Complexidade de Algoritmos

Edson Prestes

Complexidade de Algoritmos

- Um problema pode ser resolvido através de **diversos algoritmos**;

O fato de um algoritmo resolver um dado problema não significa que seja aceitável na prática.

ordem	Método de Cramer	Método de Gauss
2	$22\mu s$	$50\mu s$
3	$102\mu s$	$159\mu s$
4	$456\mu s$	$353\mu s$
5	$2.35ms$	$666\mu s$
10	$1.19min$	$4.95ms$
20	15255 séculos	$38.63ms$

Complexidade de Algoritmos

- Na maioria das vezes, a escolha de um algoritmo é feita através de critérios subjetivos como

  1) facilidade de compreensão, codificação e depuração;

  2) eficiencia na utilização dos recursos do computador e rapidez.

A análise de algoritmo fornece uma **medida objetiva de desempenho proporcional ao tempo de execução** do algoritmo.

Complexidade de Algoritmos

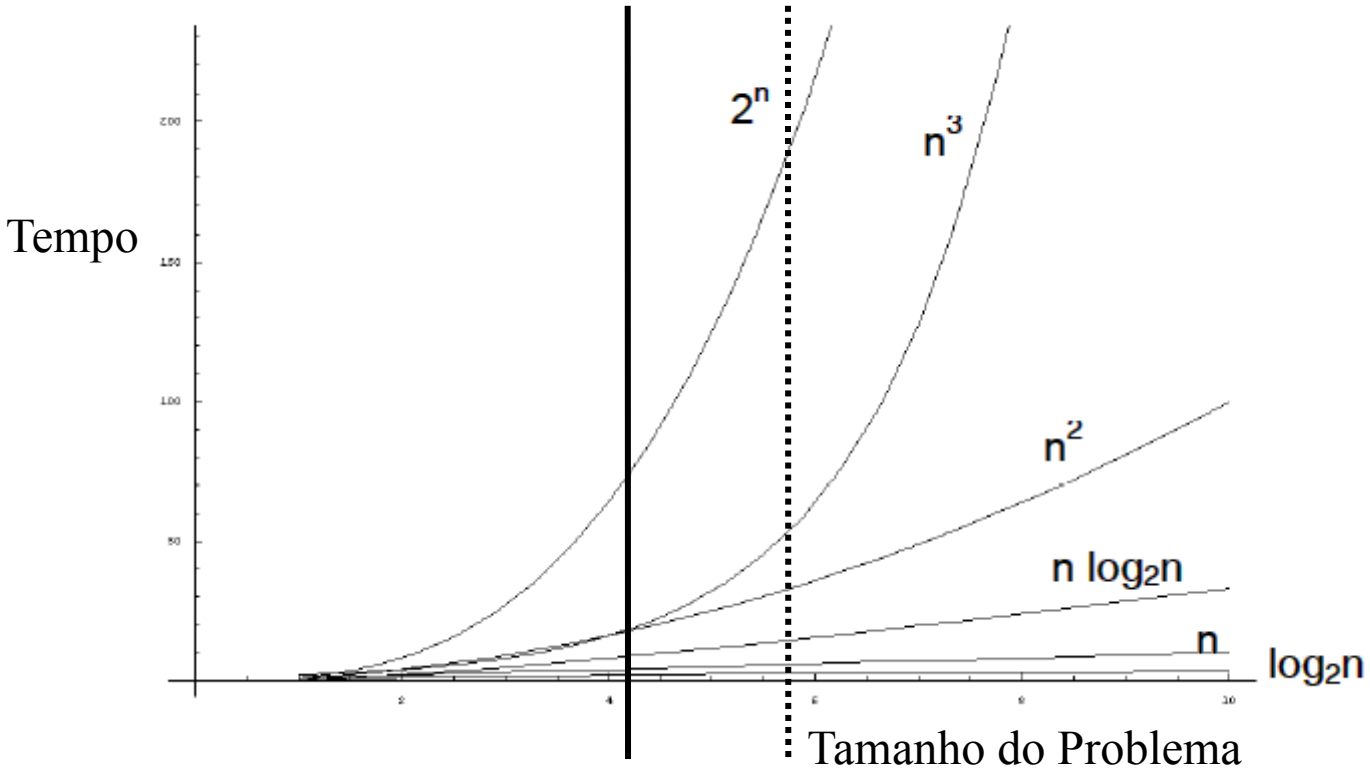
Por que analisar a eficiência de algoritmos se os computadores estão cada dia mais rápidos ?

Complexidade de Algoritmos

complexidade de tempo	máquina lenta	máquina rápida ($10x$)
$log_2 n$	x_0	x_0^{10}
n	x_1	$10x_1$
$n \log_2 n$	x_2	$10x_2$ (p/ x_2 grande)
n^2	x_3	$3.16x_3$
n^3	x_4	$2.15x_4$
2^n	x_5	$x_5 + 3.3$
3^n	x_6	$x_6 + 2.096$

Complexidade do Algoritmo x Tamanho máximo de problema resolvível

Complexidade de Algoritmos



Complexidade de Algoritmos

Tamanho da entrada

complexidade	10	100	10^3	10^4	10^5	10^6
$\log_2 n$	3	6	9	13	16	19
n	10	100	1000	10^4	10^5	10^6
$n \log_2 n$	30	664	9965	10^5	10^6	10^7
n^2	100	10^4	10^6	10^8	10^{10}	10^{12}
n^3	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
2^n	10^3	10^{30}	10^{300}	10^{300}	10^{3000}	10^{300000}

1 ano = $365 \times 24 \times 60 \times 60 \approx 3 \times 10^7$ segundos

1 século $\approx 3 \times 10^9$ segundos

1 milénio $\approx 3 \times 10^{10}$ segundos

Complexidade de Algoritmos

- A eficiência de um algoritmo pode ser medida através de seu tempo de execução.

É a melhor medida ???

O tempo de execução não depende somente do algoritmo, mas do conjunto de instruções do computador, a qualidade do compilador, e a habilidade do programador.

Complexidade de Algoritmos

O tempo de execução de um algoritmo para uma determinada entrada pode ser medido pelo número de operações primitivas que ele executa.

Como esta medida fornece um nível de detalhamento grande convém adotar medidas de **tempo assintótica**.

Complexidade de Algoritmos

Medidas de Complexidade

- Complexidade é também chamada **esforço requerido** ou **quantidade de trabalho**.
  - **Complexidade no pior caso :** Considera-se a instância que faz o algoritmo funcionar mais lentamente;
  - **Complexidade média :** Considera-se todas as possíveis instâncias e mede-se o tempo médio.

Complexidade de Algoritmos

Medidas de Complexidade

- A complexidade pode ser calculada através do:
  - **Tempo de execução do algoritmo determinado pelas instruções executadas :**quanto "tempo" é necessário para computar o resultado para uma instância do problema de tamanho *n;*
  - **Espaço de memória utilizado pelo algoritmo** : quanto "espaço de memória/disco" é preciso para armazenar a(s) estrutura(s) utilizada(s) pelo algoritmo.

- O esforço realizado por um algoritmo é calculado a partir da quantidade de vezes que **a operação fundamental é executada**.
  - –Para um algoritmo de ordenação, uma operação fundamental é a comparação entre elementos quando à ordem.

Complexidade de Algoritmos

Comparação entre Complexidades

- A complexidade exata possui muitos detalhes
- A escolha de um algoritmo é feita através de sua taxa de crescimento
- Esta taxa é representada através de cotas que são funções mais simples.
- A ordem de crescimento do tempo de execução de um algoritmo fornece uma caracterização simples de eficiência do algoritmo.

Complexidade de Algoritmos

Comparação entre Complexidades

- Imagine um algoritmo com complexidade

$$\mathbf{an^2+bn+c}$$

desprezamos os termos de baixa ordem

ignoramos o coeficiente constante

Logo o tempo de execução do algoritmo tem cota igual a n^2, ou seja, $O(n^2)$.

Complexidade de Algoritmos

Comparação entre Complexidades

- $O(1)$: constante – mais rápido, impossível
- $O(\log \log n)$: super-rápido
- $O(\log n)$: logarítmico – muito bom
- $O(n)$: linear – é o melhor que se pode esperar se algo não pode ser determinado sem examinar toda a entrada
- $O(n \log n)$: limite de muitos problemas práticos, ex.: ordenar uma coleção de números
- $O(n^2)$: quadrático
- $O(n^k)$: polinomial – ok para n pequeno
- $O(k^n)$, $O(n!)$, $O(n^n)$: exponencial – evite!

Complexidade de Algoritmos

Comparação entre Complexidades

A complexidade por ser vista como uma **propriedade do problema.**

Logo, é possível obter uma medida independente do tratamento dado ao problema ou do caminho percorrido na busca da solução, portanto independente do algoritmo.

Complexidade de Algoritmos

Comparação entre Complexidades

- Considere dois algoritmos A e B com tempo de execução $O(n^2)$ e $O(n^3)$, respectivamente.

Qual deles é o mais eficiente ?

Complexidade de Algoritmos

Comparação entre Complexidades

- Considere dois programas A e B com tempos de execução $100n^2$ e $5n^3$, respectivamente, qual é o mais eficiente ?

Se considerarmos um conjunto de dados de tamanho $n<20$, o programa B será mais eficiente que o programa A.

Entretanto, se o conjunto de dados é **grande**, a diferença entre os dois programas se torna bastante significativa e o programa A é preferido.

Complexidade de Algoritmos

Comparação entre Complexidades

- Considere dois computadores :

  C_1 que executa 10^7 instruções por segundo (10 milhões);

  C_2 que executa 10^9 instruções por segundo (1 bilhão).

- Considere dois algoritmos de ordenação:
  - A - linguagem de máquina para **A** cujo código exige $\mathbf{2n^2}$ instruções para ordenar n números;
  - B - linguagem de alto nível para B cujo código exige **50nlog n** instruções.

- Quanto tempo C_1 e C_2 gastam para ordenar um milhão de números usando os algoritmos A e B ?

Complexidade de Algoritmos

Comparação entre Complexidades

Algoritmo	Comp. C_1	Comp. C_2
Alg A	$\frac{2.(10^6)^2 \text{instruções}}{10^7 \text{instruções/s}} = \mathbf{2.10^5 s}$	$\frac{2.(10^6)^2 \text{instruções}}{10^9 \text{instruções/s}} = \mathbf{2.10^3 s}$
Alg B	$\frac{50.10^6 \log 10^6 \text{instruções}}{10^7 \text{instruções/s}} = \mathbf{30 s}$	$\frac{50.10^6 \log 10^6 \text{instruções}}{10^9 \text{instruções/s}} = \mathbf{0.3 s}$

Com 10 milhões de números o algoritmo A demoraria 2,3 dias no computador C_2, enquanto que o algoritmo B levaria apenas 20 minutos.

Analisar a complexidade de um algoritmo é utilizar adequadamente os recursos disponíveis!!!!

Complexidade de Algoritmos

Comparação entre Complexidades

- Considere dois algoritmos A e B com complexidades $8n^2$ e n^3, respectivamente, qual é o mais eficiente ? Qual é o maior valor de n, para o qual o algoritmo B é mais eficiente que o algoritmo A ?

Os algoritmos têm igual desempenho quando **n=8**.

Até **n=7**, B é mais eficiente que A.

Complexidade de Algoritmos

Comparação entre Complexidades

- Um algoritmo tem complexidade $2n^2$. Num certo computador **A**, em um tempo *t*, o algoritmo resolve um problema de tamanho 25. Imagine agora que você tem disponível um computador **B** 100 vezes mais rápido. Qual é o tamanho máximo do problema que o mesmo algoritmo resolve no mesmo tempo *t*?

No computador A, a quantidade de instruções executadas é $2.(25)^2$

No computador B, a quantidade de instruções executadas é $100.2.(25)^2$

O tamanho do problema resolvido em B é

$2n^2 = 100.2.(25)^2$ ➔ **n = 250.**

Complexidade de Algoritmos

Comparação entre Complexidades

Problemas NP-Completos

- Não são conhecidos algoritmos eficientes que resolvam problemas NP-Completos.
- Eles possuem a característica de que se um deles puder ser resolvido de forma eficiente então todos os problemas desta classe terão solução eficiente
- Vários problemas NP-Completos são, a principio, semelhantes a problemas que têm algoritmos eficientes.
  - Ciclo Euleriano x Ciclo Hamiltoniano
  - Satisfabilidade 2-CNF x Satisfabilidade 3-CNF