Universidade Federal da Paraíba

Centro de Informática

Departamento de Informática

Estrutura de Dados

Introdução a Análise de Algoritmos

- Tiago Maritan
- tiago@ci.ufpb.br

Como escolher um algoritmo?

- Tempo de Execução
  - Um algoritmo que realiza uma tarefa em 10h é melhor que outro que realiza em 10 dias

- Quantidade de Memória necessária
  - Um algoritmo que usa 1MB de RAM é melhor que outro que usa 1GB

Tempo de Execução

- Medir o tempo gasto por um algoritmo
  - Não é uma boa opção
  - Depende do compilador
    - Pode preferir algumas construções ou otimizar melhor
  - Depende do hardware
    - GPU vs. CPU, desktop vs. smartphone

- **Alternativa mais indicada: Analisar o número de vezes que as operações são executadas**

Exemplo: Tempo de Execução

- Achar o máximo de um vetor

```
int vmax(int *vec, int n) {          -
        int i;                       -
        int max = vec[0];            1
        for(i = 1; i < n; i++) {     n-1
                if(vec[i] > max) {   n-1
                        max = vec[i];  A < n-1
                }                    n-1
        }                            n-1
        return max;                  1
}
```

- Complexidade: **f(n) = n-1**
- Bom algoritmo

Análise do tempo de processamento

- Análise de complexidade feita em função de **n**
  - **n** indica o tamanho da entrada
    - Ex: nº de elementos no vetor
    - Ex: nº de linhas de uma matriz
    - Ex: nº de vértices num grafo

- Diferentes valores de entradas podem ter custo diferente
  - Melhor caso
  - Pior caso
  - Caso médio

Notação Assintótica

Eficiência Assintótica

- Para valores suficientemente pequenos de **n**, *a* **maioria dos algoritmos custa pouco para ser executado, mesmo os ineficientes.**
  - Escolha de um algoritmo não é um problema crítico
- Mas para valores grandes, a escolha do algoritmo pode influenciar muito...
- **Eficiência assintótica**: Estuda o crescimento do tempo de execução para grandes valores de **n**

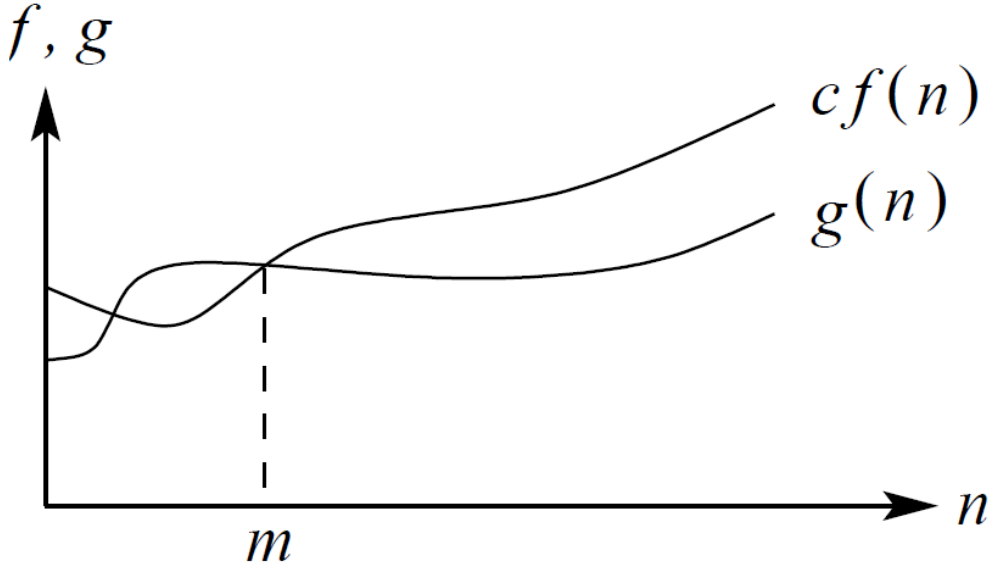
Notação O (ou Big-O)

- **Notação O**: Geralmente utilizada para representar a complexidade computacional de um algoritmo
  - Complexidade ⇔ Ordem de crescimento do tempo de execução

- Podemos comparar algoritmos usando a **Notação O**
  - Um algoritmo **O(n) é melhor do que um $O(n^2)$**
  - Algoritmos com a mesma complexidade assintótica são equivalentes

Notação O (ou Big-O)

- Uma função f(*n*) **domina assintoticamente outra função** g(*n*), se existem duas constantes positivas *c* e *m* tais que **$|g(n)| \leq c|f(n)|$,** para $n \geq m$.
  - Tradução grosseira: a partir de um certo ponto *m*, ***cf*(*n*) cresce mais rapidamente do que g(*n*)**.



Notação O (ou Big-O)

- Definimos **g(*n*) = O(f(*n*))** *se* f(*n*) *domina assintoticamente* g(*n*)
  - Lê se g(*n*) é da ordem no máximo f(*n*)
  - (...Ou f(*n*) cresce mais rápido do que g(*n*))

- Se o tempo de execução de um programa é

  T(*n*) = O(*n*²)... então **T(*n*) ≤ *cn*²**

- Exemplo: Se **T(n) = (n+1)²** ... então **O(n²)**
  - (Pois **(*n*+1)² ≤ 4*n*²**, *para n* ≥ 1 e c = 4)

Notação O - Exemplos

- Maior polinômio T(n) geralmente determina o valor da complexidade assintótica (Big-O)

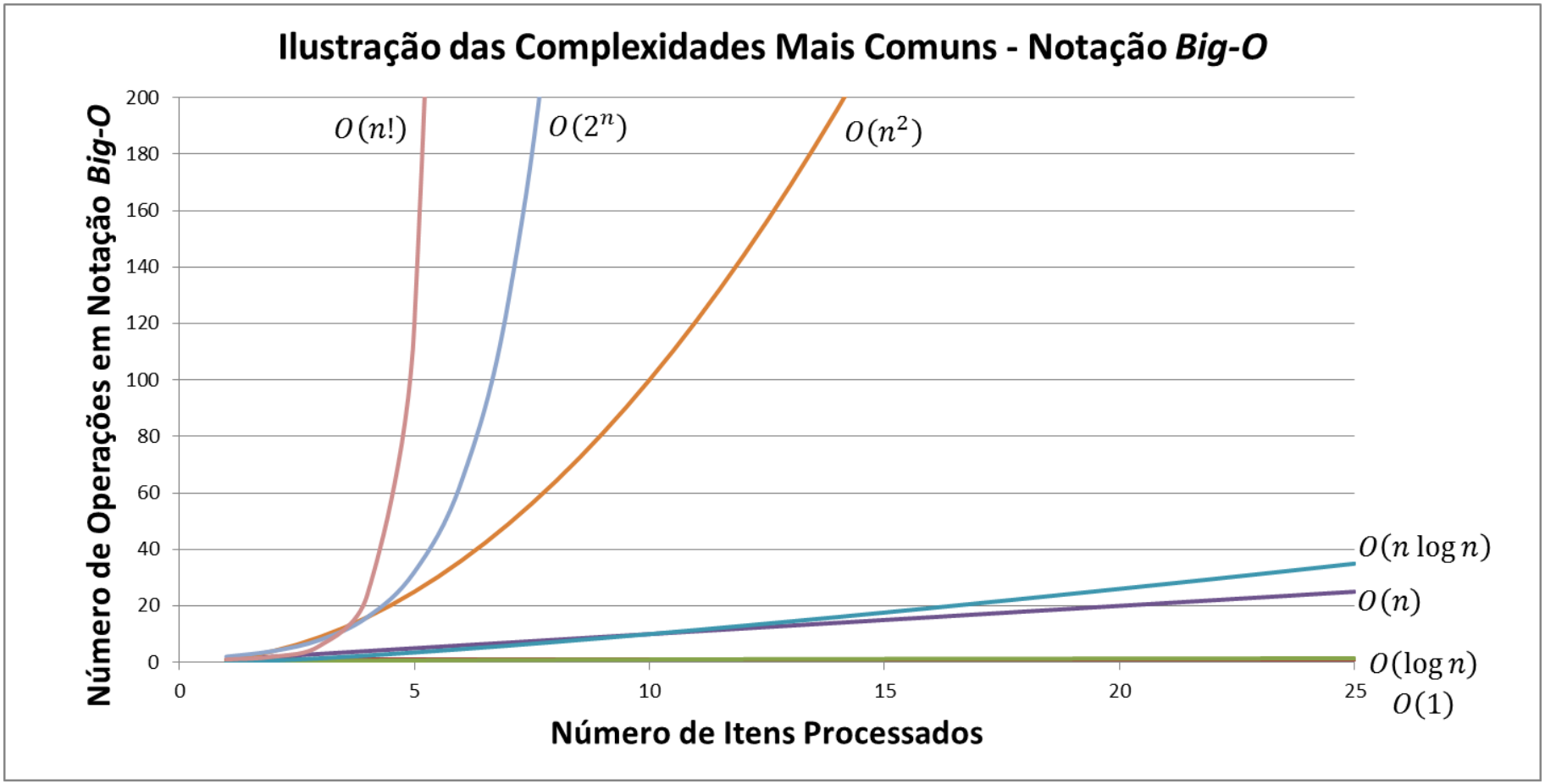
- Ex1: Se **$T(n) = (n+1)^2$** ... então **$O(n^2)$**
  - (Pois **$(n+1)^2 \leq 4n^2$**, *para* $n \geq 1$ e c = 4)

- Ex2: Se **$T(n) = 3n^3 + 2n^2 + n$** ... então **$O(n^3)$**
  - (Pois, $T(n) = 3n^3 + 2n^2 + n \leq 6n^3$, para $n \geq 0$)

- Ex3: Se **$T(n) = \log_5(n)$** ... então **$O(\log(n))$**

- Ex4: Se **$T(n) = 2^{n+1}$** ... então **$O(2^n)$.**
  - (Pois, $T(n) = 2^{n+1} \leq 2.2^n$, para $n \geq 0$)

Notação O - Complexidades mais comuns



Comparação das Classes de Complexidade

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s
n^5	0,1 s	3,2 s	24,3 s	1,7 min	5,2 min	13 min
2^n	0,001 s	1 s	17,9 min	12,7 dias	35,7 anos	366 séc.
3^n	0,059 s	58 min	6,5 anos	3855 séc.	10^8 séc.	10^{13} séc.

Tempo de Execução vs Tamanho da Entrada

Função de custo de tempo	Computador atual	Computador 100 vezes mais rápido	Computador 1.000 vezes mais rápido
n	t_1	$100\ t_1$	$1000\ t_1$
n^2	t_2	$10\ t_2$	$31,6\ t_2$
n^3	t_3	$4,6\ t_3$	$10\ t_3$
2^n	t_4	$t_4 + 6,6$	$t_4 + 10$

Análise de Algoritmos

Análise de Algoritmos

- Determinar o tempo de execução de um algoritmo pode ser complexo

- Determinar a complexidade assintótica, sem preocupação com as constantes envolvidas, pode ser uma tarefa mais simples

Análise de Algoritmos

- Instruções simples (atribuição, comparação, operação aritmética, acesso a memória): **O(1)**
- Sequência de instruções:
  - Maior tempo de execução das instruções;
- Estrutura de decisão (condicional):
  - Tempo pra testar a condição: **O(1) +**
  - Tempo de execução das instruções dentro do condicional;
- Estrutura de Repetição:
  - Tempo pra testar a condição de parada: O(1) +
  - Tempo de execução dos comandos * nº de iterações.

Exemplo 1:

```
-  int soma_acumulada(int n) {
-      int i;
1      int acumulador = 0;
n      for(i = 0; i < n; i++) {
n          acumulador += i;
-      }
1      return acumulador;
-  }
```

- Qual é a ordem de complexidade assintótica?

O(n)

Exemplo 2:

```
          -  void exemplo(int n)
          -  {
          -      int i, j;
          1      int a = 0;
          n      for(i = 0; i < n; i++)
n(n+1)/2            for(j = n; j > i; j--)
n(n+1)/2                a += i + j;
          1      exemplo1(n);
          -  }
```

- Qual é a ordem de complexidade assintótica?

$O(n^2)$

Exemplo 3:

```
-     // A, B e C sao vetores globais
-     void e1(int n) {
-         int i, j, k;
n         for(i = 0; i < n; i++)
n*n           for(j = 0; j < n; j++) {
n*n               C[i][j] = 0;
n*n*n             for(k = n-1; k >= 0; k--) {
n*n*n                 C[i][j] = C[i][j] +
                          A[i][k] * B[k][j];
-             }
-         }
-     }
```

O que faz essa função? Qual sua complexidade assintótica?

$O(n^3)$

Exemplo 4:

```
            -  void ordena(int *V, int n) {
            -     int i, j, min, x;
          n-1     for(i = 0; i < n - 1; i++) {
          n-1        min = i;
    n(n-1)/2         for(j = i + 1; j < n; j++)
    n(n-1)/2            if(V[j] < V[min])
A < n(n-1)/2               min = j;
            -        /* troca A[min] e A[i]: */
          n-1        x = V[min];
          n-1        V[min] = V[i];
          n-1        V[i] = x;
            -     }
            -  }
```

Qual a complexidade assintótica do número de comparações?

$O(n^2)$

Universidade Federal da Paraíba

Centro de Informática

Departamento de Informática

Estrutura de Dados

Introdução a Análise de Algoritmos

- Tiago Maritan
- tiago@ci.ufpb.br