# Projeto e Análise de Algoritmos

Prof. Dr. Ednaldo B. Pizzolato

www.dc.ufscar.br/~ednaldo

ednaldo@dc.ufscar.br

# Análise de Algoritmos

Framework

# Eficiencia de tempo e de espaço

- Complexidade temporal
- Complexidade espacial

- Evolução dos computadores, a questão espacial foi bem atenuada.

# Tamanho da entrada

- Algoritmos demoram mais quando têm que trabalhar com maiores quantidades de dados.
- Como medir o tamanho da entrada?
  - Matriz n x n
    - Ordem n
    - Tamanho N = n x n
  - Verificação ortográfica
    - Número de palavras?
    - Número de letras?

# Tamanho da entrada

- Conhecer um pouco melhor o problema permitirá avaliar o tamanho da entrada
  - Número primo

# Tamanho da entrada

- Conhecer um pouco melhor o problema permitirá avaliar o tamanho da entrada
  - Número primo

    O que importa neste caso é a magnitude do número.

    Uma possível medida é o número de bits em sua representação binária.

# Comparações

- Benchmark (prática)
  - Quando se compara 2 ou mais programas projetados para resolver a mesma tarefa, normalmente um conjunto padrão de dados é reservado para avaliar o desempenho da solução. Isso significa que este conjunto deve ser representativo do universo de dados aos quais o programa estará exposto e pode servir como referência de teste para outras soluções.
- **Análise teórica**
- Exemplos : reconhecimento de face, fala...

# Comparações

- Benchmark (tempo)
- Análise teórica - Operações básicas
  - Uma possibilidade: contar quantas vezes cada operação básica é executada → muito detalhista
  - Normalmente os maiores custos estão nos loops mais internos

# Regra 90-10

- Muitos programas executam um pequeno conjunto de instruções muitas vezes (90% do tempo de execução em 10% do código).

# O que é complexidade?

A complexidade de um algoritmo consiste na quantidade de "trabalho" necessária para a sua execução, expressa em função das operações fundamentais, as quais variam de acordo com o algoritmo, e em função do volume de dados.

# O que é complexidade?

- Ou seja, normalmente, programas demoram mais para executar de acordo com sua estruturação de instruções e na medida em que se aumenta a quantidade de dados de entrada.
- Esta “demora” pode ser linearmente proporcional, quadrática...
- Em alguns casos, o tempo de execução não depende somente da quantidade de dados, mas sim de sua forma.

# Para que?

Para permitir a comparação teórica de soluções diferentes para o mesmo problema e identificar as melhores soluções (eficiência).

# Unidades para medir tempo de execução

- $T(n) \approx c_{op}.C(n)$
- $c_{op}$ é o custo da operação op
- C(n) é o número de vezes que ela é repetida
- T(n) é a estimativa do tempo de execução

# Ordem de Crescimento

- A diferença em tempo de execução para entradas pequenas não é o que irá distinguir um algoritmo eficiente de um ineficiente.

| entrada | Log n | N | N log N | $N^2$ | $N^3$ | $2^n$ | N! |
|---|---|---|---|---|---|---|---|
| 10 | 3.3 | 10 | 33 | $10^2$ | $10^3$ | $10^3$ | $3.6\ 10^6$ |
| $10^2$ | 6.6 | $10^2$ | 660 | $10^4$ | $10^6$ | $1.3\ 10^{30}$ | $9.3\ 10^{157}$ |
| $10^3$ | 10 | $10^3$ | $10^4$ | $10^6$ | $10^9$ | | |
| $10^4$ | 13 | $10^4$ | $1.3\ 10^5$ | $10^8$ | $10^2$ | | |

# Ordem de Crescimento

- Algoritmos que requerem um número exponencial de operações são práticos para resolver apenas problemas de pequeno porte.

# Perspectivas de análise

- Pior caso
- Caso médio
- Melhor caso

# Perspectivas de análise

**Pior caso**

Este método é normalmente representado por **O ( ).** Por exemplo, se dissermos que um determinado algoritmo é representado por g(x) e a sua complexidade no pior caso é n, será representada por g(x) = **O**(n).

Consiste, basicamente, em assumir o pior dos casos que podem acontecer, sendo muito usado e sendo normalmente o mais fácil de se determinar.

# Perspectivas de análise

**Caso médio**

Representa-se por θ().

Este método é, dentre os três, o mais difícil de se determinar pois necessita de análise estatística (muitos testes). No entanto, é muito usado pois é também o que representa mais corretamente a complexidade do algoritmo.

# Perspectivas de análise

**Melhor caso**

Representa-se por Ω ( )

Método que consiste em assumir que vai acontecer o melhor. Pouco usado. Tem aplicação em poucos casos.

# Limite Superior

Seja dado um problema, por exemplo, multiplicação de duas matrizes quadradas de ordem n (n*n). Conhecemos um algoritmo para resolver este problema(pelo método trivial) de complexidade $O(n^3)$. Sabemos assim que a complexidade deste problema não deve superar $O(n^3)$, uma vez que existe um algoritmo desta complexidade que o resolve. Um limite superior (upper bound) deste problema é $O(n^3)$. O limite superior de um algoritmo pode mudar se alguém descobrir um algoritmo melhor. Isso de fato aconteceu com o algoritmo de Strassen que é de $O(n^{\log 7})$. Assim o limite superior do problema de multiplicação de matrizes passou a ser $O(n^{\log 7}) \approx O(n^{2.807})$. Outros pesquisadores melhoraram ainda mais este resultado. Atualmente, o melhor resultado é o de Coppersmith e Winograd:

$$O(n^{2.376}).$$

# Limite Superior

O limite superior de um algoritmo é parecido com o record mundial de uma modalidade de atletismo. Ele é estabelecido pelo melhor atleta (algoritmo) do momento. Assim como o record mundial o limite superior pode ser melhorado por um algoritmo (atleta) mais veloz.

# Limite Inferior

Às vezes é possível demonstrar que para um dado problema, qualquer que seja o algoritmo a ser usado o problema requer pelo menos um certo número de operações. Essa complexidade é chamada Limite inferior (Lower Bound). Veja que o limite inferior dependo do problema mas não do particular algoritmo. Usamos a letra Ω em lugar de O para denotar um limite inferior.

Para o problema de multiplicação de matrizes de ordem *n,* apenas para ler os elementos das duas matrizes de entrada leva $O(n^2)$. Assim uma cota inferior trivial é $\Omega(n^2)$.

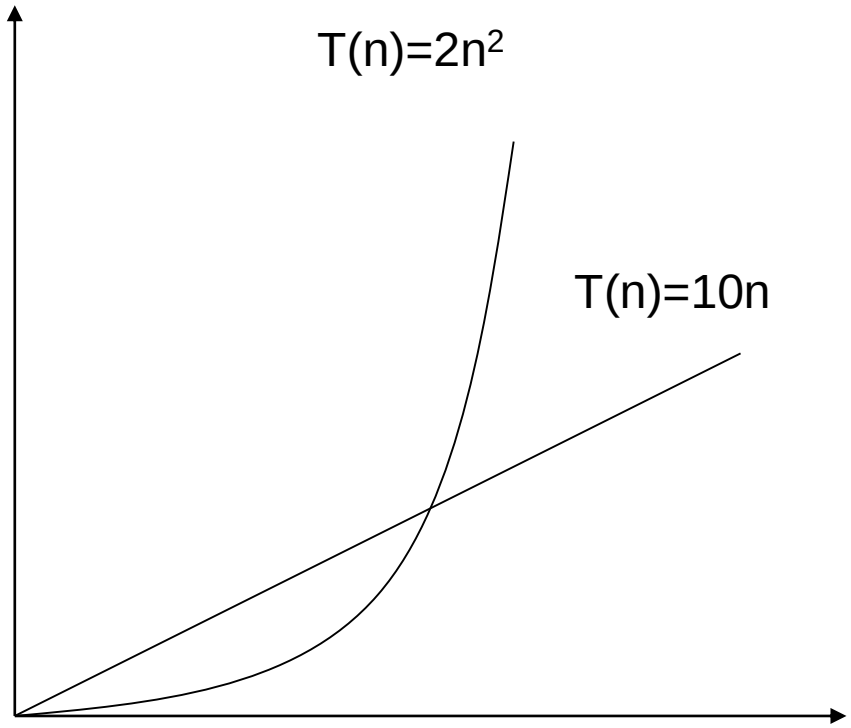
# Limite Inferior

Na analogia anterior, um limite inferior de uma modalidade de atletismo não dependeria mais do atleta. Seria algum tempo mínimo que a modalidade exige, qualquer que seja o atleta. Um limite inferior trivial para os 100 metros seria o tempo que a velocidade da luz leva a percorrer 100 metros no vácuo.

Se um algoritmo tem uma complexidade que é igual ao limite inferior do problema então o algoritmo é ótimo.

O algoritmo de CopperSmith e Winograd é de $O(n^{2.376})$ mas o limite inferior é de $\Omega(n^2)$. Portante não é ótimo. É possível que este limite superior possa ainda ser melhorado.

# Comparando




| Tempo | Tam. A | Tam. B |
|---|---|---|
| 1s | 10 | 22 |
| 10s | 100 | 70 |
| 100s | 1.000 | 223 |
| 1.000s | 10.000 | 707 |

# THE END