

# Teoria dos Grafos

## Aula 7

### Aula passada

- Implementação BFS
- DFS, implementação
- Complexidade
- Aplicações

### Aula de hoje

- Classe de funções e notação  $O, \Omega, \Theta$
- Propriedades da notação
- Funções usuais
- Tempo de execução

# Comparando Quantidades



## Como comparar quantidades?

- sem considerar detalhes, comparação aproximada

## Ordem de grandeza

- Ordem de grandeza é a parte interia de  $\log_{10} x$ , onde x é a quantidade
- Muito usada na academia (físicos, etc)

# Ordem de Grandeza

- Número de pessoas no planeta?  $\sim 10^{10}$
- Número de páginas web?  $\sim 10^{11}$
- PIB dos EUA?  $\sim 10^{13}$  dólares
- Peso de uma célula humana?  $\sim 10^{-12}$  Kg
- Massa de Saturno é duas ordens de grandeza maior do que a Terra

**Escala poderosa (potência de 10)**

- Fácil comparação entre quantidades (grandes e pequenas)

# Comparando Crescimento



## Como comparar crescimento?

- sem considerar detalhes, comparação aproximada

## Classe de funções

- Notação  $O$ ,  $\Omega$ ,  $\Theta$
- Muito usada na Computação (e algumas outras áreas)

# Comparando Crescimento



- Idéia: capturar o quanto rápido (ou devagar) uma quantidade cresce

**Tipos de crescimento?**

- Linear, quadrático, logarítmico, exponencial, polinomial, etc.

**Como formalizar este conceito?**

# Comparando Crescimento

- Seja  $f(x) = x^2 - x$ ,  $g(x) = 10x + 10$ ,  $x > 0$

**Qual cresce mais rápido?**

- Comparação para valores grandes de  $x$
- Seja  $f(x) = x^2 - x$ ,  $g(x) = 10x^2 + 10$ ,  $x > 0$

**Qual cresce mais rápido?**

- Comparação de primeira ordem,  
crescimento igual

# Notação O

- Seja  $f(n)$  uma função positiva,  $n > 0$
- Dizemos que  $f(n)$  é  $O$  de  $g(n)$  se  $f(n)$  é *limitada superiormente* por uma constante vezes  $g(n)$  para todo  $n$  grande suficiente
- Ou seja, se existe constante  $c > 0$ ,  $n_0 > 0$ , tal que para todo  $n > n_0$ ,  $f(n) \leq c g(n)$
- Dizemos neste caso que “ $f(n)$  é  $O(g(n))$ ” ou “ $f(n) = O(g(n))$ ” ou “ $f(n)$  pertence a  $O(g(n))$ ”
- Importante:  $c$  não pode depender de  $n$

# Notação O

- $O(g(n))$  define uma classe de funções

**Qual classe?**

- Todas as funções “menores ou iguais” a  $g(n)$
- Todas as funções  $f(n)$  tal que existe  $c > 0$  e  $n_0 > 0$ , tal que para todo  $n > n_0$   
 $f(n) \leq c g(n)$
- *Limitante superior assintótico para  $f(n)$*

# Exemplos

- Seja  $f(n) = 7n^2 + 10n + 2$ , para  $n > 0$
- $f(n)$  é  $O(n^2)$ ?
- $$\begin{aligned} f(n) &= 7n^2 + 10n + 2 \\ &< 7n^2 + 10n^2 + 2n^2 \\ &= 19n^2 \end{aligned}$$
- Com  $c = 19$ ,  $n_0 = 1$ , temos que  $f(n) \leq c n^2$  para todo  $n > n_0$
- $f(n)$  é  $O(n^3)$ ?
- $f(n)$  é  $O(n)$  ?

# Notação $\Omega$

- Complementar a notação  $O$ 
  - limite inferior assintótico
- Seja  $f(n)$  uma função positiva,  $n > 0$
- Dizemos que  $f(n)$  é omega de  $g(n)$  se  $f(n)$  é ao menos uma constante vezes  $g(n)$  para todo  $n$  grande suficiente
  - Ou seja, se existe constante  $c > 0$ ,  $n_0 > 0$ , tal que para todo  $n > n_0$ ,  $f(n) \geq c g(n)$
  - Importante:  $c$  não pode depender de  $n$

# Notação $\Omega$

- $\Omega(g(n))$  define uma classe de funções

**Qual classe?**

- Todas as funções “maiores ou iguais” a  $g(n)$
- Todas as funções  $f(n)$  tal que existe  $c > 0$  e  $n_0 > 0$ , tal que para todo  $n > n_0$   
 $f(n) \geq c g(n)$
- *Limitante inferior assintótico* para  $f(n)$

# Exemplos

- Seja  $f(n) = 7n^2 + 10n + 2$ , para  $n > 0$
- $f(n)$  é  $\Omega(n^2)$ ?
- $$\begin{aligned} f(n) &= 7n^2 + 10n + 2 \\ &> 7n^2 \end{aligned}$$
- Com  $c = 7$ ,  $n_0 = 1$ , temos que  $f(n) \geq c n^2$  para todo  $n > n_0$
- $f(n)$  é  $\Omega(n^3)$ ?
- $f(n)$  é  $\Omega(n)$  ?

# Notação $\Theta$

- Captura crescimento exato
  - limitante superior e inferior, simultaneamente
- Seja  $f(n)$  uma função positiva,  $n > 0$
- Dizemos que  $f(n)$  é  $\Theta(g(n))$  se  $f(n)$  cresce igual a  $g(n)$  ao menos de uma constante multiplicativa para todo  $n$  grande suficiente
- Ou seja, se  $f(n) = O(g(n))$  e  $f(n) = \Omega(g(n))$ , com duas constantes  $c_1$  e  $c_2$ , e dois  $n_1$  e  $n_2$

# Notação $\Theta$

- $\Theta(g(n))$  define uma classe de funções

**Qual classe?**

- Todas as funções “iguais” a  $g(n)$
- Todas as funções  $f(n)$ , tal que  
 $f(n) = O(g(n))$  e  $f(n) = \Omega(g(n))$
- *Limitante assintótico apertado (tight bound)*  
para  $f(n)$

# Exemplos

- Seja  $f(n) = 7n^2 + 10n + 2$ , para  $n > 0$
- $f(n)$  é  $\Theta(n^2)$ ?
- Sim, pois  $f(n) = O(n^2)$  e  $f(n) = \Omega(n^2)$
- $f(n)$  é  $\Theta(n^3)$ ?
- $f(n)$  é  $\Theta(n)$  ?

# Limite e Notação Θ

- Notação Θ pode ser calculada como limite
- Seja  $f(n)$  e  $g(n)$  duas funções positivas

- Se

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c > 0$$

- Então  $f(n) = \Theta(g(n))$
- Definição de limite atende as definições de  $O$  e  $\Omega$  para as funções

# Propriedades de O, Ω, Θ

- Considere as funções positivas  $f(n)$ ,  $g(n)$ ,  $h(n)$
- Transitividade
- Se  $f = O(g)$  e  $g = O(h)$ , então  $f = O(h)$
- Se  $f = \Omega(g)$  e  $g = \Omega(h)$ , então  $f = \Omega(h)$
- Se  $f = \Theta(g)$  e  $g = \Theta(h)$ , então  $f = \Theta(h)$
- Provar estas propriedades
  - duas primeiras implicam a terceira

# Propriedades de O, Ω, Θ

- Considere as funções positivas  $f(n)$ ,  $g(n)$ ,  $h(n)$
- Soma de funções
- Se  $f = O(h)$  e  $g = O(h)$ , então  $f + g = O(h)$
- Vale também para número fixo de parcelas de funções
- Se  $f = O(g)$  então  $f + g = \Theta(g)$ 
  - Provar este resultado interessante

# Polinômios

- Considere o polinômio

$$f(n) = a_d n^d + a_{d-1} n^{d-1} + \dots + a_0$$

- Crescimento assintótico dado pelo termo de mais alta ordem (que determina seu grau)
- Temos que  $f(n) = O(n^d)$
- Temos ainda que  $f(n) = \Omega(n^d)$
- Prova desta propriedade?

# Logaritmo

- Lembrando  $\log_b n$  é o número  $x$ , tal que  $b^x = n$
- Logaritmo cresce muito devagar
  - $\log$  (número de átomos no universo)  $\sim 60$
- Mais devagar do que qualquer polinômio
- Para todo  $b > 1$  e qualquer  $x > 0$ , temos  $\log_b n = O(n^x)$
- Não precisa informar a base em  $O(\log n)$ 
  - mudança de base é multiplicação por constante

# Exponencial

- Lembrando  $f(n) = r^n$  com  $r > 1$
- Crescem muito rápido
  - expoente é  $n$  (e não fixo, como polinômio)
- Mais rápido do que qualquer polinômio
- Para todo  $r > 1$  e todo  $d > 0$ , temos  
 $n^d = O(r^n)$
- Diferente de logaritmo, cada exponencial (base) define sua família
- Seja  $r > s > 1$
- Então nunca é o caso de  $r^n = \Theta(s^n)$ 
  - Prova?

# Tempo de Execução

	$n$	$n \log_2 n$	$n^2$	$n^3$	$1.5^n$	$2^n$	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	$10^{25}$ years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	$10^{17}$ years	very long
$n = 1,000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10,000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100,000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1,000,000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

- $n$  é o tamanho da entrada
  - número de “elementos” na entrada
- Tempo de execução típico em função de  $n$  para diferentes funções de crescimento

# Tempo de Execução Típico

- Logarítmico,  $O(\log n)$ 
  - encontrar um valor em um vetor ordenado (busca binária)
  - percorrer uma árvore balanceada
- Linear,  $O(n)$ 
  - calcular valor máximo em um vetor
  - imprimir valores de uma lista
  - BFS, DFS (onde n aqui é vértices + arestas)
- $O(n \log n)$ 
  - ordenar um vetor com merge-sort
  - Dijkstra

# Tempo de Execução Típico

- Quadrático,  $O(n^2)$

- Ordenação com quicksort (pior caso)
  - Encontrar o par de pontos mais próximos
  - Dijkstra (sem heap)

- Cúbico,  $O(n^3)$

- multiplicação de matrizes

- Exponencial,  $O(r^n)$

- TSP, problema do caxeiro viajante
  - muitos outros em grafos