Complexidade Algoritmos

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

- A escolha de um algoritmo na maioria das vezes é feita através de critérios subjetivos como
  - facilidade de compreensão, codificação e depuração;
  - 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

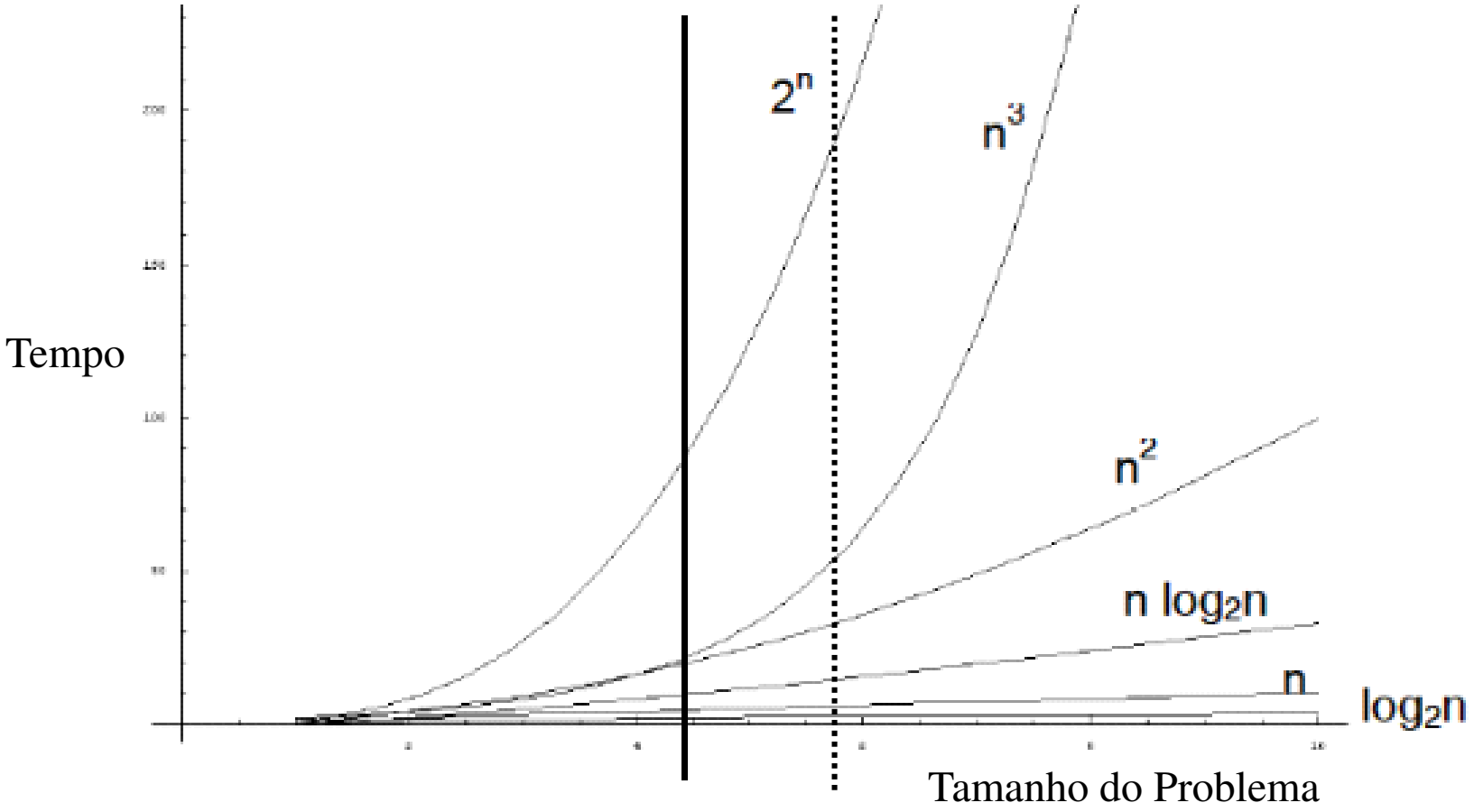
- Por quê analisar a eficiência de algoritmos se os computadores estão cada dia mais rápidos?

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

	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.**

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.

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.

Comparação entre Complexidades

- A complexidade exata possui muitos detalhes, então a escolha de um algoritmo é feita através de sua taxa de crescimento.
- A 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.

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!

Comparação entre Complexidades

- Imagine um algoritmo com complexidade:

$$an^2 + bn + c$$

- Desprezamos os termos de baixa ordem
- Ignoramos o coeficiente constante
- Logo, o tempo de execução do algoritmo tem cota igual a n^2, $O(n^2)$.

Comparação entre Complexidades

- A complexidade por ser vista como uma **propriedade do problema**, o que significa dar uma medida independente do tratamento do problema, independente do caminho percorrido na busca da solução, portanto independente do algoritmo.

- Será?

Comparação entre Complexidades

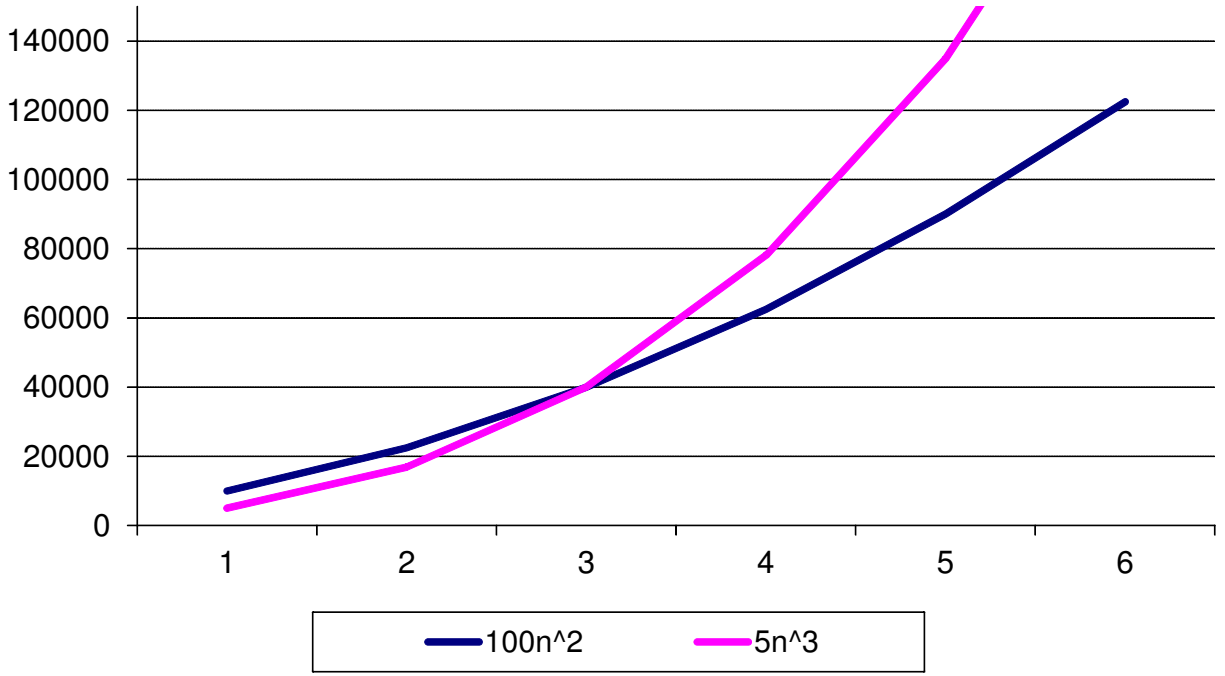
- Considere dois algoritmos A e B com tempo de execução $O(n^2)$ e $O(n^3)$, respectivamente.
- Qual deles é o mais eficiente?

Comparação entre Complexidades

- Considere dois programas A e B com tempos de execução $100n^2$ milisegundos, e $5n^3$ milisegundos, 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.

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?



Comparação entre Complexidades

- Considere dois computadores:

  CompA que executa 10^9 instruções por segundo;

  CompB que executa 10^7 instruções por segundo.

- Considere que dois algoritmos de ordenação:
  - *Prog1*: linguagem de máquina para *CompA* cujo código exige $\mathbf{2n^2}$ instruções para ordenar n números;
  - *Prog2*: linguagem de alto nível para *CompB* cujo código exige $\mathbf{50n \log_2 n}$ instruções.

- Quanto tempo *CompA* e *CompB* demoram para ordenar um milhão de números - 1.000.000 ?

Comparação entre Complexidades

- Para ordenar um milhão de números (10^6)...
  - *CompA* demora:

$$\frac{2(10^6)^2 \text{instruções}}{10^9 \text{instruções/segundo}} = 2000 \text{segundos}$$

  - *CompB* demora:

$$\frac{50(10^6)\log_2 10^6 \text{instruções}}{10^7 \text{instruções/segundo}} = 100 \text{segundos}$$