

## Aula 05

### Equações com Recorrência

Prof. Marco Aurélio Stefanos

marco em dct.ufms.br

www.dct.ufms.br/~marco

Aula 05 – p. 1

## Busca em um vetor ordenado

**Entrada:** Um vetor ordenado crescentemente e um inteiro

**Saída:** O índice do número encontrado ou -1

$\langle a_0, a_1, a_2, \dots, a_{n-1} \rangle; k$	$\longrightarrow$	$j$
3, 6, 8, 9, 12, 19, 21; 9	$\longrightarrow$	3
3, 6, 8, 9, 12, 19, 21; 11	$\longrightarrow$	- 1

Algoritmo Busca Linear - Tempo  $O(n)$

Aula 05 – p. 2

## Busca Binária

Idéia: Divisão e Conquista

Busca\_Binária( $A[l \dots r], k$ )

```
1: if  $r < l$  then
2:    $index = -1$ 
3: else
4:    $mid = (l + r) / 2$ 
5:   if  $k = A[mid]$  then
6:      $index = mid$ 
7:   else
8:     if  $k < A[mid]$  then
9:        $index = \text{Busca\_Binária}(A[l \dots mid - 1], k)$ 
10:    else
11:       $index = \text{Busca\_Binária}(A[mid + 1 \dots r], k)$ 
```

Aula 05 – p. 3

## Análise do Busca Binária

$$T(n) = \begin{cases} c & \text{se } n \leq 1 \\ T(n/2) + c & \text{se } n > 1 \end{cases}$$

$$\begin{aligned} T(n) &= T(n/2) + c \\ &= (T(n/4) + c) + c \\ &= ((T(n/8) + c) + c) + c \\ &\vdots \\ &= T(1) + c \lg n \end{aligned}$$

Portanto  $T(n) = \Theta(\lg n)$

Aula 05 – p. 4

# Árvore de Decisão

- melhoramos o tempo da busca de  $\Theta(n)$  para  $\Theta(\log n)$ .  
Dá para fazer melhor?
- árvore de decisão
  - Para uma entrada de tamanho  $n$ , é uma árvore binária com nós rotulados de 0 a  $n - 1$
  - A raiz é rotulada com o índice da primeira entrada do vetor que o algoritmo busca binária compara à  $k$
  - Logo, para um dado nó  $i$ , o rótulo do filho esquerdo é o índice da entrada quando  $k < A[i]$  (análogo ao filho direito). Se o nó não tem filho esquerdo(direito) o algoritmo termina.

Aula 05 - p. 5

# Árvore de Decisão

- o pior caso do número de comparações é o maior caminho  $p$  da raiz até uma folha.
- Supondo que a árvore de decisão tem  $n$  nós e cada nó pelo menos 2 filhos.
  - $n \leq 1 + 2 + 4 + \dots + 2^{p-1} = 2^p - 1$
  - $2^p \geq n + 1 \Rightarrow p \geq \log(n + 1)$

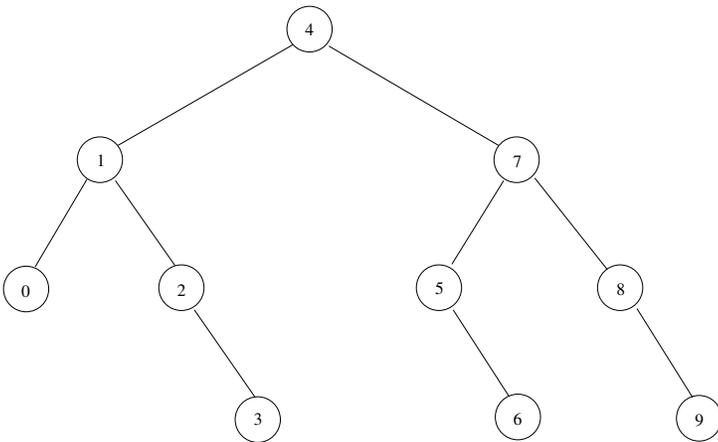
Será que uma árvore de decisão pode ter menos que  $n$  nós?

**Lema** Existe um nó rotulado com  $i$  para cada  $i$ ,  $0 \leq i \leq n - 1$

Aula 05 - p. 7

# Árvore de Decisão - Exemplo

Árvore de decisão para  $n = 10$



Aula 05 - p. 6

# Demostração

Suponha que não existe um nó rotulado com algum  $i$ . Considere dois vetores de entrada  $A1$  e  $A2$  tal que

$$A1[i] = k \text{ e } A2[i] = k' > k$$

para  $j < i$ ,  $A1[j] = A2[j]$  usando valores menores que  $k$

para  $j > i$ ,  $A1[j] = A2[j]$  usando valores maiores que  $k'$

Visto que nenhum nó da árvore de decisão está rotulado com  $i$  um algoritmo nunca compara  $k$  com  $A1[i]$  ou  $A2[i]$ , mas isto dá a mesma saída para ambas as entradas.

Tal algoritmo dá uma saída errada para pelo menos uma das entradas. Conclusão: A árvore tem pelo menos  $n$  nós.

**Teorema** Qualquer algoritmo com um vetor de tamanho  $n$  deve fazer ao menos  $\Theta(\log n)$  comparações para alguma entrada.

**Corolário** O algoritmo busca binária é ótimo

Aula 05 - p. 8

# Resolvendo Recorrências

**Recorrência:** Equação ou inequação que descreve uma função em termos de seus próprios valores em entradas menores.

- As análises do Mergesort e do Busca Binária necessitaram da resolução de uma recorrência
- Recorrências aparecem em análise de tempo de algoritmos recursivos, mas não só.
- Recorrências são como demais equações matemáticas
  - Necessitam de alguns truques
  - Há vários métodos de resolver: Iterativo, Substituição, Mestre.

Aula 05 – p. 9

# Método substituição

$$\begin{aligned}T(n) &= 4T(n/2) + n \\ &\leq 4c(n/2)^3 + n \\ &= \frac{c}{2}n^3 + n \\ &= cn^3 - \left(\frac{c}{2}n^3 - n\right) \\ &\leq cn^3\end{aligned}$$

Contanto que  $\frac{c}{2}n^3 - n \geq 0$ . Por exemplo para  $n = 1$  e  $c = 2$ .

Aula 05 – p. 11

# Método substituição

O método mais geral

- **Chute** uma equação para a solução
- **Verifique** por indução
- **Resolva** para achar as constantes

Exemplo

$$T(n) = 4T(n/2) + n$$

- Supor  $T(1) = \Theta(1)$
- Chute  $\Theta(n^3)$ . (prove  $O$  e  $\Omega$  separadamente)
- Suponha  $T(k) \leq ck^3$ , para  $k < n$
- Prove  $T(n) \leq cn^3$

Aula 05 – p. 10

# Método substituição

- Nós devemos também verificar as condições iniciais, ou seja a base da indução
- $T(n) = \Theta(1)$  para todo  $n < n_0$ , onde  $n_0$  é uma constante adequada.
- Para  $1 \leq n < n_0$  temos  $\Theta(1) \leq cn^3$ , para uma constante  $c$  suficientemente grande.

Este limitante superior não é justo !

Aula 05 – p. 12

# Método substituição

Vamos mostrar que  $T(n) = \Theta(n^2)$   
Suponha que  $T(k) \leq ck^2$ , para  $k < n$

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4c(n/2)^2 + n \\ &= cn^2 + n \\ &\leq cn^2 \quad \text{Errado !} \end{aligned}$$

Não há escolha para  $c$  que torne a expressão verdadeira

Aula 05 – p. 13

# Método Substituição

Um limitante superior mais justo.

$$T(n) = \Theta(n^2)$$

Idéia: Reduzir a hipótese de indução

Subtrair um termo de menor grau.

Hipótese indutiva:  $T(k) \leq c_1k^2 - c_2k$ , para  $k < n$

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4(c_1(n/2)^2 - c_2n/2) + n \\ &= c_1n^2 - 2c_2n + n \\ &= c_1n^2 - c_2n - (c_2n - n) \\ &\leq c_1n^2 - c_2n \end{aligned}$$

Tomando  $c_2 > 1$

Aula 05 – p. 14

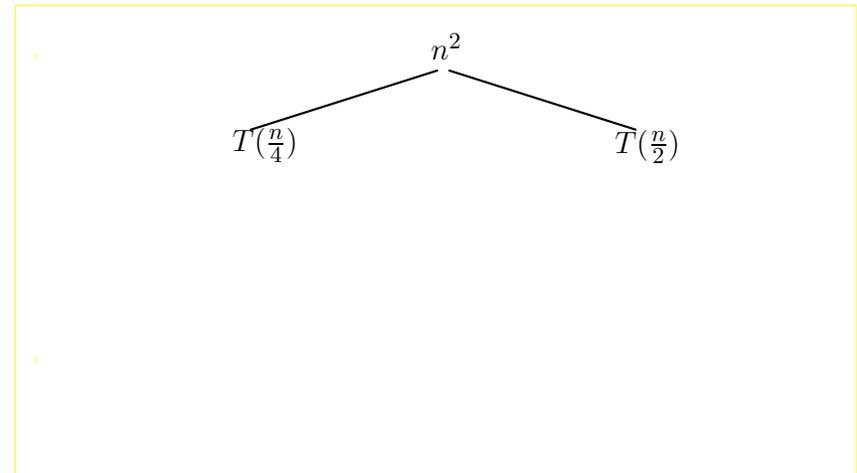
# Árvore de Recursão

- Uma árvore de recursão modela o tempo de execução recursivo de um algoritmo
- A árvore de recursão é uma forma de produzir um chute para o método de substituição
- A árvore de recursão pode não ser confiável.
- A árvore de recursão ajuda na intuição.

Aula 05 – p. 15

# Exemplo da Árvore de Recursão

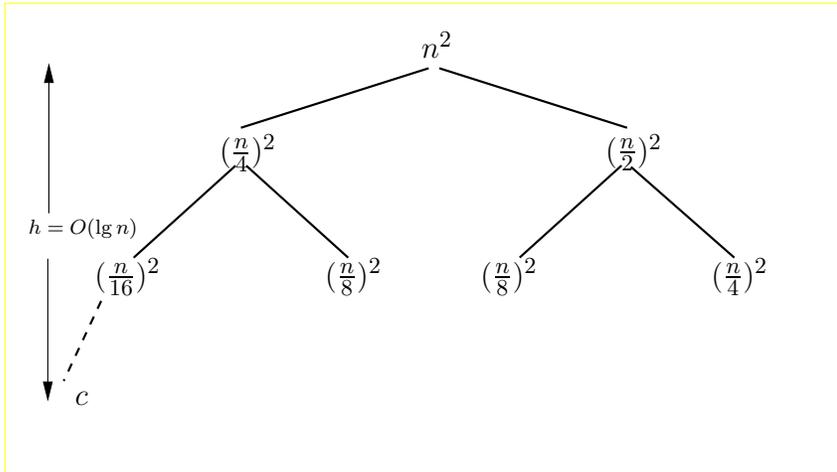
Resolva  $T(n) = T(n/4) + T(n/2) + n^2$



Aula 05 – p. 16

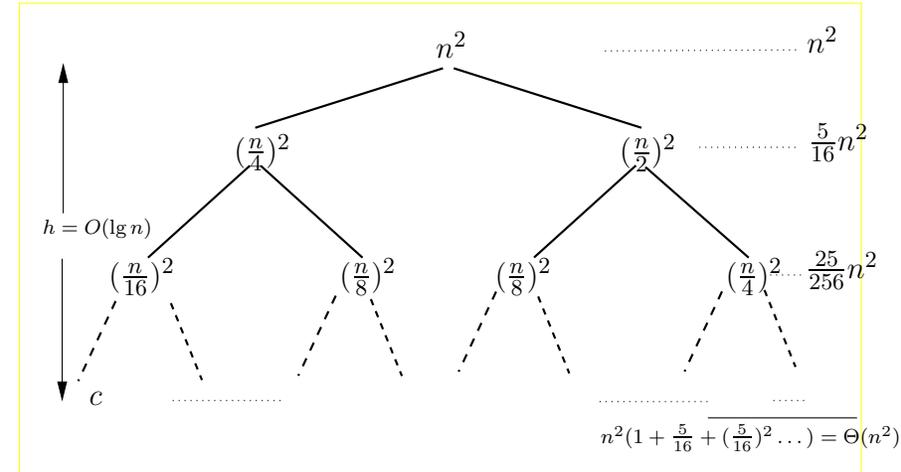
# Exemplo da Árvore de Recursão

Resolva  $T(n) = T(n/4) + T(n/2) + n^2$



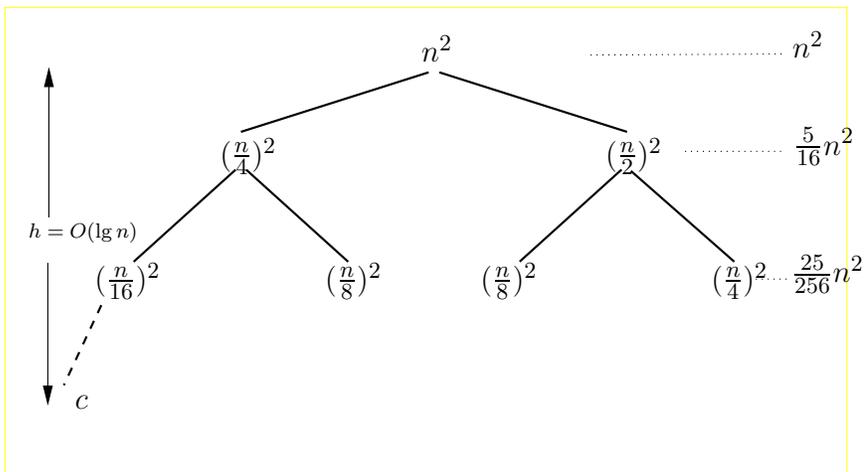
# Exemplo da Árvore de Recursão

Resolva  $T(n) = T(n/4) + T(n/2) + n^2$



# Exemplo da Árvore de Recursão

Resolva  $T(n) = T(n/4) + T(n/2) + n^2$



# Método Mestre

O Método Mestre se aplica a recorrências da forma:

$$T(n) = aT(n/b) + f(n),$$

Onde  $a \geq 1$ ,  $b > 1$  são constantes e  $f(n)$  é uma função assintoticamente positiva.

Há três casos comuns: Compare  $f(n)$  com  $n^{\log_b a}$

- se  $f(n) = O(n^{\log_b a - \epsilon})$  para algum  $\epsilon > 0$ 
  - $f(n)$  cresce menos que  $n^{\log_b a}$  (por um fator  $n^\epsilon$ )
  - $T(n) = \Theta(n^{\log_b a})$
- se  $f(n) = \Theta(n^{\log_b a} \lg^k n)$ , para algum  $k > 0$ 
  - $f(n)$  e  $n^{\log_b a}$  crescem assintoticamente iguais
  - $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$

# Método Mestre

Há três casos comuns: Compare  $f(n)$  com  $n^{\log_b a}$

- se  $f(n) = \Omega(n^{\log_b a + \epsilon})$  para algum  $\epsilon > 0$ 
  - $f(n)$  cresce polinomialmente mais rápida que  $n^{\log_b a}$  (por um fator  $n^\epsilon$ )
  - se  $f$  satisfaz a condição de regularidade que  $af(n/b) \leq cf(n)$
  - então  $T(n) = \Theta(f(n))$

Aula 05 - p. 18

# Método Mestre - Exemplos

$$T(n) = 4T(n/2) + n$$

$$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2, f(n) = n$$

**Caso 1**  $f(n) = O(n^{2-\epsilon})$  para  $\epsilon = 1$

**Solução :**  $T(n) = \Theta(n^2)$

$$T(n) = 4T(n/2) + n^2$$

$$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2, f(n) = n^2$$

**Caso 2**  $f(n) = \Theta(n^2 \log^0 n)$  i.e  $k = 0$

**Solução :**  $T(n) = \Theta(n^2 \log n)$

Aula 05 - p. 19

# Método Mestre - Exemplos

$$T(n) = 4T(n/2) + n^3$$

$$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2, f(n) = n^3$$

**Caso 3**  $f(n) = \Omega(n^{2+\epsilon})$  para  $\epsilon = 1$  e

Cond. Reg.  $4(cn/2)^3 \leq cn^3$  para  $c = 1/2$

**Solução :**  $T(n) = \Theta(n^3)$

$$T(n) = 4T(n/2) + n^2 / \log n$$

$$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2, f(n) = n^2 / \log n$$

Método Master não se aplica. Para todo  $\epsilon > 0$  temos

$$n^\epsilon = \omega(\log n)$$

Aula 05 - p. 20

# Exercícios - Resolva

- Usando método iterativo

- $T(n) = 3T(n/4) + n$

- Usando método substituição

- $T(n) = 2T(n/2) + n$

- Usando método master

- $T(n) = 9T(n/3) + n$

- $T(n) = T(2n/3) + 1$

- $T(n) = 3T(n/4) + n \log n$

Aula 05 - p. 21