada e dos valores de todos os sensores, vai reconstituindo do corpo que está sendo "cortado".

A técnica mais simples de reconstrução é a chamada "back projection", e todas as demais que vieram depois são melhoramentos e variações desta.

Sua formulação matemática é bem compacta:

$$P(\theta, t) = \int_R f(x, y)\delta(x \sin\theta - y \cos\theta - t)dxdy$$

Essa fórmula pode ser lida como: P é a atenuação medida no sensor t, posicionado em um ângulo θ em relação ao eixo horizontal. x, y são as coordenadas dos pontos da imagem e δ é um filtro que tem a função de só considerar os pontos que estão no caminho do fluxo t de Raios X.

O coeficiente de atenuação linear média μ_t de cada pixel é comparado com o coeficiente da água, μ_a, definindo o número CT:

$CT = 1000(\mu_t - \mu_a)/\mu_a$.

A água é utilizada como referência porque seu coeficiente de atenuação é similar ao dos tecidos moles, e é um material fácil de obter para calibrar os aparelhos. O coeficiente 1000 é utilizado para obter números inteiros.

O número CT, ou coeficiente de Hounsfield, é definido como -1000 para o ar e 0 para a água. Para os tecidos em geral, ele depende da energia do feixe empregado. Por exemplo, para 80 keV, se o coeficiente de atenuação linear típico de ossos é de 0,38 cm^{-1}, e da água 0,19 cm^{-1}, o número CT dos ossos é de +1000. Pode ser ainda maior para ossos corticais. Estes valores também variam de aparelho para aparelho, já que os coeficientes dependem da distribuição de energia do feixe. A radiação observada, I, está relacionada com a radiação na fonte, I_0, por: $I = I_0 e^{-\mu x}$.

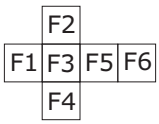
Tecido	CT	Tecido	CT
Ar	-1000	Sangue	35 a 55
Pulmão	-900 a -400	Sangue Coagulado	80
Gordura	-110 a -65	Músculo	40 a 60
Água	0	Fígado	50 a 85
Rim	30	Osso	130 a 250

Por convenção, altos valores de CT são imageados como branco, e baixos como preto. Como o olho humano não pode distinguir os milhares de coeficientes, utilizamos a técnica de janelas (windowing), para graficar somente os valores em uma certa faixa.

atividade prática é vivo750: o conceito da tomografia e sua reconstituição e vivo745: tomografia usando um resolvedor de sistemas lineares

Aplicação prática de grafos: Insanidade Instantânea Existe um quebra cabeça que recebeu o nome comercial de Insanidade Instantânea (Parker Brothers Game Co.) que consiste de 4 cubos cujas faces estão pintadas por 4 cores distintas. O objetivo do quebra cabeça é empilhar os 4 cubos formando um prisma de $1 \times 1 \times 4$. Cada uma das 4 faces do prisma deve ostentar as 4 cores. Não pode haver repetição ou falta de alguma cor em nenhuma das 4 faces.

Os 4 cubos serão representados na forma



que deve ser entendida como segue: F2 é a face superior, F4 a inferior, F3 a frontal, F6 a traseira, F1 a face esquerda e F5 a direita.

Se o quebra cabeças for resolvido por tentativa e erro, as alternativas são de $3 \times 24 \times 24 \times 24 = 41.472$.

Seja um exemplo:

```
Cubo 1:  BR      Cubo 2:  VM
  VM BR AZ VE       AZ VE VM VE
     BR       BR
```

```
Cubo 3:  VM      Cubo 4:  VE
  VE BR AZ AZ       AZ AZ VM VE
     BR       BR
```

Algoritmo de solução:

1. Construa 4 grafos (um para cada cubo), cada um com 4 vértices (VM, BR, VE e AZ).
2. Em cada grafo conecte as faces opostas (C1: VM-AZ, VE-BR e BR-BR; C2: VM-AZ, VM-BR e VE-VE; C3: VM-BR, BR-AZ e AZ-VE; C4: VM-AZ, BR-VE e AZ-VE)
3. Junte os 4 multigrafos (tem loops e/ou paralelas) em um único multigrafo, identificando cada aresta com o número do cubo original (1 a 4)
4. Localize dois subgrafos H1 e H2, com arestas disjuntas, que tenham as seguintes características obrigatórias:
   (a) Cada subgrafo deve ter os mesmos 4 vértices (VM, BR, VE e AZ)
   (b) Cada vértice deve ter grau 2 (2 arestas)
   (c) Em cada subgrafo deve haver 4 arestas, numeradas de 1 a 4
   (d) As arestas devem ser disjuntas entre H1 e H2 (i. é não podem ser repetidas).
5. Neste exemplo, uma possibilidade seria: H1=1:VM,AZ, 2:VM,BR; 3:VE,AZ; 4:VE,BR e H2=1:BR,VE; 2:VM,AZ; 3:VM,BR e 4:VE,AZ
6. Com a ajuda destes 2 subgrafos (H1 e H2) empilhe os cubos: O primeiro cubo terá vermelho na frente, azul no fundo, verde à esquerda e branco à direita.
   O segundo cubo: branco na frente, vermelho no fundo, vermelho à esquerda e azul à direita.
   O terceiro cubo: azul na frente, verde no fundo, branco à esquerda e vermelho à direita.
   O quarto cubo: verde na frente, branco no fundo, azul a esquerda e verde a direita.

Em resumo, a solução do problema pressupõe a criação dos 4 grafos individuais, a montagem do grafo coletivo e a busca de H1 e H2.

atividade prática é vivoz10: o próprio

PDF: o papel digital O padrão *Portable Document Format (PDF)* é o padrão de fato em descrição de páginas impressas, e corre a passos largos para se tornar a maneira pela qual vastos documentos são e serão produzidos de agora para o futuro. É comum hoje, ao comprar algum artefato tecnológico que exija documentação para sua operação, que o mesmo venha acompanhado de um CD ao invés de uma caixa de manuais. É óbvio que o CD conterá os mesmos (ou mais) manuais escritos em formato PDF.

Um arquivo PDF descreve como uma página ficará depois de impressa e ele é melhor do que imagem dessa página pelas seguintes razões

- A imagem como tal sempre é muito maior do que sua descrição. A razão para isso é a evidente redundância de qualquer imagem.
- A imagem como tal é fixa e só pode ser rotacionada, redimensionada ou de alguma maneira modificada à custa de processamento e perda de qualidade.
- Sobre uma imagem não há como manipular o conteúdo, por exemplo conduzindo uma busca textual (^F alguma coisa ou então ^C e ^V).
- Alterar uma única palavra sobre uma imagem exige habilidades de pintura. Sobre a descrição de uma página é tarefa simples: basta trocar uma palavra pela outra.
- Imagine uma imagem enviada para um monitor (72 DPP) ou a mesma imagem enviada para uma impressora (300 DPP). Não podem ser a mesma imagem.

A versão 1.0 do padrão PDF nasceu em 1993 quando a Adobe lançou 2 programas: o Distiller (para gerar PDF) e o Reader (para ler): ambos pagos. A coisa começou a mudar quando o departamento de rendas americano comprou uma licença que permitia aos contribuintes baixar o Reader. Rapidamente a Adobe deu-se conta de que seria melhor para ela distribuir gratuitamente o Reader. Nos dez anos seguintes o PDF acabou se tornando o padrão de fato de descrição de imagens.

Apresenta as seguintes vantagens:

- acesso aleatório: qualquer página pode ser montada independente das demais.
- linearização: todos os elementos necessários para montar uma página estão juntos.
- atualização incremental: mudanças ficam registradas no fim do arquivo. Vantagem: opção de desfazer ilimitada e rápida atualização. Desvantagem: o arquivo não pára de crescer.
- fontes incorporadas: o destino sempre será montado corretamente. A fonte é desbastada de tudo que não é necessário e vai junto. Nada é deixado por conta do ambiente leitor.

- padronização ISO: em 2008 a ISO abraçou o padrão Adobe PDF 1.7, o que o transformou em padrão aberto.
- compatibilidade para trás e para a frente: Qualquer leitor lê as versões antigas e deve ler as futuras, já que todos os leitores ignoram o que não conhecem.
- integrado a todos os principais browsers do protocolo HTTP.

```
atividade prática é vivo680: arquivo PDF
vivou10: compilador PDF
vivou11: extrator de textos planos
```

Esteganografia: Nem tudo o que reluz é ouro Trata-se da ciência (arte ?) de esconder mensagens em locais inesperados.

A idéia é comunicar mensagens que não sejam entendidas como tal. Usa-se a esteganografia, das palavras gregas que significam "escrita cifrada". Heródoto escreveu sobre um general que raspou a cabeça de um escravo, tatuou lá uma mensagem, esperou o cabelo crescer e o enviou atravessando as linhas inimigas, que não perceberam a passagem do mensageiro. As técnicas modernas são conceitualmente as mesmas, só com uma largura de banda mais alta e com latência mais baixa.

A seguir duas mensagens inocentes de um gato:

A segunda imagem contém além da imagem do gato, a íntegra do Hino Nacional Brasileiro com um total de 3060 caracteres.

O truque está em usar uma imagem BMP em true-color, que como vimos admitem 16.777.216 (2^{24}) cores. Se usurparmos um único bit de cada byte, haverá uma diminuição nas cores, mas garantidamente nenhum olho humano será capaz de percebê-la. A idéia é deixar apenas 7 bits para codificar a cor e usar o bit menos significativo de cada byte para codificar a mensagem. A quantidade de cores agora é de apenas 2.097.152 (2^{21}), mas como está se usando o bit menos importante, a diferença é imperceptível.

Não fiz isto aqui, mas a boa prática manda que a mensagem seja enviada criptografada, quando então um degrau significativo na segurança da mensagem será adicionado.

Outra observação bem importante é que esta técnica só se aplica a imagens NOVAS. Se o autor pegar uma imagem manjada na Internet e codificar nela uma mensagem, qualquer desconfiança poderá ser confirmada simplesmente rodando um FILE COMPARER (utilitário FC) sobre a imagem enviada e a que existe por aí na Internet. O FC vai apontar diferenças de conteúdo, e vai escancarar o uso da esteganografia.

Finalmente, para encerrar o nosso curso de resistência às ditaduras, segue a recomendação de criar um site dedicado ao ditador cheio de imagens laudatórias dele, de suas pretensas conquistas e essas coisas. Cada imagem dessa pode carregar mensagens secretas e tão virulentas (do ponto de vista do ditador) quanto se queira.

```
atividades práticas: vivo713 - esteganografia
vivo719: achar uma mensagem escondida
vivou42: compilador de mensagens
```

certificação de objetos binários Esta é uma técnica que usa de uma função hash, para analisar o conteúdo de um arquivo e emitir uma "assinatura"para este conteúdo. Se apenas 1 bit for modificado dentro do arquivo, a assinatura novamente gerada terá que ser completamente diferente da original. Comparando duas assinaturas para o mesmo arquivo, obtidas em dois momentos diferentes é possível garantir a integridade do arquivo entre esses dois momentos. O algoritmo que segue é uma adaptação dos padrões usualmente seguidos:

1. Considere um alfabeto composto pelos seguintes caracteres:

```
(/): 0   F : 6   L :12   R :18   X :24
 A : 1   G : 7   M :13   S :19   Y :25
 B : 2   H : 8   N :14   T :20   Z :26
 C : 3   I : 9   O :15   U :21   ! :27
 D : 4   J :10   P :16   V :22   ? :28
 E : 5   K :11   Q :17   W :23   * :29
```

2. Note que o alfabeto acima tem base=30

- Para converter conjunto de 4 caracteres (L4L3L2L1) em número faça:
  NUMERO = $L_4 \times 30^3 + L_3 \times 30^2 + L_2 \times 30 + L_1$, ou
  NUMERO = $(L_4 \times 27000) + (L_3 \times 900) + (L_2 \times 30) + L_1$
  Exemplo: ABCD = $1 \times 27000 + 2 \times 900 + 3 \times 30 + 4 = 28894$
- Para converter um número N (N $\leq$ 810000) em L4L3L2L1, faça:
  L_4 = ALFABETO[$A_1 \leftarrow$ N div 27000]
  L_3 = ALFABETO[$A_2 \leftarrow$ (N - (27000 $\times$ A1)) div 900]
  L_2 = ALFABETO[$A_3 \leftarrow$ (N - ((27000 $\times$ A1)+(900xA2)) div 30]
  L_1 = ALFABETO[N-((27000$\times$ A1)+(900 $\times$ A2)+ (30 $\times$ A3))]

3. Obtenha o arquivo para o qual se quer obter a assinatura
4. Aplique o algoritmo abaixo ao arquivo

Construindo seu assinador Para ter o seu utilitário próprio, capaz de implementar um algoritmo proprietário só seu, basta fazer as seguintes alterações:

1. Basta ler 4 caracteres ora como caracteres ora como números inteiros.
2. A base de numeração deixa de ser 30 e passa a ser 256.
3. O valor a usar nas operações de *mod* é $256^4 - 1 = 4294967295$
4. use as constantes A,B,C e D com valores iniciais secretos, escolhidos por você.
5. A resposta agora deve ser dada em hexadecimal (2 dígitos por byte), já que muitos bytes não têm representação em ASCII.
6. Alguns exemplos: HASH255('AAA') é A39649774F6A1CECF005F06919220BB0
   HASH255('Universidade Positivo') é D12588875A0FDB3E2CB210EED2C02CAF
7. Neste caso, A = h'01234567' B = h'ABCDEF01' C = h'01075533' D = h'89ABCDEF'

Autenticação de mensagens No universo digital uma necessidade premente é autenticar mensagens, onde mensagem aqui pode significar qualquer objeto digital (e binário): imagens, arquivos, mensagens trocadas, objetos que sofreram upload ou dowmload, etc etc.

As funções que fazem esta autenticação recebem o nome de *funções hash* e *hash* em inglês significa picar, cortar miúdo, triturar. Uma função hash é aplicada sobre uma mensagem (não há chaves envolvidas) e e ela gera uma assinatura do arquivo. Esquematicamente:$A = hash(M)$ As características da função *hash* são:

- M pode ter qualquer tamanho ($0 \leq len(M) < \infty$)
- A função de hash é irreversível. Isto é, dado M é fácil calcular A, mas dado A é computacionalmente inviável achar M.
- A sempre tem tamanho fixo, usualmente medido em bits.
- hash deve ter efeito cascata: Mudando 1 bit em M, espera-se que mude a metade dos bits em A e de maneira imprevisível.
- A função $hash()$ é conhecida, publicada (e testada).
- A busca de A sem conhecer M só pode se dar por força bruta (Aliás, na arquitetura bitcoin, é exatamente isto que caracteriza o $proof - of - work$ lá).
- Nada impede de implementar o $hash()$ em hardware especializado (de novo, na arquitetura bitcoin, é exatamente isto que se faz nas estações especializadas de mineração).

A é conhecido como sumário de M, pois não deixa de ser um resumo de M.

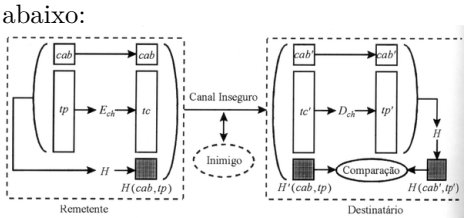
Uma boa função de hash, garante que o "inimigo" não conseguirá:

i obter informações de M a partir de A.
ii Criar um substituto de M a partir do conhecimento de A
iii Achar dois objetos M_1 e M_2 distintos que gerem ambos a mesma A

Ou seja, em termos abstratos, espera-se que uma função H_1 que gera uma assinatura A_1 de comprimento n_1 bits tenha uma dificuldade computacional para ser quebrada maior do que 2^{n_1} Na busca de i e ii acima.

Já para o critério iii a complexidade esperada é apenas $2^{n_1/2}$ já que a busca de qualquer par M_1 e M_2 é muito mais simples do que a busca de um par específico.

Em termos conceituais e especificamente no contexto de criptografia as funções hash podem ser estudadas na figura abaixo:



 ©75-20, Pedro Kantek

O remetente quer criptografar e enviar a mensagem t_p para o destinatário. Ele cria o cabeçalho *cab* e a mensagem t_p calculando o hash sobre o bloco completo. Depois ele criptografa t_p usando a chave *ch* gerando t_c e monta a mensagem a transmitir: *cab*, t_c e *hash* enviando-a por um canal inseguro. [4] O destinatário ao receber a mensagem decriptografa t'_c obtendo t'_p. além de cab' e $hash'$. O que ele precisa fazer agora é calcular o hash de $cab' + tp'$. Comparando este hash com o $hash'$ recebido pode certificar que a mensagem não foi alterada pelo possível inimigo.

Em um contexto mais simples, pode-se usar a função hash simplesmente para garantir que um texto plano qualquer não foi alterado na sua manipulação. O esquema bitcoin usa ambos os contextos acima (criptografia ou simples transmissão).

Famílias de funções *hash* Existem inúmeras funções de hash. Você pode inclusive criar a sua função, embora isto não é recomendado por mais capaz que você seja. Isso, porque as funções consagradas já foram exaustivamente testadas sendo provadas como fortes contra ataques de criptoanálise. O uso da estratégia *olhe mamãe, agora sem as mãos* pode acabar com muitos dentes quebrados...

Historicamente uma das primeiras a ser usada em larga escala foi a família MD que começou com o MD_2 que foi substituído pelo MD_4 em 1990 e depois pelo MD_5 em 1992. Todos da autoria de R. Rivest do MIT e da RSA. Todos geram assinaturas de 128 bits. O MD_4 foi desenvolvido para ser mais rápido que o MD_2 e o MD_5 veio resolver algumas deficiências do 4 (Por exemplo, foram encontradas colisões com apenas 2^{20} operações).

Outra família é a SHA desenvolvida pelo governo norte americano em 1994. O SHA_1 gera um sumário de 160 bits e seus sucessores foram aumentando este valor. No Bitcoin, usa-se a função $SHA256$ que gera um sumário de 256 bits.

```
atividades práticas: vivod69b,
vivon05 - hash no mundo bitcoin
vivon55 - árvores de Merkle
vivon60 - mineração
```

Criptografia Durante séculos, a cifra de substituição monoalfabética simples foi suficiente para guardar segredos. Neste caso, a mensagem 'ivoviuauvaeamelancia' criptografada com a chave EWTKIYVPXGOZNJLMRHDCABQFSU,

```
alfabeto = ABCDEFGHIJKLMNOPQRSTUVWXYZ
chave    = EWTKIYVPXGOZNJLMRHDCABQFSU
```

Daria origem à mensagem: XBLBXAEABEIENIZEJTXE. O desenvolvimento subsequente da análise de frequência, primeiro no mundo árabe e depois na Europa (entre os séculos XII e XVII), [5] destruiu sua segurança. Quem primeiro propôs uma melhoria no método foi Leon Alberti (1404-1472) que sugeriu usar 2 alfabetos e alternar as letras da mensagem entre eles.

Embora houvesse descoberto o avanço mais significativos dos últimos 1000 anos, Alberti não desenvolveu sua idéia, coisa que acabou sendo feita pelo diplomata francês Blaise de Vigenère (1523-1596). A cifra de Viginère não usa 2, mas sim 26 alfabetos diferentes. Agora a chave endereça 26 alfabetos cifrados, cada um deslocado uma letra em relação ao alfabeto anterior. A idéia é utilizar uma linha diferente para cifrar cada letra da mensagem original, de maneira a inviabilizar o ataque por análise de frequência. Para obter esta variação, utiliza-se uma senha.

Cifras de Playfair Uma das últimas cifras usadas antes de inventarem-se os computadores, chegou a ser operada ainda na Segunda Grande Guerra. Proposta por volta de 1850, fez sua estréia na Guerra dos Boers. Idealizada por Charles Wheatstone, quem convenceu o Governo Inglês a usar foi o Barão de Playfair, daí o nome com que ficou conhecida.

Possui as fraquezas de sempre: sucumbe a uma análise de frequência, mas por trabalhar com dígrafos (2 letras), a análise tem que ser feita sobre tais grupos. Em contrapartida, tem algumas vantagens: não precisa de tabelas, usa chaves fáceis de memorizar e trocar quando necessário, é simples de operar e consequentemente é pouco sujeita a erros. Por isso esta cifra é ótima para ser usada como "cifra de campo".

Na cifra Playfair as letras do texto plano são usadas duas a duas (o que caracteriza um bloco dígrafo). O alfabeto cifrante é colocado em uma grade 5×5, o que implica em matar uma letra. Pode-se excluir o K (substituindo por C), o W (por V) ou o Y (por I). Há ainda quem mate o J e deixe I e J valerem I. Nesta folha, vai-se excluir a letra Y substituindo-a quando necessário por I.

O alfabeto começa com as letras da chave, sem repetições. As demais casas são preenchidas com o alfabeto (menos as letras da chave) em ordem alfabética. Por exemplo, se a chave for `CURITIBAPARANA`, o início do alfabeto

[4] t_p significa texto plano e t_c texto criptografado

[5] A propósito, quem se interessar pelo método da análise da frequência, leia o conto de Conan Doyle "Os dançarinos", onde Sherlock Holmes descreve o método de maneira magistral.

cifrante é `CURITBAPN`. Eis como fica o alfabeto cifrante:

C	U	R	I	T
B	A	P	N	D
E	F	G	H	J
K	L	M	O	Q
S	V	W	X	Z

A seguir, a mensagem a criptografar deve ser dividida em grupos de 2 caracteres. Esses dois caracteres formarão a linha e a coluna da resposta. Em outras palavras, vai-se consultar cada bloco destes no alfabeto cifrante: 3 casos podem ocorrer:

No caso mais comum, duas letras formam os vértices de um retângulo. A substituição destas 2 letras é feita pelas letras que estão nos outros vértices do retângulo. Tratamento especial é necessário quando as duas letras estão na mesma linha ou na mesma coluna.

Um exemplo: seja criptografar a mensagem MINHA TERRA TEM PALMEIRAS ONDE CANTA O SABIA. Formando os blocos fica: MI NH AT ER RA TE MP AL ME IR AS ON DE CA NT AO SA BI A. O último bloco como se viu deve ser AW.

A decriptografia segue os mesmos passos, agora em ordem inversa.

Criptografia One Way Há um caso histórico famoso, que usa este método e que nunca foi quebrado. Ele é conhecido como a "cifra de Beale".

Este método é muito adequado para criptografar um único texto. Usa como chave um outro texto, que pode ser famoso e bem conhecido (por exemplo, o Hino Nacional Brasileiro) ou então um texto especialmente preparado para ser chave.

A mensagem cifrada usa números como elementos da mensagem. Cada número destes remete à letra inicial da palavra que tem este número no chave original.

Por exemplo, usando como chave o texto

Ouviram[1] do[2] Ipiranga[3] as[4] margens[5] plácidas[6]
De[7] um[8] povo[9] heróico[10] o[11] brado[12] retumbante[13],
... e assim por diante.

Poderia se criptografar a seguinte frase: 55,30,34,42,13,20,45

Que seria traduzida por "SOCORRO". Note-se que letras repetidas, vêm escritas com números diferentes, já que existem diversos locais no texto original onde a mesma letra pode ser pinçada. Este fato inviabiliza o ataque por frequência de letras.

A cifra ADFGVX A cifra ADFGVX foi introduzida no dia 5 de março de 1918, um pouco antes da ofensiva alemã de 21 de março. Tal cifra fora selecionada por um comitê de criptógrafos alemães especialmente para dar garantia a este avanço. Nessa época a artilharia alemã estava a menos de 100Km de Paris e se preparava para o assalto final.

Na noite do dia 2 de junho de 1918, conseguiu-se decifrar uma mensagem em ADFGVX. Essa mensagem levou a decifrar outra que dizia *Corram com as munições, mesmo durante o dia, se não forem vistos.* O início da mensagem indicava que fora enviada de um local entre Montdidier e Compiègne, cerca de 80Km ao norte de Paris. A necessidade urgente de munição sugeriu que alí se daria o ataque. A inteligência confirmou uma movimentação suspeita. Daí, soldados franceses foram enviados para reforçar a defesa. Uma semana depois começou a ofensiva. Tendo perdido o elemento surpresa, após 5 dias de batalhas infernais, o exército alemão foi obrigado a recuar.

É uma mescla entre substituição e transposição. A cifragem começa com o desenho de uma grade 6×6, contendo 36 células. Nelas se escrevem aleatoriamente as 26 letras e os 10 dígitos. Cada uma das linhas e das colunas assim produzidas é identificada pelas letras **ADFGVX**. Naturalmente, esta distribuição já faz parte da chave da mensagem. Suponhamos a grade

	A	**D**	**F**	**G**	**V**	**X**
A	u	n	i	c	e	p
D	f	d	y	2	s	m
F	o	0	a	q	x	j
G	k	8	7	v	t	4
V	9	b	6	h	5	z
X	3	l	1	r	w	g

o primeiro estágio, pega-se cada letra da mensagem e esta é substituída pelas letras que rotulam a linha e a coluna. Assim, "a"será substituída por "FF", "3"seria substituído por "XA"e assim por diante.

 ©75-20, Pedro Kantek

Criptografia Simétrica (DES) *DES = Data Encription Standard*, padrão do governo americano para criptografia simétrica.

Para converter uma mensagem, a primeira coisa que emissor e receptor devem combinar é a respeito de uma chave comum.
A mensagem sofre uma série de operações a partir da chave e no lado oposto tais operações são desfeitas (o método é simétrico). Esta criptografia é bastante eficiente, sendo implementada em hardware em muitos casos.

TEA O algoritmo *Tiny Encryption Algorithm* foi desenvolvido na Universidade de Cambridge em 1994, pelos autores David Wheeler e Roger Needham. Foi apresentado no Fast Software Encryption Workshop. Não está sujeito a nenhum tipo de patente.

O algoritmo TEA usa várias passagens de adição de inteiros, combinados com XOR e deslocamentos lógicos em nível de bit ($<<$ e $>>$) para obter mistura e difusão dos padrões de bit no texto puro. O texto puro é um bloco de 64 bits representado como dois inteiros longos de 32 bits no vetor *Text[]*. A chave tem 128 bits de comprimento, representado como quatro inteiros de 32 bits.

Em cada uma das 32 passagens as duas metades do texto são repetidamente combinadas com as partes deslocadas da chave e uma com as outras. O uso da operação XOR e das partes deslocadas do texto proporciona a mistura e o deslocamento. A troca das duas partes do texto proporciona a difusão. A constante *delta* que não se repete, é combinada, em cada ciclo, com cada parte do texto para ocultar a chave. Isso para evitar o caso dela poder ser revelada por uma seção de texto que não varie.

O TEA oferece criptografia de chave secreta de forma segura e razoavelmente rápida. Ele é bem mais rápido do que o algoritmo DES e o caráter conciso do programa se presta à otimização e implementação em hardware. A chave de 128 bits é segura contra ataques de força bruta. Estudos realizados por seus autores e por outros autores revelaram apenas duas deficiências muito pequenas.

Em trabalhos posteriores, tais deficiências foram removidas no algoritmo XTEA.

Uma curiosidade Quanto tempo demoraria uma tentativa de quebra do algoritmo por "força bruta" ? Suponhamos um supercomputador capaz de examinar 1.000.000.000 de chaves por segundo. Quanto tempo este computador gastaria na tarefa ?

As chaves são $2^{126} = 8.5070 \times 10^{37}$. Se o computador examina 1 bilhão de combinações por segundo, gastará 8.5070×10^{28} segundos. Em dias, serão 9.8461×10^{23} dias. Em anos serão 269757076712328700000 anos.

Criptografia Assimétrica: RSA RSA são as iniciais dos autores deste algoritmo (Rivest, Shameir e Adleman). A segurança deste método está baseada em dois princípios:

- A dificuldade computacional esperada (mais ainda não provada) em fatorar grandes números. Este fato vai ser melhor estudado nas próximas aulas, quando vocês serão desafiados a "quebrar" o método RSA.
- A garantia de que quem diz ser alguém de fato o é. Esta garantia é obtida pela certificação digital.

Trocando em miúdos, para o RSA funcionar, a chave pública de todo mundo deve ser sempre conhecida e espera-se ter certeza de que quem diz ser o proprietário do certificado é de fato quem diz ser.

Algoritmo RSA A seguir, as etapas necessárias para calcular as chaves pública e privada e depois como criptografar e decriptografar mensagens.

Para o cálculo de CP e CS há que se escolher 2 números primos p e q	Por exemplo: p = 3 e q = 7
Calcula-se n = p x q	n = 21
Calcula-se f = (p-1) x (q-1)	f = 2 x 6= 12
Escolhe-se c, tal que o mdc entre c e f seja 1	c = 5 já que mdc(5,12) = 1
Escolhe-se d tal que (c x d) mod f = 1	5 x d mod 12 tem que ser 1, d =17 já que 5 x 17 = 85 e 85 mod 12 = 1
A CP do indivíduo é (c,n)	CP = (5,21)
A CS do indivíduo é (d,n)	CS = (17,21)
Quando alguém quer mandar uma mensagem para o indivíduo, quebra a mensagem m em blocos (números)	digamos que m = 3, 14, 9
Para cifrar m, esse alguém faz $CP(m) = m^c \ mod \ n$	$3^5 \ mod \ 21 = 12$, $14^5 \ mod \ 21 = 14$, $9^5 \ mod \ 21 = 18$
Quando a mensagem chega, há que se fazer $m' = [CP(m)]^d \ mod \ n$	$12^{17} \ mod \ 21 = 3$, $14^{17} \ mod \ 21 = 14$, $18^{17} \ mod \ 21 = 9$

Neste método há que se fazer coisas como $1234567^{3456789}$ por exemplo. Não se pode usar o método canônico para isto, não haveria tempo. A chave é que estamos no conceito de grupo e ao final há uma operação de resto. Então, pode-se usar uma função recursiva e muito eficiente para tanto.

Fatoração Existem diversos métodos para fatorar um número grande. Veja-se em CORMEM, et alli, ALGORITMOS, pág 710, diversos métodos, por exemplo, a heurística ρ de Pollard (que tem custo $O(\sqrt[4]{n})$, ou uma lista completa de abordagens em ATALLAH, M, Handbook of..., página 42-8, entre outros.

atividades práticas
vivo709, vivo708 e vivo717: exemplos históricos
vivo710: simulação de DES
vivo711: algoritmo TEA
vivo712: criptografia assimétrica RSA
vivo716: vamos quebrar o RSA ?

Dinheiro virtual: o problema dos generais bizantinos No estudo de sistemas distribuídos é importante ver o que pode acontecer quando um ou mais computadores deixam de funcionar, ou pior, passam a funcionar de maneira inesperada, ou pior ainda, passam a falhar de maneira intermitente.

Esta questão foi proposta há mais de 30 anos, e recebeu um nome interessante: *O problema dos generais bizantinos*, ou também a *falha bizantina*.

O problema dos generais bizantinos é um problema teórico na Ciência da Computação que vem sendo estudado há décadas, e que teve uma possível boa resposta na recém implementação do protocolo Bitcoin. Uma referência clássica para o problema é *The Byzantine Generals Problem* de LESLIE LAMPORT, ROBERT SHOSTAK, e MARSHALL PEASE. Ela pode ser recuperada na Internet.

Na CC trata-se de obter consenso em uma rede distribuída (sem hierarquia), com nodos funcionais e *bem comportados* embora possa haver nodos *mal comportados*. Este caso engloba tanto os nodos maliciosos quanto aqueles com problemas no funcionamento.

No caso dos generais bizantinos, imagina-se um cenário de guerra no qual os exércitos bizantinos cercam uma cidade inimiga. Cada exército dos invasores é comandado por um general. A garantia de vitória só existe se todos os exércitos (ou sua maioria) atuar coordenadamente. Se apenas um exército resolver invadir sem o auxílio dos demais, certamente será derrotado.

Para facilitar o raciocínio define-se um dos m generais como sendo o comandante. É ele que pode dar a ordem de ataque. Todos os demais $m-1$ exércitos são comandados por um tenente general. Todos os generais só podem se comunicar por mensageiros e cada mensageiro só pode levar uma de duas ordens: atacar ou retirar.

Mais ainda, os generais podem ser leais ou traidores e o objetivo dos traidores é evitar que os leais se ponham de acordo. Para fazer isso, os traidores oferecem informação errônea. Por exemplo, se o comandante é traidor, pode mandar ordens diferentes para os diferentes tenentes. Se um tenente é traidor, poderia tentar enganar a outros tenentes de modo a que eles achassem que o comandante seria traidor ou seja que o comandante mandou a ordem contrária à que realmente enviou.

A falha bizantina é a classe mais geral e mais difícil de detectar. Por exemplo, a falha-parada é fácil de detectar. Já a falha bizantina pode gerar dados arbitrários, fingindo ser um componente correto.

Para resolver satisfatoriamente o problema precisamos achar algoritmos que tenham os seguintes objetivos:

- Todos os tenentes leais devem tomar a mesma decisão
- Se o comandante é leal, todos os tenentes leais executam a ordem que este lhes enviou.

Normalmente para chegar a uma solução, se impõe algumas condições adicionais

- cada mensagem enviada chega corretamente (os mensageiros são ok)
- cada receptor de mensagem conhece quem a enviou.
- a ausência de mensagem pode ser detectada
- havendo ausência de mensagem, tem-se uma ação *default*. Isso ocorre para evitar o problema do comandante ser traidor e não mandar nenhuma ordem

No artigo acima citado, os autores propõe alguns algoritmos de solução:

Mensagens orais e todos falam com todos A estratégia começa buscando saber se o comandante é traidor: Todos os tenentes se reenviam a informação que o comandante lhes enviou. Se o tenente é leal, a informação que ele retransmitirá será a que lhe enviou o comandante. A consequência de usar mensagens orais (não assinadas) é que um tenente traidor pode dizer o oposto do que o comandante lhe disse.

A seguir, apresentam-se os casos com 3 e 4 generais e os diversos papéis do traidor. Ao final, busca-se uma generalização do problema.

Para ela funcionar, tendo t traidores, se necessita que m seja maior ou igual a $3t+1$. O algoritmo se chama $OM(m)$ onde OM vem do inglês *Oral Messages* e usa a seguinte função de maioria: $M(v_1, v_2, ..., v_n)$ onde v_i é a ordem mandada ao general que estamos avaliando.

No cenário de mensagens assinadas e todos se comunicando com todos, as mensagens vão assinadas por serem mensagens escritas. Neste caso os traidores não os podem modificar e dizer que proveem do comandante. Nesta situação pode-se resolver o problema com 3 generais e um traidor. O algoritmo solução se chama SM (Signed Messages) e é o seguinte: Primeiro o comandante envia uma ordem assinada a todos os tenentes. Cada vez que um tenente recebe uma mensagem assinada ele a guarda, faz uma cópia, a assina, e reenvia a todos os tenentes que cuja assinatura não aparece na mensagem que chegou. Segundo este algoritmo os tenentes não receberão mais mensagens além de todas as combinações possíveis. Uma vez recebidas as mensagens, cada tenente toma a decisão baseando-se na ordem transmitida pela maioria.

Neste cenário, os comandantes traidores não descobertos imediatamente, já que assinam ordens contraditórias.

Outro caso é quando nem todos podem se comunicar com todos. Se falta alguma conexão, as coisas se complicam. Tudo depende agora de condições de conectividade: como sempre graus crescentes de redundância aumentam a confiabilidade. Na literatura citada, podem-se encontrar condições de conectividade necessárias e suficientes para cada caso.

Algumas observações:

- O problema dos generais bizantinos é o mesmo de se enviar dinheiro virtual sem um intermediário confiável. Bitcoin ofereceu a primeira solução prática aqui.
- No mundo real as conexões falham de maneira não deliberada. Para as detectar pode-se usar *códigos de detecção de erros*. No cenário de mensagens orais a conexão que falta pode ser considerada como um nodo traidor. Se se usam mensagens assinadas, uma falha de conexão se detectaria de maneira irrefutável.
- Para conhecer o enviante de mensagens orais só é possível se usarmos linhas fixas e não redes de comunicações. Com mensagens assinadas não há problema em reconhecer o emissor.
- A ausência de mensagem pode ser detectada usando *time-out*.
- No mundo real não se pode garantir que um erro aleatório não possa falsificar uma assinatura. No entanto, isso pode ter uma probabilidade tão baixa quanto se queira, usando-se métodos adequados.
- Evitar fraudes deliberadas se converte em um problema criptográfico. Portanto é fundamental escolher algoritmos de assinatura seguros.
- Deve-se detectar se uma mensagem é enviada duas (ou mais) vezes pela comprovação da assinatura. Assim, uma assinatura não deve poder ser gerada se o processo já a viu em instante anterior.

Para encerrar, a literatura apresenta dezenas de algoritmos que buscam sanar o problema da falha bizantina. Abundante literatura da NASA (por exemplo) mostra casos reais da falha acontecendo. Os aviões Boeing (777 e 787) usam o algoritmo com tempos de latência da ordem do microssegundo (por razões óbvias). Cada um desses algoritmos caracteriza o problema e busca a solução de acordo com os parâmetros dados pela tecnologia, plataforma, arquitetura e problema específicos.

```
atividades práticas: vivon02: generais bizantinos
vivon05a: Autenticação de mensagens,
vivon10a: Criptografia assimétrica de curva elíptica,
vivon20a: Números Randômicos ou Aleatórios,
vivon25a: sistema de numeração de Base 58
```

Dinheiro virtual: Bitcoin Premissas:

- Computação abundante: o limite é o custo do Kwh. Mas já há alternativas
  - Ethereum: O participante adquire moeda e a põe num fundo. Troca prova de trabalho por prova de participação.
  - Peercoin: procedimento similar. Mote: moeda sustentável.
- Abaixo a entidade central: surge o blockchain
- Norton: não importa... não é o bastante
- Criptografia assimétrica é a espinha dorsal
- Solução brilhante para o Problema dos Generais Bizantinos
- Matemática, matemática e ... matemática

O que é o Bitcoin?

- conjunto de conceitos
- conjunto de tecnologias
- ecossistema de dinheiro digital
- software de código aberto
- transações transferem bitcoin de A para B
- a posse se dá pelo conhecimento da senha privada
- roda na arquitetura P2P: não há autoridade central
- bitcoins criados por "mineração"
- Limite de 21.000.000 de bitcoins em 2.140
- 1 satoshi = $\frac{1}{100.000.000}$ de bitcoin

Mineração

- É um bom nome ? – não
- qualquer um pode minerar
- Prova de processamento: disputa computacional
- A solução é facilmente atribuível e remunerada por 50 bitcoins
- A plataforma se ajusta para 1 bloco minerado a cada 10 minutos
- A prova fica mais ou menos difícil em função do poder computacional alocado à mineração
- A cada 4 anos: taxa de mineração cai à metade
- Quem vence a disputa da mineração também pode ser remunerado pelas transações

Moedas digitais

- O dinheiro é bom (não é falsificado ?)
- Ninguém vai reclamar de que o dinheiro é teu e não meu ? (duplo gasto)
- no modelo convencional: segurança do papel moeda, e quem tem o papel tem a propriedade.
- TE resolve-se pela compensação de todas as transações
- antes do Bitcoin, as moedas digitais sofriam na mão do BC (e temiam os hackers)

Bitcoin

- Rede P2P descentralizada rodando protocolo Bitcoin
- Registro público de transações (a blockchain)
- Emissão de moeda de maneira matemática, descentralizada e determinística
- Verificação descentralizada de transações

 ©75-20, Pedro Kantek

A história

- gestado na primeira década do século XXI
- 2008: Bitcoin: a P2P Electronic Cash System (S. Nakamoto)
- a rede surge em 2009
- abril 2011: Satoshi Nakamoto se afasta
- em 2014: valor da rede 10.000.000.000 US$
- Hoje: há problemas – 1 bloco de 1MB a cada 10 minutos é pouco. Surge a starvation

O que se pode fazer

- Varejo de baixo valor: moeda convencional
- Varejo de alto valor
- Contratos internacionais
- Redes amplas de Doações
- Mau uso: drogas, contrabando, armas, etc etc
- Mau uso bem comum: investimento de risco (pirâmides, fraude...)

Como começar

- Ache um micro com acesso à Internet e:
  - cliente completo: armazena o histórico de todas as transações da rede desde 2009, gerencia carteiras de usuários e pode lançar transações (win, Mac e Linux)
  - cliente compacto: armazena a carteira do usuário mas depende de terceiros para acessar as transações e a rede Bitcoin
  - cliente web: armazena a carteira em um servidor de terceiros
- Para começar:
  - Gere uma senha privada e calcule a pública
  - compre alguns bitcoins (ou satoshis) que serão transferidos para o teu endereço público
  - Quando quiser gastar, deve provar a posse da moeda (pela apresentação da senha privada)
- A transação existe quando entra no blockchain: com 6 blocos adiante há um consenso de que a transação é irreversível
- Riscos: alguém roubar tua senha privada. Você perder a senha privada.

Prova de posse Cada entrada na transação que é transferida precisa uma prova de posse (assinatura digital do dono antigo). Pode ser validada facilmente por qualquer um. No Bitcoin, *gastar* é assinar uma transação de transferência para um novo endereço Bitcoin.

Esta transferência associa um valor ao novo dono através de um *script de travamento*. Este exige uma assinatura ou outra forma de validação (através de um script de destravamento) para este valor ser usado como entrada em uma transação futura. veja o exemplo
to spend = gastar (pret. e pp = spent)

Transação bitcoin Envolve três partes: recebimento do endereço destinatário, criação da transação, transmissão da transação

Gerada em algum computador no mundo (ligado ou não à rede bitcoin)

No local de **pagamento à**: endereço bitcoin (que é uma impressão digital da chave pública do destinatário)

Pode ser um usuário ou um script (um programa: logo absolutamente flexível)

Carteiras em Bitcoin Carteiras são recipientes para chaves privadas. Estas podem ser geradas de maneira aleatória (usando uma moeda), de forma pseudo-aleatória (usando pedaços de software certificados) ou através de geração determinística de chave. Neste terceiro método cada chave privada nova é derivada a partir de uma chave privada prévia usando-se uma função de hash que liga as diversas chaves em sequência. Para recriar a série de chaves você só necessita a primeira (chamada de semente, seed, ou chave mestre).

O nome carteira não é lá grande coisa. Simplesmente por que as carteiras não contém bitcoins e sim pares de chaves (privada e pública).

Nos primeiros clientes bitcoin, as carteiras eram coleções de chaves privadas geradas aleatoriamente. Este tipo de carteira é apelidada de JBOK (*just a bunch of keys*=apenas um monte de chaves) e tais carteiras estão caindo de desuso, substituídas por carteiras determinísticas. A razão é que carteiras JBOK são difíceis de administrar, fazer backups e manusear. Se quiser não reutilizar chaves (uma boa política) vai precisar gerar e manter um nonte delas, precisando fazer backups frequentes de todas elas. Lembre-se que fundos geridos por uma chave serão irremediavelmente perdidos se a carteira se tornar inacessível. Em dezembro de 2017 apareceu a notícia:

> Um técnico de informática britânico planeja escavar um aterro sanitário para recuperar um disco rígido de computador que contém o equivalente a 276,6 milhões de reais em bitcoins. O equipamento foi jogado fora por engano há quatro anos. Com o aumento astronômico da cotação da moeda virtual nos últimos tempos, a fortuna guardada nele aumentou consideravelmente... (veja, 5/12/17)

Para evitar este problema pode-se reutilizar endereços, mas isto reduz a privacidade na medida em que associa múltiplas transações e endereços uns com os outros. Em resumo, não é uma boa idéia usar carteiras JBOK, também conhecidas como carteiras do tipo-0.

As carteiras determinísticas são conhecidas como HD (carteira determinística hierárquica) e elas têm formato de árvore. Na raiz está a semente original e dela derivam as senhas filho. Cada senha filho pode gerar infinitas senhas neto e assim por diante, sem limite.

Uma vantagem adicional desta estrutura de árvore é que ela pode ser usada para exprimir significado organizacional adicional, como por exemplo quando um ramo específico é usado para receitas de um estado e outro ramo para receitas da matriz ou ainda para trocos de pagamentos feitos.

Outra vantagem da carteira HD é que usuários podem criar sequências de chaves públicas sem precisar acessar as chaves privadas correspondentes. Isto permite que carteiras HD estejam em um servidor inseguro ou que alguém possa gerar muitas chaves públicas.

`atividades práticas:`
`vivon35a: Carteiras em Bitcoin,`
`vivon40a: Scripts de transação,`
`vivon55a: Árvore de Merkle`

Dinheiro virtual: mineração de bitcoins Montado o candidato a bloco novo na blockchain, agora começa uma disputa entre todos os mineradores para ver quem chega antes num objetivo comum: a prova de trabalho. A disputa se dá entre funções de hash (já estudadas) e o objetivo é chegar a um hash que tenha uma determinada característica. Quando alguém alegar ter achado um resultado satisfatório, qualquer participante poderá (e deverá) refazer o hash e comprovar a alegação.

Lembrando o que já estudamos de hash, é virtualmente impossível, a partir de um resultado esperado, montar um bloco que chegue a este resultado. O que se pode fazer é criar um campo que varie dentro do bloco, e a cada valor do campo calcular o hash e verificar se ele satisfaz o critério pedido.

Lembrando que o hash256 sempre gera uma resposta de 256 bits independente do tamanho da entrada.

Note o que acontece quando se acrescenta uma pequena variação ao final do texto do qual se vai calcular o hash. Para participar de um desafio vai-se definir um alvo arbitrário: encontre uma frase que produza um hash hexadecimal que comece com um zero. Olhando uma série como a acima, ve-se que a frase terminada em 7 gerou um hash como o esperado. Pela estatística poderia se esperar este resultado uma vez a cada 16 tentativas (já que há 16 dígitos hexadecimais de `0` a `F`. Em termos numéricos isto significa achar um hash que é menor do que

`0x1000000000000000000000000000000000000000000000000000000000000000` Chama-se este limite de `alvo` e o objetivo é achar um hash que seja numericamente menor do que o alvo. Ao diminuir o alvo, a tarefa de encontrar um hash que seja menor se torna cada vez mais difícil.

Quando você alega que achou o hash pedido com um determinado nonce, todos devem verificar o sucesso: basta calcular o hash 1 vez, ao passo que para achar o nonce vencedor foram necessários 7 hash. Na dificuldade atual do bitcoin os mineradores têm que fazer quatrilhões de tentativas de hash até achar um nonce que resulte um hash baixo o suficiente. A seguir algumas computações reais:

Dificuldade em bits	Tempo decorrido	Sucesso no nonce:
8 (3 bits)	0.0004 seg	9
16 (4)	0.0005 seg	25
4194304 (22)	13.3	1759164
8388608 (23)	110.1	14214729
16777216 (24)	195.2	24586379
67108864 (26)	665.0	84561291

Em 2013, a rede tentava achar um hash que fosse menor do que 000000000000004c296e6376... . Precisava-se calcular cerca de 150 quatrilhões de hashes. Nessa época a rede já tinha uma capacidade de processamento de 100 petahashes por segundo (PH/seg) o que permitia encontrar um bloco vencedor a cada 10 segundos. A dificuldade no bloco 277.316 tinha o valor de alvo=0x0000000000000003A30C0000... zeros até o fim. Em binário este número teria os primeiros 60 bits definidos como zero. Com este nível de dificuldade um único minerador processando 1 trilhão de hashes por segundo (1TH/seg) encontraria uma solução em média a cada 8496 blocos ou uma vez a cada 59 dias.

A questão é como a dificuldade é ajustada e quem a ajusta. A definição original (1 bloco a cada 10 minutos) é a frequência cardíada do bitcoin e precisa ser mantida para sempre. Mas, o número de mineradores e principalmente a sua capacidade computacional pode se alterar.

A cada bloco a dificuldade é reajustada examinando quanto tempo levou para achar os últimos 2016 blocos e o compara com o tempo esperado de 20160 minutos (2 semanas com 1 bloco a cada 10 min). A razão entre os tempos é calculada e um ajuste (para cima ou para baixo) na dificuldade é realizado. O reajuste só é feito a cada 2 semanas, e tem um limite de 4×. Se o reajuste for maior ou menor do que 4× ele será carregado para o próximo ciclo de 2 semanas. Isto é para evitar volatilidade extrema na dificuldade. Note-se que a principal influência no mercado de mineração é o preço de um kilowatt-hora em bitcoin: ele determina a rentabilidade da mineração.

Quando um minerador acha um hash menor que o alvo ele imediatamente o propaga na rede. Ao ser recebido pelos demais nodos ele é validado e estando tudo certo ele determina o fim da disputa por esse bloco e o início pela disputa do próximo. Só depois de validado é que ele é retransmitido para os demais nodos. Eis uma lista das validações que são feitas:

- A estrutura do bloco é sintaticamente válida
- O hash é menor do que a dificuldade
- o timestamp é menos do que 2 horas no futuro
- o bloco é menor que 1MB
- a primeira (e só a primeira) transação é uma geração de B$
- todas as transações do bloco são válidas usando o checklist de transações

A seguir, o poder de mineração ao longo dos primeiros 5 anos:

2009 0.5MH/seg a 8MH/seg. Crescimento de 16×

2010 8MH/seg a 116GH/seg. Crescimento de 14.500×

2011 16GH/seg-9TH/seg. Crescimento de 562×

2012 9TH/seg-23 TH/seg. Crescimento de 2.5 ×

2013 23TH/seg-10PH/seg. Crescimento de 450×

2014 10PH/seg-150PH/seg. Crescimento de 15×

O crescimento da dificuldade introduziu uma questão: todos os valores de *nonce* eram percorridos sem que o desafio fosse resolvido. Inclui-se no timestamp o tempo gasto (o que permitiria reinicializar a busca pelo nonce em cada tempo). Mas, novo encalço: o ciclo todo de *nonce* era percorrido em menos de 1 segundo (com o mesmo timestamp, portanto). A solução foi usar a transação coinbase (a primeira) como fonte de valores extra para *nonce*. Por enquanto está aguentando...

`atividades práticas: vivon60a: Mineração e Prova de Trabalho`

Dinheiro virtual: Blockchain – bifurcação e disputa Como já se viu, a disputa dos mineradores é pela busca de um bloco que tenha um hash menor do que o objetivo buscado. No detalhe, a busca é por um valor de nonce que inserido no bloco sendo minerado gere um hash menor que o alvo. Por exemplo, se o alvo é 0000000000000003A30C0000... Na mineração do bloco 277.316 (por volta de 2013) o nonce 4.215.469.401 produz um hash que é 0000000000000002a7bbd25a... Em outras palavras, este bloco acabou de ser minerado com sucesso. O minerador imediatamente envia o bloco 277.316 para toda a rede. Quem o recebe, antes de reenviá-lo, faz uma validação, como se verá a seguir e se a validação der OK, inclui este nodo na sua blockchain local, interrompe a própria mineração do bloco 277.316 e imediatamente começam a minerar o bloco 277.317.

Validação de um novo bloco Isto garante somente blocos válidos (bons) serão propagados. Alternativamente, garante aos mineradores honestos que eles terão seus blocos incorporados à blockchain, recebendo sua recompensa pelo trabalho. Os mineradores que agirem desonestamente terão seus blocos rejeitados e não apenas deixarão de receber a recompensa como terão gasto energia elétrica à toa. Os critérios de validação incluem:

- A estrutura do bloco candidato é sintaticamente válida.
- O hash do bloco candidato é menor do que a dificuldade de destino.
- O timestamp do bloco é menor do que 2 horas no futuro.
- O tamanho do bloco está nos limites aceitáveis.
- Somente a primeira transação é de geração de coinbase.
- Todas as transações do bloco são válidas e a soma das taxas de transação está correta.

Aqui é possível responder a uma pergunta razoável: *porque um minerador não inclui uma transação mandando pagar 1000 BTC de recompensa ao invés de calcular a recompensa correta (por hipótese 25 BTC) no ano que corre ?* A resposta é que todos os nodos validam o bloco candidato de acordo com as mesmas regras fixas. Uma transação coinbase inválida, invalida todo o bloco,[1] Portanto, todos os partipantes da rede P2P do Bitcoin invalidarão este bloco candidato. Ele nunca entraria na blockchain. Daqui, que a validação independente é um componente chave do consenso descentralizado.

O último passo do consenso descentralizado é o aumento da blockchain. Os nodos mantém 3 conjuntos de blocos:

- a blockchain principal
- correntes secundárias: em algum momento passado construíram ramificações da blockchain
- blocos órfãos: ainda não tem um pai ou seja, o pai ainda não chegou
- blocos inválidos: rejeitados assim que qualquer teste falhar, não são incluídos em nenhum conjunto pendente (daí que os 4 são apenas 3).

A corrente principal, a blockchain, a qualquer momento é a cadeia que tiver a maior dificuldade acumulada. Possivelmente será a blockchain que tiver maior quantidade de blocos. As ramificações passadas são guardadas, caso uma dessas correntes for estendida e passar a ter maior dificuldade. Isto ocorre quando blocos distintos são minerados com sucesso quase ao mesmo tempo.

Quando um bloco novo é recebido, um nodo qualquer irá tentar incluí-lo na blockchain. No bloco vem o campo *hash do bloco anterior*. É a referência ao bloco pai deste. Na maioria das vezes o pai é o topo da blockchain, ou seja este novo bloco vai estender a blockchain e segue a vida. Se não for, o bloco recém chegado estende uma cadeia secundária e daí o nodo que está analisando isto precisa decidir: qual das duas cadeias tem maior dificuldade: a campeã passa a ser a blockchain e a perdedora passa a ser uma cadeia secundária.

Se um bloco válido é recebido e nenhum pai é encontrado nas cadeias existentes ele passa a ser um órfão. Fica na cadeia de órfãos até seu pai ser recebido, quando então ambos são integrados. Isto ocorre quando dois blocos são minerados em um pequeno intervalo de tempo e como a rede é assíncrona, o filho chega antes do pai.

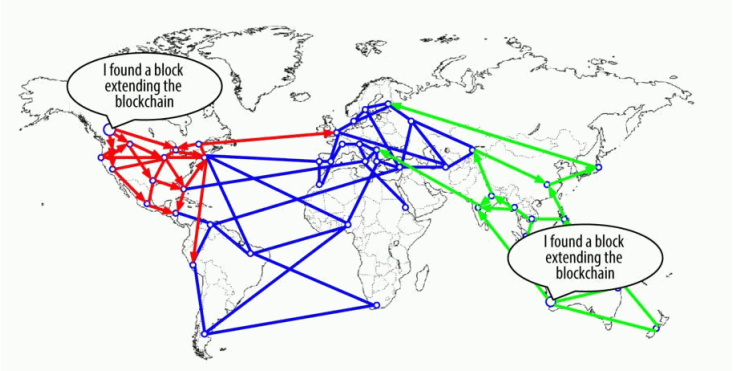
Como todos selecionam a corrente com mais dificuldade, todos os nodos acabam alcançando o consenso. As discrepâncias temporárias sendo resolvidas lá na frente, à medida em que a prova de trabalho maior se impõe. Os nodos mineradores "votam" com seu poder de mineração ao escolher qual corrente estender através da mineração do próximo bloco.

Forks (bifurações) da blockchain Ao somar a dificuldade registrada em cada bloco de uma corrente qualquer nodo pode calcular a quantia total de prova de trabalho que foi dispendida. No exemplo a seguir vamos estudar como isso ocorre. A separação geográfica foi colocada apenas para facilitar o entendimento, lembre que o que interessa aqui é o aspecto topológico e não o geográfico. Na rede bitcoin verdadeira a distância entre nodos é calculada em saltos (*hops*) e não em sua localização física. Veja como a rede está agora:

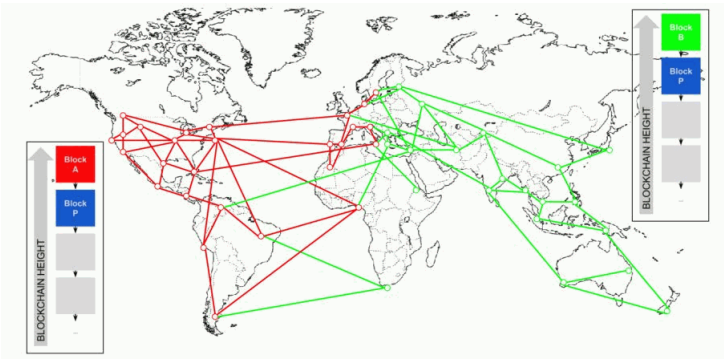
[1] vale aqui aquela brincadeira: por falta de um prego, meu cavalo perdeu uma ferradura, por falta de uma ferradura, perdi um cavalo. Por falta de um cavalo perdi uma batalha, por falta de uma batalha perdi uma guerra, por falta de uma guerra, perdi um reino. Isto leva à equivalência: um prego ≡ um reino.

 ©75-20, Pedro Kantek

Nesta situação a rede converge, todos têm o bloco azul no topo da blockchain.



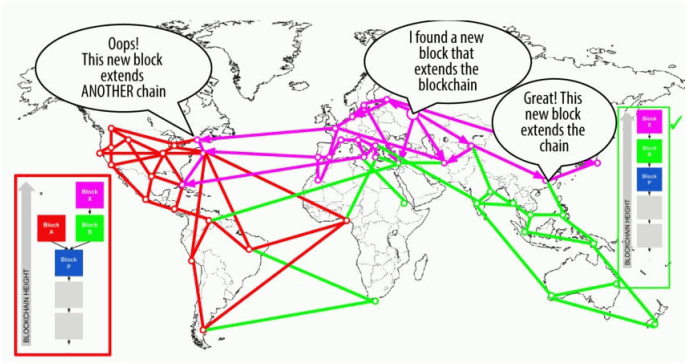
Agora dois mineradores encontraram ao mesmo tempo um bloco. Um deles achou o bloco vermelho no Canadá enquanto outro achou o bloco verde na Austrália.



À medida que o verde e o vermelho são propagados alguns recebem o vermelho e outros recebem o verde antes. A rede se bifurca em 2 perspectivas diferentes de blockchain.

Note que o poder computacional da rede se divide. Parte vai tentar achar o sucessor do verde enquanto a outra parte vai buscar o sucessor do vermelho. Ainda que o poder computacional se divida ao meio, é improvável que o empate persista. Imagine-se que a turma que está com o verde no topo acha uma solução: o bloco rosa. Para estes, agora, a solução é azul-verde-rosa. Eles imediatamente propagam o rosa e a rede o enxerga como uma solução válida.

Os verdes vão simplesmente estender a corrente em mais um bloco. Os nodos vermelhos enxergam agora duas correntes: azul-verde-rosa e azul-vermelho. A corrente azul-verde-rosa é maior (mais dificuldade acumulada). Com isso os nodos vermelhos redefinem as correntes: a azul-verde-rosa passa a ser a principal e a azul-vermelha como secundária.



Qualquer minerador que estiver trabalhando em estender a corrente azul- vermelho terá que parar porque seu bloco candidato é órfão já que seu pai (o vermelho) não está mais na corrente principal.

É teoricamente possível que uma bifurcação se estenda por 2 blocos, mas a chance disso é muito pequena. Uma bifurcação de um bloco pode ocorrer semanalmente, mas uma bifurcação de dois é extremamente rara.

O intervalo de bloco de 10 minutos é um ajuste equilibrado entre tempos de confirmação rápidos (liquidação de transações) e a probabilidade de uma bifurcação.

```
as atividades práticas aqui podem ser:
vivon65a: Filtros de Bloom,
vivon75a: Bifurcação e disputa de blockchain. (veja os anexos)
```

GPS No dia 3 de setembro de 1989, o Maracanã estava lotado para assistir ao Brasil X Chile, pelas eliminatórias da Copa do Mundo de 90. Longe dali, em algum ponto a princípio entre Marabá e Belém, Cezar Augusto Garcez comanda um vôo cego. Perdido em pleno ar, tenta se posicionar. Localiza, em vão, uma rádio que transmitia a partida que entraria para a história como "o jogo da fogueteira". Aquele vôo também estaria nos jornais no dia seguinte: "Avião desaparece na Amazônia", publicou O Globo.

Herói e vilão, ao mesmo tempo, o comandante Garcez é a principal personagem do RG-254 que caiu na selva amazônica em 1989. O que deveria ser um vôo rotineiro se transformou numa tragédia. Desorientado, Garcez permaneceu durante três horas em vôo cego. Temendo que o erro fosse descoberto, passou diversas informações truncadas para a base, afirmou estar onde não estava. Sem combustível, arriscou o aparentemente impossível: um pouso na copa das árvores, em plena noite, com visibilidade praticamente nula. Garcez foi acusado de negligenciar rotinas básicas da aviação. Por outro lado, salvou a vida de muitos passageiros ao conseguir aterrissar a aeronave e cuidar dos feridos. Ainda hoje, aguarda julgamento.

A desorientação completa em voo, embora rara (todo o treinamento dos pilotos é para impedir isso), de vez em quando dá as caras. Há um episódio famoso, durante a crise dos mísseis em Cuba, em que um piloto de U2, Charles Maultsby, resolveu se perder no Polo Norte. Desorientado, acabou entrando mais de 500 Km em território da antiga União Soviética.

Teve uma sorte dos diabos, se achou e retornou, mas o dia escolhido por ele para se perder não podia ser pior: Foi dia 27 de outubro de 1962, o dia em que o mundo esteve mais próximo da IV Guerra Mundial, que seria também a última: a humanidade teria sido aniquilada. Aliás, os russos juravam que a missão do U2 era preparar a invasão dos mísseis.

GPS Na década de 70, os EUA começaram a projetar e implementar o sistema conhecido como GPS (Global Positioning System). O primeiro satélite *Navstar* foi lançado em 1978. O objetivo principal era permitir que os estimados 40.000 usuários militares americanos se orientassem sobre a terra. Os civis começaram a usar o sistema a partir da década de 80. Hoje, estima-se em mais de 50.000.000 de dispositivos capazes de receber os sinais de GPS e apresentar uma estimativa de localização.

Além do sistema americano, existe o russo *Glonass* de uso exclusivo pelos militares e a partir da última semana de 2005, o sistema *Galileu* construído e operado pela Agência Espacial Européia. Há também o chinês, chamado *Beidou*.

A grande explosão de uso, vem quando celulares e automóveis começam a vir de fábrica com a possibilidade de informar sua localização. Associados a sistemas de mapas embarcados, podem informar ao motorista qual caminho tomar para ir virtualmente a qualquer lugar sobre a superfície terrestre. Associados a telefones celulares, podem informar a localização do telefone. Associados a sistemas de proteção contra roubo de automóveis, informam a localização em tempo real e assim por diante. A lista de aplicações possíveis parece inesgotável. Seja como for surge uma tendência muito forte que é, ao cadastrar um imóvel, cadastrar também suas coordenadas geográficas.

Todos os sistemas acima se baseiam na triangulação (ou quadrangulação) do receptor em relação a 3 ou 4 (ou mais) satélites em órbita da terra.

O sistema começa pela identificação de diversos pontos fixos sobre a superfície terrestre. Tais pontos contam com estações transmissoras e permitem a cada satélite a recepção de diversos pontos. Medindo o tempo gasto por cada sinal desde a estação terrena até o satélite, e fazendo isso com diversas estações, o satélite conhece sua real posição com precisão de centímetros.

Cada satélite possui um relógio atômico, integrado à rede GPS e que garante precisão de tempo de 1 bilionésimo de segundo.

Todos os satélites emitem 2 mensagens: A primeira, envia a identificação do satélite, sua posição no espaço e sua hora interna. Uma segunda mensagem é uma sequência de impulsos digitais em um padrão inconfundível.

Tudo funciona como se o receptor também gerasse no mesmo instante de tempo a mesma sequência de impulsos digitais. Da análise dos dois sinais (o gerado pelo satélite e o gerado internamente no receptor) é possível medir o retardo do sinal do satélite. Conhecendo a velocidade da luz (fixa e igual a 299.792.458 m/seg) e conhecendo o tempo gasto pelo sinal do satélite ao receptor, é possível estabelecer a distância entre satélite e receptor.

Daqui, pode-se inferir que o receptor está sobre a esfera de centro x, y, z (as coordenadas no espaço do satélite) e de raio R, que é a distância entre o receptor e o satélite.

Ao fazer o mesmo cálculo com um segundo satélite, obtem-se uma segunda esfera, e o receptor está sobre o círculo de intersecção das duas esferas.

Finalmente ao fazer o cálculo com um terceiro satélite, obtêm-se uma terceira esfera. O receptor ainda pertence às 3 esferas, e a intersecção das 3 apresenta apenas 2 pontos no espaço. Um deles apenas sobre a superfície terrestre, e logo esta é a posição do receptor, com precisão de metros.

Problemas Se os receptores pudessem ter um relógio atômico 3 satélites bastariam. Como um relógio atômico é muito caro e pesado, os receptores usam relógios de quartzo comum (sujeitos a erros de 1 seg/dia).

Então usa-se um quarto satélite para refazer os cálculos e permitir acertar o atraso/adiantamento do relógio local ao relógio da rede.

Se o usuário estiver em movimento, os sinais ainda vêm afetados pelo efeito Doppler, que precisa ser analiticamente corrigido antes de estabelecer a posição do receptor. Idem para os efeitos relativísticos associados.

O sistema americano usa duas radiofrequências: L1 e L2. A L1 é chamada Sinal Civil, embora os militares também a usem. A L2 é de uso exclusivo militar, já que seus códigos (a segunda parte das mensagens) não são públicos e são protegidos por criptografia. Ao usar apenas o sinal L1, receptores conseguem determinar sua posição com erros de 5 a 10 m, causados pela ação da ionosfera. A camada de ar sobre a terra aumenta a densidade da atmosfera, causando refração nos sinais. É como um lápis colocado dentro de um copo de água.

A potência de um satélite é de 500W, o equivalente a 5 lâmpadas de 100W. Apenas para efeito de comparação, o sinal de TV de um satélite é 1 bilhão de vezes mais forte. Este fato permite que interferências (às vezes propositais) obscureçam ou até invalidem o sinal dos satélites. Este fato é usado em teatros de guerra, quando as forças americanas introduzem geradores locais de ruído centrados na banda usada pelo sistema. Essa é a razão pela qual embora o sistema seja público, russos e europeus tratam de ter cada um o seu.

`atividade prática é vivo007: Localização no espaço`

Demonstração automática de teoremas Como se viu a Lógica permite uma descrição bastante precisa do mundo ou pelo menos de um mundo em particular. O tema de estudo agora é como usar a lógica para recuperar e produzir conhecimento novo sobre este mundo.

Vai-se pegar alguns conceitos emprestados da matemática. As premissas iniciais sobre o mundo vão receber o nome de axiomas. Este nome será usado tanto para fatos como para regras, desde que ambos sejam dados.

A partir deles, ou baseado neles, vão ser feitas perguntas, que poderão ser respondidas como SIM ou NÃO. Tais perguntas vão receber o nome de TEOREMAS e vai-se estudar aqui o processo pelo qual a Lógica nos habilita a demonstrar um teorema. O significado disto é afirmar, ao final, que o teorema é válido ou seja que a resposta à pergunta que ele faz é SIM.

Este exercício está preparando o caminho para o estudo da linguagem PROLOG, já que esta faz exatamente a mesma coisa que este exercício, só que de maneira automática.

Grosso modo, o processo sempre é o mesmo e composto das seguintes etapas:

preparação

1. Reescrever as informações contidas no enunciado na forma de Lógica de Predicados. Utilizar apenas cláusulas de Horn (apenas uma cláusula após a implicação).
2. Eliminar $\rightarrow$, trocando-a por $\sim$ A $\vee$ B.

3. Reduzir o escopo de $\sim$, usando
   (a) $\sim\sim$ A equivale a A
   (b) $\sim$ (A $\wedge$ B) equivale a $\sim$ A $\vee \sim$ B
   (c) $\sim \forall$ x P(x) equivale a $\sim$ P(x)
   (d) $\sim \exists$ x P(x) equivale a $\forall$ x $\sim$ P(x)
4. Renomear as variáveis e passar os quantificadores para a esquerda
5. Retirar os quantificadores
6. Transformar as expressões em conjunção de disjuntos

demonstração

1. Acrescentar o que se quer provar NEGADO.
2. Localizar pares de afirmações contraditórias.
3. Se for possível efetuar substituições de modo que as afirmações contraditórias se refiram ao mesmo objeto, substituí-las e criar um novo teorema, onde as afirmações contraditórias são cortadas (simplificadas)
4. Prosseguir nesse ciclo até obter a cláusula NIHIL. Se isto foi possível o teorema original está demonstrado.
5. Se não for possível, não é possível demonstrar o teorema original.

A propósito vai-se estabelecer uma regra de escrita que será bem útil no PROLOG: variáveis serão escritas em maiúsculo e constantes em minúsculo.

```
atividades práticas
vivo100 - demonstração de teoremas lógicos
vivo103 e 104 - demonstração automática de teoremas
vivo860b e c - demonstração usando PROLOG
```

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. Exemplos desta abordagem estão no projeto Euler e são os problemas 67, 81 e 82.

Seja um exemplo: achar a 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: em

```
 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.

```
atividade prática é vivo671: multiplicação de matrizes
vivo672: algoritmos memoizáveis
vivo240: problema da mochila
vivo265: programação dinâmica
vivo666d: mais exemplos
```

Compressão digital 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*.

Uma árvore binária completa, chamada de árvore de Huffman é construída recursivamente a partir da junção dos dois símbolos de menor probabilidade, que são então somados em símbolos auxiliares e estes símbolos auxiliares recolocados no conjunto de símbolos. O processo termina quando todos os símbolos foram unidos em símbolos auxiliares, formando uma árvore binária. A árvore é então percorrida, atribuindo-se valores binários de 1 ou 0 para cada aresta, e os códigos são gerados a partir desse percurso.

A codificação de Huffman tem desempenho ótimo quando as probabilidades dos símbolos são potências negativas de dois ($2^{-1}, 2^{-2}, ...$). A codificação gerada tem também a garantia de ser não ambígua, pois nenhum código pode ser o prefixo de outro código, e aqui está o pulo do gato do método.

Este algoritmo começa pela análise das frequências de ocorrência de cada um dos caracteres do arquivo a comprimir.

Suponhamos, a título de exemplo, um arquivo que contenha `ABCDEFABCDEABCABAABD`

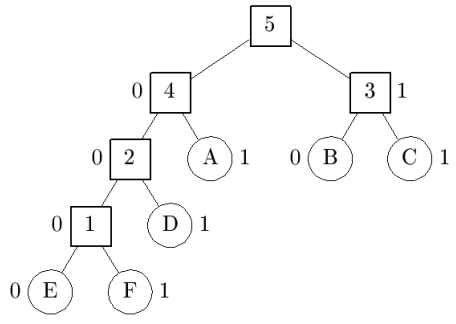
Contabilizando as ocorrências de cada caractere, ficamos com as seguintes frequências: `B=25\%, E=10\%, A=30\%, C=15`

A seguir, o algoritmo prevê a colocação desta tabela em ordem decrescente de ocorrências. No exemplo, fica: `A=30\%, B=25\%, C=15\%, D=15\%, E=10\% e F=5\%`.

Na terceira etapa, é criada uma árvore binária, onde as folhas representam os caracteres a comprimir, e o número de nós é o número de folhas diminuído de uma unidade. A regra para a montagem da árvore é: cada nó terá dois sub-nós, sendo que o caractere mais frequente ficará sempre à esquerda. Se houver frequências iguais entre nó e folha, colocamos à esquerda o nó. Vejamos no exemplo:

Comparam-se os primeiros (de baixo para cima, ou seja os menores), dois elementos, E (10%) e F (5%). Então o E fica à esquerda e o F à direita. Agrupamos o E e o F sob um nó, a quem chamaremos 1, que passará a ter a frequência dos dois somados (15%). Continuamos a buscar os 2 menores valores, que agora são o nó 1 e o caractere D. (Note que aqui há um empate, assim escolhem-se quaisquer dois menores). Agrupamos os 2, formando o nó 2, ao qual estão subordinados o nó 1 (a esquerda) e a letra D (a direita). O nó 2 tem frequência 30%.

Os próximos 2 menores são o B e o C. Cria-se o nó 3 que os agrega, com frequência 40%. Os próximos são o nó 2 e a letra A, que criam o nó 4, com frequência 60%. Finalmente encerra a árvore o nó 5, que agrega os nós 4 e 3, com frequência 100%.



O que resultaria nos seguintes códigos de bit

A	01
B	10
C	11
D	001
E	0000
F	0001

A partir desta árvore, cada caractere a transmitir será enviado pelos bits que permitem o caminhamento pela árvore, entendendo-se o bit zero como "esquerda"e o bit 1 como "direita".

No nosso exemplo, ao transmitir a mensagem `ABAEC`, enviaríamos `01.10.01.0000.11`, com os pontos colocados apenas por clareza. Note-se que os caracteres de maior frequência terão comprimento menor, o que caracteriza a compressão e o consequente ganho em termos de transmissão. Por outro lado, olhando a mensagem, sem conhecer a árvore original é muito difícil determinar onde começa e termina cada caractere, o que torna a sua quebra mais difícil.

Para determinarmos o comprimento do arquivo a mandar, e que em última análise determinará a economia pela compressão, usamos a regra do comprimento ponderado da árvore gerada. Isto significa que cada nó deverá gerar uma multiplicação entre o comprimento em bits do caractere representado no nó pela sua frequência de ocorrência.

No nosso exemplo, isto ficaria:
$A = 2 \times 0.30 = 0.60$
$B = 2 \times 0.25 = 0.50$
$C = 2 \times 0.15 = 0.30$
$D = 3 \times 0.15 = 0.45$
$E = 4 \times 0.10 = 0.40$
$F = 4 \times 0.05 = 0.20$
A somatória destes valores é, em bits, o comprimento médio de cada caractere neste arquivo. No nosso exemplo, a soma deu 2.45, e como são 20 caracteres no total, o comprimento do arquivo comprimido (2.45×20) é 49 bits. Existe um teorema, que não será demonstrado, que informa ser a árvore construída segundo o critério visto acima, a árvore de menores caminhos ponderados, ou seja a árvore de Huffman é a solução ótima para este problema.

Um método mais sofisticado de compressão usado pelo BZIP2, e um dos campeões de compressão é a Transformada Burrows-Wheeler, depois a transformada Move To Front (MTF) e finalmente, o algoritmo de compressão de Huffman, nosso velho conhecido.

O método de compressão de Burrows-Wheeler foi descrito em 1994. Bastante recente, portanto. Ele é baseado em uma pesquisa (não publicada) de Wheeler em 1983. O método é composto pela combinação de três algoritmos:

Inicialmente, submete-se os dados desejados a uma função de transformação, a"Burrows-Wheeler Transform", ou simplesmente BWT, que reordena os bytes orignais, tornando-os bastante propícios para compressão.

A seguir, aplica-se a heurística "Move To Front", (MTF). Aplicá-la logo após o BWT faz com que os dados de saída contenham muitos zeros e grande tendência para números positivos pequenos.

Finalmente submete-se os dados resultantes a algum método de compressão que atue sobre estatísticas dos dados (por exemplo, código de Huffman).

Descrição do algoritmo "Burrows-Wheeler Transform" O BWT opera em blocos de dados, e não em streams que lêem poucos dados de cada vez. Quanto maior o tamanho dos blocos, maior a taxa de compressão atingida. O algoritmo recebe um vetor A de tamanho N com os dados a serem transformados e devolve um vetor B (permutação dos elementos de A) e um índice k

Ele opera gerando todas as N rotações à direita possíveis para o vetor A, formando uma matriz N × N. A seguir, ordena estas N linhas e copia o último caracter de cada uma para o vetor B. Atribuí a k o número da linha da matriz ordenada que corresponde à A.

A saída da BWT é o vetor B e a variável k. Mas como obter a sequência original A a partir de B e k? Note que a partir de B pode-se obter a primeira coluna da matriz N x N. Para isto, basta ordenar os dados de B.

Agora, monta-se o vetor que leva a resposta → resposta ordenada.

Para recuperar o string original:
* recupera-se k (no caso $k = 5$)
* obtem-se a k-ésima letra do vetor resposta (não ordenado): (no caso 'I')
* faz-se $k \leftarrow INDICES[k]$ onde INDICES é o vetor que leva da resposta para a resposta ordenada.
* Sucessivamente para todas as respostas.
* Finalmente inverte-se a sequência obtida e eis a mensagem original.

Descrição da heurística "Move-To-Front" Esta heurística consiste em montar uma fila F de caracteres, onde o mais recentemente utilizado é movido para o topo. A fila é inicializada com os elementos ordenados por seu próprio valor. Para cada caracter, armazenamos sua posição na fila naquele instante e, em seguida, movemos tal caracter para o início da mesma.

A transformada MTF é um codificador de dados, tipicamente um conjunto de bytes desenhado para melhorar a performance da entropia em técnicas de compressão. Quando eficientemente implementada, ele é rápido o suficiente, de modo que seus benefícios justificam incluí-lo como etapa extra em algoritmos de compressão.

Etapa final: compressão dos dados Para compressão dos dados resultantes, podemos usar código de Huffman ou algum codificador aritmético, sobre sub-blocos da sequência de dados original.

Compressão JPG

Ao contrário da compressão de texto ou de programas, quando não é aceitável a perda de qualquer informação, no caso de imagens (e sobretudo no caso de vídeo) é perfeitamente aceitável algum nível de degradação da imagem. Isso ocorre devido a dois fatores:

- Nossos olhos (e ouvidos) não são capazes de distinguir mudanças sutis e principalmente
- Imagens (e vídeos) são objetos muito grandes que praticamente exigem serem comprimidos para um manuseio adequado.

As etapas pelas quais uma imagem passa antes de ser guardada em memória como um objeto JPG são as seguintes:

1. A imagem é dividida em blocos de 8×8 pixels. A razão é diminuir a quantidade de operações aritméticas necessárias durante a compressão (ainda que pagando para isso ter uma compressão ligeiramente ineficiente).
2. Cada bloco de imagem sofre a ação da transformada discreta do cosseno. Este procedimento (reversível) gera outra matriz 8×8, mas com uma propriedade interessante. Apenas os valores mais à esquerda e acima é que têm valores significativos. Embaixo e à direita, o valor mais comum é zero.
3. Os dados são quantizados visando diminuir o comprimento dos valores e aumentar a redundância. Aqui ocorre a degradação na compressão, já que este processo é irreversível.
4. Na compressão, os dados quantizados são percorridos em zig-zag. A seguir processos de compressão, tais como run-length, Huffman, QM (esta última protegida por patente) são aplicados gerando o arquivo final.

É de se notar que a maioria das câmeras digitais de fotografia (ou como dizem os franceses: *photo numérique*) já fazem todo este processamento, a fim de economizar memória.

Ao longo do processo, inúmeras otimizações são feitas para torná-lo mais eficiente. A primeira já citada é trabalhar com blocos de 64 valores ao invés de trabalhar com a imagem inteira. Uma segunda otimização ocorre quando os cossenos (uma função transcedental) são previamente calculados e armazenados, sendo depois apenas consultados. Uma terceira otimização ocorre quando o processo é manuseado algebricamente a fim de que as iterações sejam convertidas em um produto matricial de matrizes. Finalmente, usa-se aritmética de ponto fixo (fazendo os ajustes necessários) já que esta sempre é muito mais eficiente que a aritmética de ponto flutuante.

Transformada discreta do cosseno

Baseada no conceito de localidade de referência espacial (probabilidade alta de dois pixeis vizinhos terem valores próximos ou iguais) esta transformação atua muito eficientemente na preparação de dados de fotografias para compressão. A fórmula da transformada bidimensional é

$$G_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} P_{xy} cos\left(\frac{(2y+1)j\pi}{2n}\right) cos\left(\frac{(2x+1)i\pi}{2n}\right)$$

Onde n é igual a 8, $C_f = 1$ quando $f > 0$ e $C_f = \frac{1}{\sqrt{2}}$ quando $f = 0$. G_{ij} é o valor na matriz transformada, P_{xy} é o valor do pixel correspondente na imagem, e i, j, x e m variam entre 0 e 7.

A transformada inversa discreta do cosseno bidirecional (conhecida como $IDCT$) apresenta a seguinte formulação:

$$P_{xy} = \frac{1}{4} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} C_i C_j G_{ij} cos\left(\frac{(2x+1)i\pi}{2n}\right) cos\left(\frac{(2y+1)j\pi}{2n}\right)$$

Onde n é igual a 8, $C_f = 1$ quando $f > 0$ e $C_f = \frac{1}{\sqrt{2}}$ quando $f = 0$. P_{xy} é o valor do pixel correspondente na imagem, G_{ij} é o valor na matriz transformada, e i, j, x e m variam entre 0 e 7.

Algoritmo Lempel-Ziv Welch (LZW) Acrônimo dos nomes dos seus inventores Lempel, Ziv e Welch. Lempel e Ziv publicaram suas idéias em 1977 (ZIV, Jacob, LEMPEL, Abraham; Compression of Individual Sequences Via Variable-Rate Coding, IEEE Transactions on Information Theory, September 1978) e Terry Welch refinou o algoritmo em 1984. (WELCH, T. A. (June 1984). "A technique for high-performance data compression."Computer. Vol. 17, pp. 8-19.)

Em síntese, o algoritmo troca conjuntos de caracteres por códigos simples. Não há análise prévia do conteúdo e a compressão ocorre quando um conjunto grande de símbolos é trocado por um único código. Os códigos LZW podem ter qualquer número de bits, mas precisam ser maiores do que a representação de 1 caracter (ou seja, maiores do que 8 bits). Os primeiros 256 códigos são assinalados para o conjunto padrão inicial. Em um exemplo com códigos de 12 bits, de 0 a 255 são caracteres simples e de 256 a 4095 são para sequências.

Este algoritmo foi o escolhido para comprimir imagens do tipo GIF.

O clássico problema dos algoritmos baseados em dicionários é a memória necessária. Quanto mais dados do arquivo são lidos, mais sequências precisam ser armazenadas no dicionário, e com isso, necessita-se de mais memória. O outro problema que isso causa é de espaço de endereçamento: os códigos das entradas no dicionário crescem junto com ele (em geral ocupam $log_2(N)$ bits, onde N é o tamanho, ou número de entradas, do dicionário). Várias abordagens podem ser adotadas para lidar com esse problema, as mais simples são:

- Congelamento do dicionário (quando o dicionário atinge o tamanho limite, ele fica ”congelado”e mais nenhuma entrada pode ser adicionada nele).
- Esvaziamento (o dicionário é esvaziado e a compressão começa toda de novo). Essa técnica pressupõe que é mais vantajoso usar a redundância local que a redundância global para a compressão. Corresponde a separar o arquivo em ”blocos”comprimidos separadamente.
- Uso de uma política de esvaziamento que seja comum tanto ao compressor quanto ao descompressor (apagar entradas mais antigas, por exemplo).

```
aplicações práticas:
vivo451 e vivo452 - huffman
vivo453 - huffman sem análise prévia (dinâmico)
vivo455 - mais huffman
vivo456 - huffman modificado
vivo457 - huffman aplicado a uma imagem BMP
vivo458 - os lusíadas
vivo690 - padrão JPG
vivo731 - algoritmo LZW
```

Desenho no monitor O desenho em vidro *drawing glass* é a tecnologia de desenhar sobre um terminal plano de vídeo com uma caneta que interaja com este monitor. Reproduz muitas das idéias e técnicas do desenho comum (lápis sobre papel) mas tem diferenças importantes.

Usando uma caneta sensível, inclusive ao quanto de pressão é aplicada sobre ela, um software reproduz sobre um *papel virtual* os movimentos e efeitos aplicados sobre a caneta, obtendo ao final uma imagem digital artística.

As vantagens deste procedimento sobre o desenho convencional podem ser assim descritas

- A cópia (ou melhor dizendo a calcografia) fica facilitada, já que o software pode projetar sobre o monitor o que se está copiando, com graus variáveis de transparência. Um eventual refinamento deste proceder é capturar a cor original na hora do desenho.
- Eventuais erros podem ser facilmente corrigidos, através de comandos `undo`. Este faz o papel de uma borracha virtual e pode ser acionada muitas vezes.
- Ao final, o desenho já é um arquivo digital, sem precisar ser capturado.
- A arte pode ser construída em camadas, com todas - exceto a última - transparentes.
- Desenhos podem ser duplicados, copiados, rotacionados, esticados, comprimidos, distorcidos à vontade.
- Cores podem ser mescladas, trocadas, clareadas e escurecidas sempre via digital, vale dizer, matemática.
- Por ser um método virtual, este procedimento permite mesclar com sucesso diversas técnicas (por exemplo, pintura a óleo com aquarela) com resultados impressionantes, o que no desenho convencional soa como herético.
- Uma vez amortizado o custo da plataforma, não há consumo de insumos (papel, giz pastel, pincéis, etc), que têm custos bem salgados.

É claro que há desvantagens (não existe almoço grátis), não existe aquele ambiente de studio artístico associado ao desenho. Em resumo, desenhar no monitor está para desenhar em papel e lápis assim como um livro PDF está para um livro tradicional.

Parte associada a este projeto é a escolha do software de apoio. Vão ser citados dois deles: o primeiro é o Smooth Draw, que é *freeware* e aparece aqui por ser o software que originalmente foi usado na produção de aulas da Academia Khan. (`https://pt.khanacademy.org/`). Junto com um capturador de vídeo, este produto permite que o professor vá falando e escrevendo de maneira que o leitor/aluno veja as fórmulas e textos sendo desenhados na tela enquanto ouve o professor.

Outro produto é o Essential Painter,
(https://www.painterartist.com/en/product/painter-essentials/) que não é gratuito, mas que tem custo acessível: cerca de 300 reais, bem menos que a plataforma de hardware necessária. Ele serve como porta de entrada para um produto *estado da arte* que é o Corel Painter, este sim já bem mais caro: mais de 1000 reais. Uma vantagem do Essential Painter é que ele pode ser instalado e usado por 30 dias, com todas as suas funcionalidades sem nenhum custo. Assim procedendo o interessado pode avaliar antes de se comprometer com investimentos e custos.

```
Aplicações práticas:
Para esta atividade estão previstas 2 atividades
(obviamente sujeitas à disponibilidade de plataforma -- hardware
e software:
1. A produção de uma vídeo aula, nos moldes da Khan Academy
2. A calcografia de uma imagem, obtendo-se ao final um desenho copiado
da imagem original.
```

Google Com um logotipo colorido (meio infantil) sobre fundo branco, o Google tornou-se indispensável a centenas de milhões em todo o mundo. O uso do verbo googlar em português, inglês, alemão, espanhol – um monte de idiomas – remete a uma ação tão simples quanto quase intuitiva. Muitas das vezes o Google é a porta de entrada para acessos à Internet. Este texto é uma revisão da história dessa empresa e de sua estratégia altamente bem sucedida para criar um nicho de mercado, ocupá-lo e praticamente impedir que concorrentes a ameacem. Embora nunca tenha gasto explicitamente um dólar sequer em se promover ou anunciar sua marca, ela é conhecida em todo o mundo, superando barreiras de idioma, de cultura ou geográficas. Essa é uma conquista e tanto, principalmente quando comparadas a marcas tradicionais. A Coca Cola investe muito dinheiro em publicidade desde o início do século XX. A Google passou a existir como empresa em meados da década de 90. De uma empresa de fundo de quintal, a Google se transformou em empresa de capital aberto em 2004, através de uma IPO (Initial Public Offer), com a ação cotada a US$ 85,00. Foi a maior IPO tecnológica já vista e levantou 2 bilhões de dólares. Em menos de um ano a ação chegou a US$ 300. Já chegou a mais de 650 dólares. Nos sete anos seguintes a fundação da Google (1998), as ações da Microsoft praticamente mantiveram seu valor. Nesse mesmo período as ações da Google passaram a valer mais de 80 bilhões de dólares. No dia em que escrevo este texto (12/02/2020), a ação está cotada a US$ 1215,41 (NASDAQ:GOOG). Um detalhe interessante é que grandes empresas de tecnologia e de buscas (Yahoo, Alta Vista, entre outras) foram abordadas pela Universidade de Stanford e recusaram a chance de comprar o sistema de busca do Google por 1 milhão de dólares. Suas recusas levaram os doutorandos Sergey Brin e Larry Page a largar o curso e abrirem eles próprios a empresa. Esses dois são os grandes responsáveis pelo sucesso. Primeiro cercaram-se de ótimos administradores para poderem se dedicar ao crescimento e inovação da Google. Além disso, Brin é um excelente negociador enquanto Page é bom em buscar reduções de custo (em particular com a eletricidade, um dos maiores custos da google). Empregados na google trabalham preferencialmente em grupos de 3 e todo empregado destina 20% do seu tempo a projetos de interesse pessoal. Esta é uma estratégia trazida do meio acadêmico em que professores têm 1 dia por semana para se dedicarem ao que lhes interessa. Essa combinação de ambiente, visão de longo prazo, lucros, recurso computacional mais que abundante funciona como um poderoso chamariz dos melhores cérebros, ao final o maior insumo em uma empresa de alta tecnologia

```
a atividade prática é vivob02
```

Xadrez e computador O xadrez é um jogo de 2 adversários, jogado sobre um tabuleiro formado por 64 casas, alternadas entre as cores branco e preto. A maneira correta de dispôr o tabuleiro é deixando as casas brancas à direita dos dois jogadores

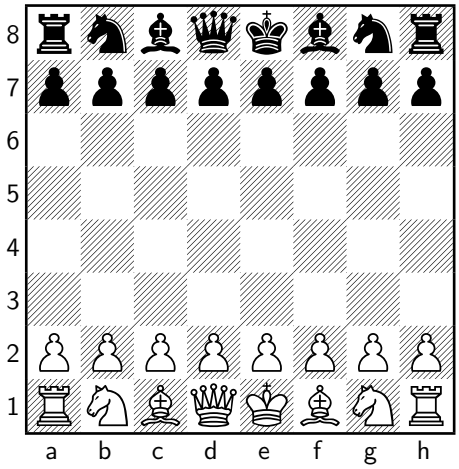
Existem dois jogos de peças, um na cor branca e outro na cor negra. Na representação do tabuleiro em textos (na vertical), o jogador das brancas fica abaixo e o das negras fica acima.

Para representar (e posteriormente estudar) partidas, há a necessidade de identificar cada uma das 64 casas do tabuleiro. Existem diversas maneiras de fazer isto, mas aqui usar-se-á a notação algébrica reduzida (também conhecida como SAN = *short algebric notation*).

Nela, o tabuleiro é dividido em linhas (de esquerda à direita de ambos os jogadores) e colunas que são as filas que vão de um jogador até o outro. As colunas são conhecidas pelas letras latinas minúsculas, começando com a letra **a** à esquerda do jogador de brancas.

As linhas são numeradas de 1 a 8, sendo a linha 1 a mais próxima do jogador das brancas e a linha 8 a mais próxima do jogador das negras.

As peças cada jogador possui um exército composto de 8 peões, 2 torres, 2 cavalos, 2 bispos, 1 rei e uma dama, totalizando 16 peças. Do ponto de vista de cada jogador, as peças são assim colocadas:



As brancas sempre começam o jogo, e depois negras e brancas vão se alternando. Com exceção de um lance chamado roque, a cada lance apenas uma pedra pode ser movimentada. Uma peça pode ocupar uma casa originalmente vazia ou então uma casa que esteja ocupada por um adversário, caso em que ele *toma* a peça do adversário após o movimento. Somente a peça que vai ser jogada pode ser tocada. Se se precisar tocar uma peça que não vai ser jogada, usa-se a fórmula francesa *j'addoube*

O objetivo único do jogo é tomar o rei adversário, quando então o mesmo (o adversário) perde.

Os movimentos das peças são:

rei pode se mover 1 casa em qualquer uma das oito direções possíveis, desde que não ocupe nenhuma casa ameaçada pelo adversário

dama pode se mover quantas casas quiser em qualquer uma das oito direções possíveis, desde que não existam peças pelo caminho. Se forem adversárias, a dama poderá tomá-las. Se forem da mesma cor da dama, elas bloqueiam o movimento. Por exemplo, no início do jogo, as duas damas estão completamente bloquedas.

torre pode se mover quantas casas quiser na fila ou na coluna em que a torre estiver.

bispo pode se mover quantas casas quiser nas diagonais que se cruzam na casa onde o bispo está. Note-se que dos dois bispos, um deles anda só em casas brancas e o outro só em casas negras.

cavalo o cavalo é a única peça que pode saltar sobre peças amigas ou inimigas. Seu movimento simula um "L", já que ele anda duas casas na linha ou coluna onde estiver e mais uma casa em diagonal. O cavalo potencialmente pode fazer 8 movimentos distintos.

Seja um cavalo no tabuleiro de xadrez. Supondo uma matriz de 5×5, e imaginando que o cavalo ocupe a posição central (a 3,3), os saltos que ele pode dar são os seguintes (numerados de 1 a 8).

	8		1	
7				2
		♞		
6				3
	5		4	

peão pode andar uma unica casa para a frente. Enquanto estiver na sua posição inicial (e só nesta) poderá andar 2 casas. O peão toma seu adversário de lado. O peão que conseguir chegar ao final do tabuleiro será promovido e poderá se transformar em qualquer peça, a critério do jogador que o promoveu. A nova peça ocupará a casa onde o peão foi promovido. A promoção é notada com um "=", após o movimento do peão, seguido pelo símbolo da peça na qual o peão promovido se transforma.

O roque é o único movimento do xadrez que permite alterar a posição de duas peças, que são sempre o rei e uma torre. Existe o roque pequeno (o rei e a torre do rei são movimentados) e o roque grande (o rei e a torre da dama são movimentados). O resultado do roque pequeno é que o rei branco vai para a casa **g1** e a torre do rei vai para a casa **f1**. Nas negras, o roque pequeno leva o rei para **g8** e a torre do rei para **f8**.

Já o roque grande, leva (nas brancas) o rei para **c1** e a torre da dama para **d1**, e nas negras o rei para a casa **c8** e a torre da dama para a casa **d8**. O roque pequeno é representado por "0-0"e o grande por "0-0-0".

Os roques só podem ser feitos nenhuma destas condições estiverem presentes

- O rei ou a torre tenham sido movidos previamente

 ©75-20, Pedro Kantek

- exista alguma pedra entre o rei e a torre
- exista alguma casa por onde o rei a torre se movimentarão que esteja ameaçada pelo adversário
- o rei esteja em cheque

Quando o rei adversário é ameaçado, é obrigatório avisar "cheque" (representado na notação por um sinal de mais "+" após o lance). Quando um rei é ameaçado ele obrigatoriamente deve ser protegido (sair do cheque). Isto pode ser feito de 3 maneiras

- movendo o rei ameaçado
- interpondo um peça qualquer entre a peça que deu o cheque e o rei (é claro, desde que a peça ameaçadora não seja um cavalo)
- tomando a peça que deu o cheque

Se o rei não puder sair do cheque, diz-se então ter sido dado um cheque mate (representado por duas cruzes) e então o jogo acaba com a vitória de quem deu o cheque mate.

O jogo não precisa obrigatoriamente seguir até um cheque mate ou empate. A qualquer momento, qualquer um dos dois oponentes podem abandonar o jogo, desde que não visualizem como vencê-lo ou sequer empatá-lo. No tabuleiro este fato é representado quando o abandonante derruba seu próprio rei na sua hora de jogar.

O empate ocorre nas seguintes circunstâncias:

- pode ser oferecido a qualquer momento por qualquer um dos contendores. Cabe ao oponente aceitar ou não. Além deste caso que é opcional, todos os demais são obrigatórios.
- quando nenhum dos jogadores possui material para vencer o jogo (por exemplo, rei contra rei, rei e bispo contra rei, rei e cavalo contra rei).
- quando o rei é afogado: o único movimento que resta a um jogador deixaria o seu próprio rei em cheque.
- cheque perpétuo: Um jogador (usualmente o que está superiorizado) não consegue escapar de cheques sucessivos do adversário.
- tripla repetição: Se uma dupla de jogadas for repetida 3 vezes.
- cinquenta lances: durante o desenvolvimento da partida, 50 lances forem feitos sem que ocorram movimentos ou capturas de peões.

Um campeonato de xadrez, usualmente denominado *match* é um conjunto de partidas. Cada vitória rende 1 ponto ao vencedor e zero ao perdedor. Empates determinam meio ponto a cada um dos contendores. Usualmente o match é jogado até um determinado número de pontos, quando se declara o vencedor ou o empate.

A anotação de um lance se faz escrevendo a letra inicial da peça movida, seguido da casa para onde a peça foi movida. A exceção é o peão, que quando movimentado não deve ser identificado por P (de peão).

Quando houver ambiguidade num certo movimento (por exemplo, os dois cavalos podendo atingir a mesma casa, ao se identificar a peça que se move, escreve-se a sua coluna (ou linha) antes de indicar para qual casa ele se move.

Quando um peão tomar outro *en passant*, anota-se o movimento com a observação "ep" logo após.

A notação algébrica em português usa as seguintes letras: P=peão, T=torre, C=cavalo, B=bispo, D=dama e R=rei. A notação algébrica em inglês, que tem a sua importância por ser a interface usual em livros estrangeiros, softwares de xadrez, etc etc é um pouco diferente e usa: P=peão (pawn), R=torre (rook), B=bispo (bishop), N=cavalo (knight), Q=dama (queen), K=rei (king).

O jogo de xadrez costuma ser dividido em 3 fases, cada uma com suas peculiaridades.

abertura compreende algo como os primeiros 10 lances do jogo. Visa sobretudo desenvolver as peças e estabelecer uma posição no tabuleiro. Lembremos que as torres, os bispos, a dama e o rei estão no início do jogo completamente bloqueados pelas peças da sua cor.

meio o meio-jogo é a consequência da abertura e visa (a) conseguir vantagem material; (b) criar debilidades no campo inimigo e (c) estabelecer combinações (ataques orquestrados de diversas peças). Em geral esta fase assiste a trocas sucessivas de peças, levando ao

final com o tabuleiro já quase vazio, o final assiste ao embate final entre os 2 reis. Ganha importância grande o peão (passado) ou a disposição de peões que permita passar algum, pela sua substituição por peça mais poderosa. Divide-se em finais "ganháveis" (rei e dama contra rei, rei e torre contra rei, entre outros) e finais "não ganháveis" (rei e cavalo contra rei, rei e bispo contra rei, entre outros)

As estratégias de consecução variam em cada uma das 3 etapas: na abertura, grande parte dos movimentos é dicionarizado constituindo-se as milhares de aberturas conhecidas, sendo que as mais famosas ganham até nome: Ruy Lopez, dos 4 cavalos, ...

Já no meio-jogo, a estratégia em estado bruto aparece. Há espaço no tabuleiro, as peças estão mais desenvolvidas. Armadilhas, blefes, tudo vale para ganhar uma posição melhor, alguma qualidade, ou preparar o bote. Muitas partidas terminam aqui, quando não há defesa possível a alguma posição. Entretanto, se os jogadores tiverem nivel equivalente esta fase é reconhecida pelo troca-troca de peças.

No final, um conjunto de regras (orientações) estabelece o que o jogador com a posição melhor deve fazer para obter mate. Outro conjunto de regras auxilia quem está pior. De novo, há necessidade de planos e de talento. As regras fornecem apenas orientações gerais.

No xadrez, há um número grande de movimentos estúpidos ou simplesmente sem sentido. A boa heurística é a que mostra um número tratável de possibilidades interessantes. Um bom contraponto a esta última afirmativa mostra como é quando alguém é dotado de habilidade heurística: "o grande Ricardo Reti (um jogador de xadrez) dramatizou a natureza de sua maestria quando alguém lhe perguntou quantos lances previa: "Um", respondeu, "o certo".

Jogadores automáticos de xadrez Esta é uma antiga tradição em IA: jogar xadrez. Desde Shannon (*Programming a computer for playing chess*, 1950) a Turing (*Can a machine think*, 1951) espera-se do jogo um excelente ambiente para desenvolvimento de teorias, técnicas e métricas da IA. O furor causado pelo match Deep Blue x Kasparov em 1997 ilustra também a importância do xadrez no panorama da IA.

Na tabela a seguir, extraída de Russel 96 (1^a edição em língua espanhola, pág. 146) a pontuação obtida por campeões humanos e automáticos ao longo dos anos:

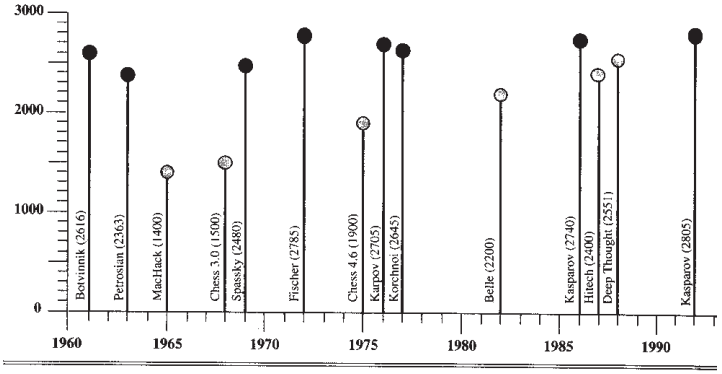


Figura 5.12 Calificación de campeones ajedrecistas, humanos y máquinas.

O primeiro programa capaz de jogar com humanos foi o MacHack 6 de 1967. Conseguiu uma avaliação FIDE de 1400, melhor que a de um iniciante, mas ainda bastante longe da necessária para jogar "a sério". Para avaliar o número FIDE imagine uma escala onde o iniciante consegue 1000 pontos, enquanto o campeão mundial tem por volta de 2800 pontos.

Em 1970, começa o *ACM North American Computer-Chess Championship*, campeonato americano de programas. Em 1974 começa o *World Computer Chess Championship*. A propósito o primeiro foi disputado em Estocolmo e foi vencido pelo Kaissa, um programa russo. Ele é disputado em média, a cada 4 anos.

Saindo do estado básico (minimax, com alfa-beta, início dicionarizado e final baseado em regras) caso em que os jogadores eram algo melhor que iniciantes, o primeiro grande desenvolvimento se deveu não ao software, mas ao hardware. **Belle** o primeiro jogador baseado em hardware especialmente desenhado foi criado em 1982. Nele tanto o gerador de jogadas quanto o avaliador estavam implementados em circuitos especializados. Belle conseguiu 2.250 pontos, o equivalente ao nivel de mestre.

Em 1980, estabeleceu-se o prêmio Fredkin: 5000 dólares ao primeiro programa que conseguisse pontuação de mestre, (ganho por Belle em 1983), 10000 dólares ao primeiro programa a alcançar 2500 pontos, pontuação próxima à Grande mestre, (ganho por Deep Thought em 1989) e 100000 dólares ao primeiro programa vencedor do campeão mundial humano (Deep Blue em 1997).

HITECH criado pelo ex-campeão mundial de xadrez postal Hans Berliner e por Carl Ebeling também tinha hardware especializado e um avaliador extraordinariamente complexo. Conseguia examinar 10 milhões de posições por jogada. Foi o campeão mundial de programas de 1985 e em 87 foi o primeiro programa a ganhar de um grande mestre humano (Arnold Denker) quando este era um dos 800 melhores jogadores do mundo.

Deep Thought foi originalmente desenvolvido na Universidade de Carnegie Mellon. Rapidamente a IBM contratou parte da equipe e esta lançou o Deep Thought 2: este utiliza um avaliador simples, mas examina cerca de 5 bilhões de jogadas, o que lhe permite ir às profundidades 10 ou 11 do jogo. Em algumas posições, ele avança mais (certa ocasião, achou um mate em 37 jogadas). Nessa época sua pontuação foi FIDE de 2600, o que o colocava entre os 100 melhores enxadristas do mundo.

O **Deep Blue** foi desenvolvido por Murray Campbell, Feng-Hsiung Hsu e Joseph Hoane na IBM, tomando como base o projeto Deep Thought, desenvolvido anteriormente por Campbell e Hsu na Carnegie Mellon. A máquina vencedora era um computador paralelo com 30 processadores IBM RS/6000 executando a "busca de software"e 480 processadores VLSI especializados para xadrez, que executavam a geração de movimentos (incluindo a ordenação de movimentos), a "busca de hardware"para os últimos níveis da árvore, e também a avaliação de nós folhas. O

Deep Blue buscava 126 milhões de nós por segundo em média, com uma velocidade máxima de 330 milhões de nós por segundo. Ele gerava até 30 bilhões de posições por movimento, alcançando rotineiramente uma profundidade igual a 14. O coração da máquina é uma busca alfa-beta de aprofundamento iterativo padrão com uma tabela de transposição, mas a chave de seu sucesso parece ter sido sua habilidade de gerar extensões além do limite de profundidade para linhas suficientemente interessantes de movimentos forçados. Em alguns casos, a busca alcançou uma profundidade de 40 jogadas. A função de avaliação tinha mais de 8.000 características, muitas delas descrevendo padrões de peças altamente específicos. Foi usado um "livro de aberturas"com aproximadamente 4.000 posições, bem como um banco de dados de 700.000 jogos de grandes mestres a partir do qual podiam ser extraídas recomendações consensuais. O sistema também utilizava um grande banco de dados de finais de jogos com posições resolvidas, contendo todas as posições com cinco peças e muitas com seis peças. Esse banco de dados tem o efeito de estender de forma significativa a profundidade efetiva da busca, permitindo ao Deep Blue jogar com perfeição em alguns casos, até mesmo quando está a muitos movimentos de distância do xeque-mate.

O sucesso do Deep Blue reforçou a ampla convicção de que o progresso dos jogos de computadores vinha principalmente de um hardware cada vez mais poderoso - uma visão encorajada pela IBM. Por outro lado, os criadores do Deep Blue afirmam que as extensões à busca e a função de avaliação também eram críticos. Além disso, sabemos que vários aperfeiçoamentos algorítmicos recentes permitem que programas executados em computadores pessoais padrão vençam todos os campeonatos mundiais de xadrez por computador desde 1992, muitas vezes derrotando oponentes maciçamente paralelos que poderiam buscar um número de nós 1.000 vezes maior. Diversas heurísticas de poda são usadas para reduzir o fator de ramificação efetivo a menos de 3 (comparado ao fator de ramificação real de aproximadamente 35). A mais importante delas é a heurística de movimento nulo, que gera um bom limite inferior sobre o valor de uma posição, usando uma busca rasa na qual o oponente chega ao dobro de movimentos no início. Esse limite inferior frequentemente permite a poda alfa-beta sem o custo de uma busca em profundidade total. Também é importante a poda de futilidades, que ajuda a decidir com antecedência que movimentos causarão um corte beta nos nós sucessores.

A equipe do Deep Blue recusou a oportunidade de uma revanche com Kasparov. Em vez disso, a competição de 2002 apresentou o programa FRITZ contra o campeão mundial Vladimir Kramnik. A competição de oito jogos terminou empatada. As condições da competição eram muito mais favoráveis ao ser humano, e o hardware era um PC comum, não um supercomputador. Ainda assim, Kramnik comentou: "Agora ficou claro que o principal programa e o campeão mundial são aproximadamente iguais."

Segundo Russel e Norvig, após o match entre DB e kasparov, o valor das ações da IBM aumentou em 18 bilhões de dólares. (Só uma comparação prosaica: se uma ferrari vermelha maravilhosa custa US$300.000, com este valor é possível comprar 60.000 ferraris vermelhas maravilhosas).

A sexta partida do match foi jogada em Nova York no dia 11 de maio de 1997. Começou as 15h00 e foi a última partida do match que até então estava empatado em 2.5 a 2.5. Kasparov havia ganho o primeiro jogo, perdeu o segundo (abandonando-o) e empatando os jogos 3, 4 e 5, com posições ligeiramente vantajosas para Kasparov nos 3 jogos.

Este jogo atraiu a atenção do mundo, além de ser o último e de ser ganho pelo DB, houve o fato de Kasparov ter desistido no lance 19, após pouco mais de 1 hora de jogo.

Kasparov deixou de ser campeão mundial em por volta de 2000, em uma situação de indefinição, já que havia múltiplos campeões (uns não aceitavam as regras dos outros). Situação semelhante à vivida no box.

Em 1993, a Professional Chess Association (PCA) foi criada por Garry Kasparov, então campeão mundial pela FIDE, e Nigel Short para a organização e marketing da série de partidas pelo título mundial.

Kasparov e Short acusaram a FIDE de falta de profissionalismo e favoritismo. Além disso, os dois recusaram ceder 25% dos prêmios para a FIDE. A FIDE retirou o título de Campeão Mundial de Xadrez de Kasparov e negou a Short o direito de desafiar Kasparov.

Em Outubro de 1993, o duelo entre Kasparov e Short aconteceu no Savoy Theatre de Londres. Kasparov ganhou claramente 12.5-7.5 e tournou-se Campeão Mundial PCA. Como a FIDE expulsara Kasparov e Short, Anatoly Karpov jogou contra Jan Timman pelo título mundial da FIDE, onde Karpov triunfou. Pela primeira vez na história do Xadrez havia dois campeões do mundo, o campeão mundial da FIDE Karpov e o campeão mundial PCA Kasparov.

Em 1995, Kasparov defendeu seu título PCA contra o grande mestre indiano Viswanathan Anand no World Trade Center, num match iniciado em 1 de Setembro de 1995. Kasparov ganhou o match de 2O partidas 10.5 - 7.5. A PCA acabou por deixar de existir por falta de patrocínio, quando a Intel deixou de subvencionar a associação, e quando Kasparov enfrentou Vladimir Kramnik em 2000, em Londres, já era sob a tutela de Braingames.

Kramnik ganhou o match 8.5-6.5, e pela primeira vez em 15 anos Kasparov não era detentor de nenhum título mundial.

A lista oficial de campeões mundiais da FIDE, após Kasparov é

- Garry Kasparov 1985 - 2000 15 anos

- Vladimir Kramnik 2000 - 2005 e 2006-2007 5 anos
- Veselin Topalov 2005 - 2006 1 ano
- Viswanathan Anand 2007 - 2013 7 anos
- Magnus Carlsen 2013 -

A tendência é a de desenvolver softwares para plataformas PC convencionais. Este é o caminho seguido por Deep Jr, a última encarnação do programa.

Garry Kasparov vs Deep Junior foi um match de xadrez realizado no ínicio de 2003 sendo oficialmente o primeiro campeonato mundial homem contra máquina organizado pela federação internacional de xadrez.

O match foi realizado no New York Athletic Club em Nova Iorque nos Estados Unidos entre os dias 26 de Janeiro a 7 de Fevereiro de 2003. A premiação foi de $1,000,000 no total sendo $500,000 para a participação de Kasparov e a metade restante dividindo-se entre os dois baseado no resultado final sendo $300,000 para o vencedor, $200,000 para o perdedor e $250,000 para ambos em caso de empate.

O tempo de jogo foi com 120 minutos para cada jogador no início de jogo, acrescentado mais 60 minutos a partir do movimento 40 e mais 30 minutos a partir do movimento 60.

Eis o resultado:

	J 1	J 2	J 3	J 4	J 5	J 6	Final
Garry Kasparov	1	½	0	½	½	½	3
Deep Junior	0	½	1	½	½	½	3

O rank de Kasparov no momento em que ocorreu esse match era de 2847 pontos sendo o primeiro colocado no ranking mundial da FIDE.

Implementação Para o caso do xadrez, de longe o jogo mais estudado, diversas abordagens são estudadas para criar funções de avaliação.

Note-se que este conceito é prévio ao jogo automatizado, sendo usado por professores de xadrez ensinando a jogadores humanos.

Uma técnica bem fácil é atribuir valor material a cada uma das peças. A tabela mais ou menos aceita é

peça	valor
peão	1
cavalo ou bispo	3
torre	5
dama	9
segurança do rei	0,5
boa estrutura de peões	0,5

Outra técnica ensinada em alguns livros, é marcar com alfinetes pretos as casas do tabuleiro que são acessáveis por peças pretas e fazer o mesmo com alfinetes brancos para as peças brancas. A função de avaliação é a diferença entre alfinetes. Nada impede que os alfinetes sejam ponderados pelo valor maior das casas onde estão espetados. Por exemplo, as casas centrais.

O que interessa aqui não é um valor absoluto de cada posição, mas a informação obtida na comparação entre 2 posições candidatas. A com maior probabilidade de vencer deve ser a escolhida para as boas funções de avaliação.

Uma abordagem possível aqui (extraía da teoria da computação tolerante à falha) é trabalhar com duas ou mais funções de avaliação independentes e heuristicamente escolher qual delas deve determinar o valor de uma determinada posição.

Gerador de movimentos Este é o outro componente importante de um jogador de xadrez. Dada uma situação qualquer, ou seja, um tabuleiro corretamente preenchido, o gerador de movimentos deve devolver todas as possíveis jogadas respostas a este estado. Duas observações importantes: as palavras "corretamente preenchido" acima indicam que o tabuleiro em questão é possível seguidas as regras oficiais do xadrez a partir de uma situação inicial padrão. (Assim, por exemplo, não há que se preocupar com a existência de 2 reis brancos). A segunda observação, é que o gerador de movimentos precisa retornar suas respostas em ordem de utilidade, a fim de que o corte $\alpha - \beta$ possa ser útil.

Construtor da árvore e implementador do $\alpha-\beta$ Este é o componente mais "computer science" do projeto, já que nada tem a ver com IA. Entretanto, tem e muito a ver com desempenho e confiabilidade, fatores fundamentais para o funcionamento adequado de um jogador automático de xadrez. Note-se que a árvore não pode (não cabe) ser guardada em memória, o que introduz a este componente a utilização de um SGDB.

```
atividades práticas: vivo835 - xadrez
vivo050p e 050c - gerador de movimentos de xadrez em python e em C++
vivo253 - passeio do cavalo
```

Índice Remissivo

aplicação prática de grafos: insanidade instantânea, 20

banco de dados e modelo de Date e álgebra relacional, 10
bioinformática, 16

certificação de objetos binários, 22
certificação digital, 4
compressão digital, 38
criptografia, 24

demonstração automática de teoremas, 36
Desenho no monitor, 41
dinheiro virtual: bitcoin, 29
dinheiro virtual: blockchain – bifurcação e disputa, 32
dinheiro virtual: mineração de bitcoins, 31
dinheiro virtual: o problema dos generais bizantinos, 27

espaços de busca imensos: o xadrez, o rubik, ..., 9
estado da arte em edição de textos: LATEX, 8
esteganografia: nem tudo o que reluz é ouro, 22

Google, 42
gps, 35

idiomas do planeta terra e os computadores, 10
inteligência artificial, 4

linguagens e compiladores, 19

pdf: o papel digital, 21
probabilidade e teorema de Bayes, 9
processamento fonético, 11
programando como respiramos, 3
programação dinâmica, 37

qrcode, 17

redes de petri, 12
redes neurais e aprendizado automático, 6

simulação, 13

tomografia computadorizada cerebral, 19

verificação e correção de códigos, 15

xadrez e computador, 42