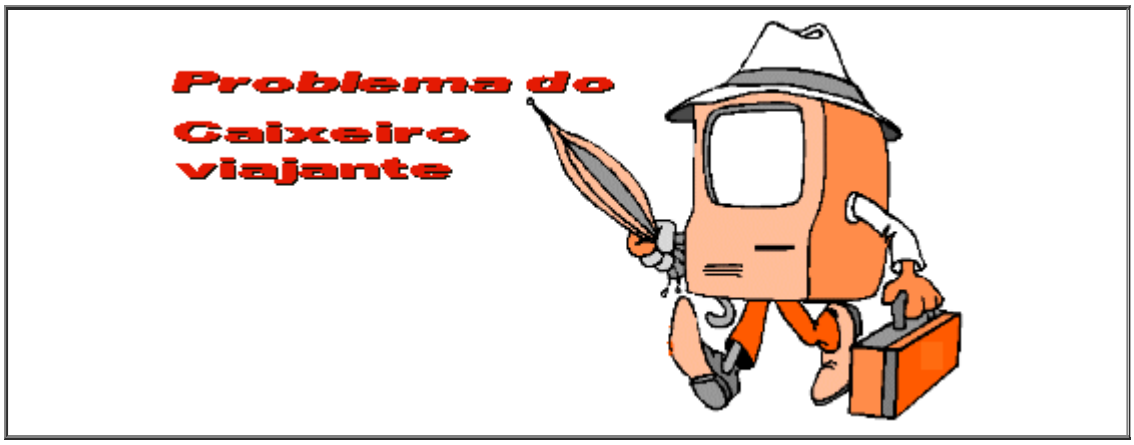



Não se iluda com a aparência de brincadeira deste problema. Ele NÃO é mais uma curiosidade inconsequente para entreter alunos desmotivados. Em verdade, ele é um problema que deve fazer parte da bagagem de todo profissional competente da área matemática. Sua importância resume-se em duas capacidades:

- de modo simples e concreto, exemplifica a enorme velocidade de crescimento da fatorial
- prova-se que muitos problemas combinatórios envolvem tantas alternativas de solução quanto este problema, de modo que ele é uma espécie de "metro" com o qual medimos a complexidade computacional dos problemas combinatórios ocorrendo em engenharia e no trabalho científico.

---

# Formulando o problema do caixeiro:

Suponha que um caixeiro viajante tenha de visitar **n** cidades diferentes, iniciando e encerrando sua viagem na primeira cidade. Suponha, também, que não importa a ordem com que as cidades são visitadas e que de cada uma delas pode-se ir diretamente a qualquer outra.
O problema do caixeiro viajante consiste em descobrir a rota que torna mínima a viagem total.

Exemplificando o caso n = 4:
se tivermos quatro cidades A, B, C e D, uma rota que o caixeiro deve considerar poderia ser: *saia de A e daí vá para B, dessa vá para C, e daí vá para D e então volte a A*. Quais são as outras possibilidades ? É muito fácil ver que existem seis rotas possíveis:

- ABCDA
- ABDCA
- ACBDA
- ACDBA
- ADBCA
- ADCBA

---

# Complexidade computacional do problema do caixeiro:

O problema do caixeiro é um clássico exemplo de **problema de otimização combinatória**. A primeira coisa que podemos pensar para resolver esse tipo de problema é reduzí-lo a um **problema de enumeração**: achamos todas as rotas possíveis e, usando um computador, calculamos o comprimento de cada uma delas e então vemos qual a menor. ( É claro que se acharmos todas as rotas estaremos contando-as, daí podermos dizer que estamos reduzindo o problema de otimização a um de enumeração ).

Para acharmos o número **R( n )** de rotas para o caso de **n** cidades, basta fazer um raciocínio combinatório simples e clássico. Por exemplo, no caso de n = 4 cidades, a primeira e última posição são fixas, de modo que elas não afetam o cálculo; na segunda posição podemos colocar qualquer uma das 3 cidades restantes B, C e D, e uma vez escolhida uma delas, podemos colocar qualquer uma das 2 restantes na terceira posição; na quarta posição não teríamos nenhuma escolha, pois sobrou apenas uma cidade; consequentemente, o número de rotas é 3 x 2 x 1 = 6, resultado que tinhamos obtido antes contando diretamente a lista de rotas acima.
De modo semelhante, para o caso de n cidades, como a primeira é fixa, o leitor não terá nenhuma dificuldade em ver que o número total de escolhas que podemos fazer é (n-1) x (n-2) x ... x 2 x 1. De modo que, usando a notação de fatorial: **R( n ) = ( n - 1 )!**.

Assim que nossa estratêgia reducionista consiste em gerar cada uma dessas R( n ) = ( n - 1 )! rotas, calcular o comprimento total das viagens de cada rota e ver qual delas tem o menor comprimento total. Trabalho fácil para o computador, diria alguém. Bem, talvez não. Vejamos o porquê.

Suponhamos temos um muito veloz computador, capaz de fazer 1 bilhão de adições por segundo. Isso parece uma velocidade imensa, capaz de tudo. Com efeito, no caso de 20 cidades, o computador precisa apenas de 19 adições para dizer qual o comprimento de uma rota e então será capaz de calcular $10^9$ / 19 = 53 milhões de rotas por segundo. Contudo, essa imensa velocidade é um nada frente à imensidão do número 19! de rotas que precisará examinar. Com efeito, acredite se puder, o valor de 19! é 121 645 100 408 832 000 ( ou , aproximadamente, 1.2 x $10^{17}$ em notação científica ). Consequentemente, ele precisará de

$$1.2 \times 10^{17} / ( 53 \text{ milhões } ) = 2.3 \times 10^9 \text{ segundos}$$

para completar sua tarefa, o que equivale a cerca de **73 anos** . O problema é que **a quantidade ( n - 1 )! cresce com uma velocidade alarmante, sendo que muito rapidamente o computador torna-se incapaz de executar o que lhe pedimos.** Constate isso mais claramente na tabela a seguir:

| n | rotas por segundo | ( n - 1 )! | cálculo total |
|---|---|---|---|
| 5 | 250 milhoes | 24 | insignific |
| 10 | 110 milhoes | 362 880 | 0.003 seg |
| 15 | 71 milhoes | 87 bilhoes | 20 min |
| 20 | 53 milhoes | 1.2 x $10^{17}$ | 73 anos |
| 25 | 42 milhoes | 6.2 x $10^{23}$ | 470 milhoes de anos |

Observe que o aumento no valor do n provoca uma muito lenta diminuição na velocidade com que o computador calcula o tempo de cada rota ( ela diminui apenas de um sexto ao n aumentar de 5 para 25 ), mas provoca um imensamente grande aumento no tempo total de cálculo. Em outras palavras: a inviabilidade computacional é devida à presença da fatorial na medida do esforço computacional do método da redução. Com efeito, se essa complexidade fosse expressa em termos de um polinómio em n o nosso computador seria perfeitamente capaz de suportar o

aumento do n. Confira isso na seguinte tabela que corresponde a um esforço computacional polinomial $R( n ) = n^5$:

| n | rotas por segundo | $n^5$ | cálculo total |
|---|---|---|---|
| 5 | 250 milhoes | 3 125 | insignific |
| 10 | 110 milhoes | 100 000 | insignific |
| 15 | 71 milhoes | 759 375 | 0.01 seg |
| 20 | 53 milhoes | 3 200 000 | 0.06 seg |
| 25 | 42 milhoes | 9 765 625 | 0.23 seg |

Então o método reducionista não é prático ( a não ser para o caso de muito poucas cidades ), mas será que não pode-se inventar algum método prático ( por exemplo, envolvendo esforço polinomial na variável número de ) para resolver o problema do caixeiro? Bem, apesar de inúmeros esforços, ainda não foi achado um tal método e começa-se a achar que o mesmo não existe.

A existência ou não de um método polinomial para resolver o problema do caixeiro viajante é um dos grandes problemas em aberto da Matemática na medida em que S. A. COOK ( 1971 ) e R. M. KARP ( 1972 )) mostraram que uma grande quantidade de problemas importantes ( como é o caso de muitos tipos de problemas de otimização combinatória, o caso do problema da decifragem de senhas criptografadas com processos modernos como o DES, etc ) podem ser reduzidos, em tempo polinomial, ao problema do caixeiro.
Consequentemente: se descobrirmos como resolver o problema do caixeiro em tempo polinomial ficaremos sendo capazes de resolver, também em tempo polinomial, uma grande quantidade de outros problemas matemáticos importantes; por outro lado, se um dia alguém provar que é impossível resolver o problema do caixeiro em tempo polinomial no número de cidades, também se terá estabelecido que uma grande quantidade de problemas importantes não tem solução prática.

Costuma-se resumir essas propriedades do problema do caixeiro dizendo que ele pertence à categoria dos problemas **NP - completos**.

---

**EXERCICIO**

Por que o computador, no caso de 21 cidades, precisaria de 20 vezes mais anos do que precisou para resolver o caso de 20 cidades pelo metodo reducionista?

**EXERCICIO**

V. seria capaz de ver relação entre o problema do caixeiro viajante e os seguintes problemas tecnológicos, de enorme importância econômica?

- a minimização do tempo que um robot industrial gasta para soldar a carcaça de um automóvel
- custo em tempo ou combustivel na rota da distribuição diária de um jornal produzido numa grande cidade?
- custo em tempo na rota de abastecimento das várias bases militares envolvidas numa guerra?

**EXERCICIO**

Imagine um computador com velocidade fantástica, à sua escolha. Faça tabela das fatoriais de 10 a 100, pulando de 10 em 10. Para cada uma dessas fatoriais, calcule o tempo que esse computador levaria para resolver o respectivo problema do caixeiro viajante. Conclusão?

**EXERCICIO**

Compare o esforço da resolução do problema do caixeiro viajante com o do cálculo - diretamente, pela definição - de um determinante n x n.

**EXERCICIO**

Faça a análise da complexidade computacional da variante do problema do caixeiro que tem cidade inicial e cidade final distintas.

---

# Exemplo do uso do problema do caixeiro como um "metro" de complexidade computacional.

Uma das grandes tarefas do século XXI será a construção de novos paradigmas computacionais que transcendam a já quase exaurida capacidade dos computadores eletrônicos digitais. Isso nos permitirá resolver problemas científicos e técnicos de tratamento atualmente inacessível. Uma promissora possibilidade de novo paradigma computacional é a dos computadores genéticos.

Os genes dos seres vivos podem ser vistos como computadores processando a informação genética contida nas moléculas de DNA que os formam. Enquanto os computadores eletrônicos processam informação codificada em sequências de 0's e 1's, os genes processam informação ( genética ) codificada em sequências feitas com os quatro nucleotídeos que formam as moléculas de DNA: adenina, citosina, guanina e timina. Os genes podem ser vistos como um computador que trabalha com base=4, explorando as correspondências:

0 - adenina, 1 - citosina, 2 - guanina e 3 - timina.

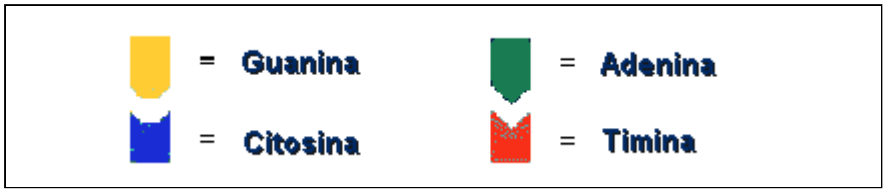
O potencial computacional aí envolvido é enorme:

- um kilograma de DNA armazena mais informação do que a soma de todos os computadores eletrônicos até hoje construídos
- o processamento da informação genética é simultâneo e, então, muito mais rápido do que o processamento sequencial dos computadores eletrônicos; provavelmente, uma mera gotícula de DNA teria maior poder computacional do que o maior computador eletrônico que já foi construído
  ( alguns pesquisadores estimam que se conseguiria velocidades de cálculo até um milhão de vezes maior do que a dos atuais computadores eletrônicos )

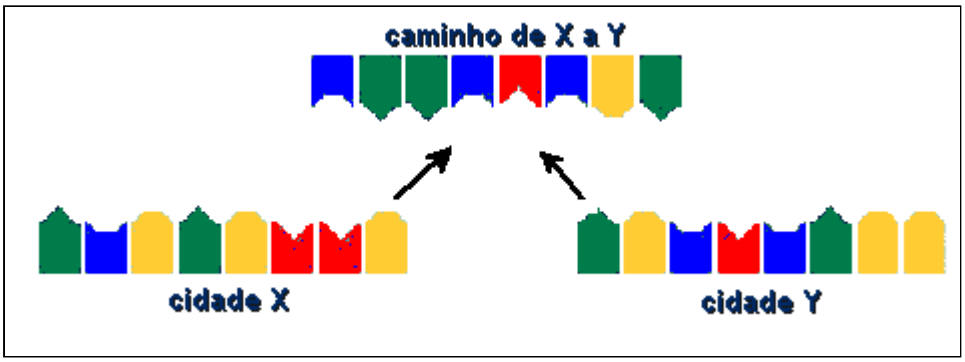
A primeira pessoa conseguindo demonstrar na prática esse potencial foi Leonard Adleman, em 1994. Ele construiu um protótipo de computador genético para resolver o Problema do Caixeiro Viajante, na variante em que se tem sete cidades, quatorze caminhos e cidade inicial distinta da final.

A primeira tarefa de Adleman foi dar uma representação biológica para os dados do problema: as cidades e os caminhos entre cidades. Para tal, ele iniciou associando bijetoramente as cidades

envolvidas à sequências com número fixo de nucleotídeos ( ele usou sequências com 20 nucleotídeos e em nossa ilustração abaixo usaremos, para simplificar, apenas 8 ). Para representar os caminhos entre pares de cidades, ele explorou um fenômeno básico que ocorre com os quatro nucleotídeos. Eles formam pares complementares, atraindo-se Guanina com Citosina e Adenina com Timina, sendo exatamente esses acoplamentos que dão origem ao conhecido formato de hélice dupla da DNA.




Assim, para representar cada "caminho da cidade X até a cidade Y", Adleman usou técnicas laboratoriais para fazer o acoplamento da segunda metade da sequência de de nucleotídeos da cidade X com a primeira metade da sequência de Y e então fabricar a sequência complementar desse acoplamento. Exemplificando, se X=ACGAGTTG e Y=AGCTCAGG então o caminho de X para Y fica: GTTG + AGCT = GTTGAGCT ---> CAACTCGA. Acompanhe na ilustração abaixo e, em particular, observe e verifique que as atrações de pares complementares é quem justifica usarmos a sequência CAACTCGA como representante do caminho da cidade X para a cidade Y:




As sequências das cidades X e Y, por força da atração complementar, juntam-se - como chaves em fechaduras - com a sequência do caminho de X para Y formando uma "sequência hibridizante":


Conseguida a representação genética dos dados do problema do caixeiro, veio a segunda e mais difícil tarefa de Adleman: fazer com que a informação associada ao problema fosse processada geneticamente e assim obter o caminho ótimo desejado. Para conseguir isso, ele precisou usar dezenas de procedimentos laboratoriais que consumiram uma semana. Esses iniciaram com a ligação das sequências hibridizantes ( de modo a obter rotas passando por todas as cidades ), continuaram com a separação das rotas legítimas ( as que iniciavam na primeira cidade e terminavam na última, e, ademais, passavam exatamente uma vez por cada outra cidade ), e terminaram com a determinação da rota minimal.


O trabalho de Adleman foi apenas um primeiro passo, sendo que ainda existem muitos obstáculos a vencer antes de se conseguir tornar a computação genética prática no trabalho científico. Por exemplo, para poder garantir que as DNA hibridizantes seriam capazes de gerar todos os possíveis caminhos, ele precisou usar cerca de um trilhão de cópias da sequência de cada cidade e da sequência de cada caminho entre pares de cidades. Isso faz com que, para resolver o problema do caixeiro com 200 cidades e pelo método de Adleman, seja preciso uma quantidade de DNA pesando mais do que nosso planeta. Outras dificuldades importantes são a demora e os erros dos procedimentos laboratoriais envolvidos. De qualquer modo, só o tempo dirá se os cientistas e matemáticos, já trabalhando na computação genética, conseguirão remover esses obstáculos.

---

versão: 29-junho-2 000
localize esta página em: http://www.mat.ufrgs.br/~portosil/caixeiro.html