Introdução à Ciência da Computação II

Merge-Sort Pt. I: Algoritmo Básico & Desempenho

Prof. Ricardo J. G. B. Campello

1

Aula de Hoje

- Merge-Sort Conceitual
- Análise de Desempenho
- Implementação em C (Recursiva)

2

Algoritmo Merge-Sort

- Merge-sort é um algoritmo de ordenação baseado no paradigma de **Divisão e Conquista**
- Divisão e Conquista é um padrão de projeto de algoritmos constituído de 3 fases:
  - Divisão: divide o conjunto de dados de entrada V em dois subconjuntos disjuntos V_1 e V_2
  - Recursão: resolve recursivamente os subproblemas associados a V_1 e V_2
  - Conquista: combina as soluções individuais para V_1 e V_2 em uma solução única para V

3

Pseudo-Código

- Dada uma seqüência V (p. ex. vetor ou lista) com n elementos:
  - Divisão: particiona V em duas seqüências V_1 e V_2 com aproximadamente $n/2$ elementos cada
  - Recursão: ordena V_1 e V_2 recursivamente
  - Conquista: faz a fusão de V_1 e V_2 em uma única seqüência ordenada

Algoritmo *Merge_Sort*(V, n)
Entrada: Vetor V com n elementos
Saída: Vetor V ordenado
se ($n > 1$) **então**
  $n_d \leftarrow n/2$ /* div. inteira */
  $V_1 \leftarrow V[0, \ldots, n_d - 1]$
  $V_2 \leftarrow V[n_d, \ldots, n - 1]$
  Merge_Sort(V_1, n_d)
  Merge_Sort(V_2, $n - n_d$)
  $V \leftarrow$ *Merge*(V_1, n_d, V_2, $n - n_d$)

4

Fusão de 2 Seqüências Ordenadas

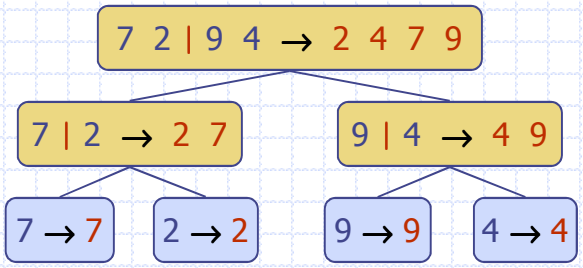
Algoritmo *Merge*(A, n_a, B, n_b)
  Entrada: Vetores ordenados A e B com n_a e n_b elementos, respectivamente
  Saída: Vetor ordenado com fusão de A e B
  $V \leftarrow$ vetor novo com $n_a + n_b$ elementos
  $cont, cont\_A, cont\_B \leftarrow 0$
  enquanto ($cont\_A < n_a$ **e** $cont\_B < n_b$) **faça**
    se $A[cont\_A] \leq B[cont\_B]$ **então**
      $V[cont{+}{+}] \leftarrow A[cont\_A{+}{+}]$
    senão
      $V[cont{+}{+}] \leftarrow B[cont\_B{+}{+}]$
  enquanto ($cont\_A < n_a$) **faça**
    $V[cont{+}{+}] \leftarrow A[cont\_A{+}{+}]$
  enquanto ($cont\_B < n_b$) **faça**
    $V[cont{+}{+}] \leftarrow B[cont\_B{+}{+}]$
  retorne V

Fusão de 2 Seqüências Ordenadas

- **Nota**:
  - É simples verificar que a fusão de duas seqüências ordenadas com n_a e n_b elementos, respectivamente, consome tempo $O(n_a + n_b)$
  - De fato, cada elemento é atribuído exatamente uma única vez ao destino e para cada atribuição uma única comparação é feita
  - Essa análise vale para vetores ou listas dinâmicas, e será importante para avaliação da complexidade computacional do algoritmo Merge-Sort

Árvore Merge-Sort

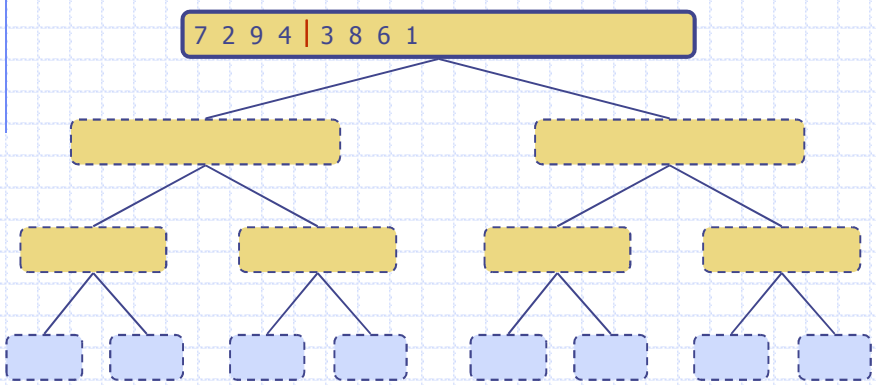
- Execução Merge-Sort como árvore de recursão binária:
  - Cada nó representa uma chamada recursiva e armazena:
    - Seqüência não ordenada antes da execução e sua partição
    - Seqüência ordenada ao fim da execução
  - Raiz representa a chamada inicial
  - Folhas representam chamadas para subseqüências unitárias



7

Exemplo de Execução

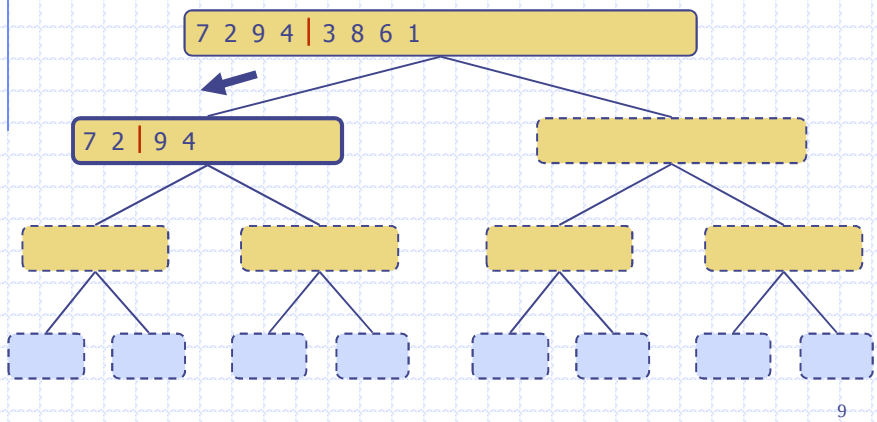
- Partição:



8

Exemplo de Execução (cont.)

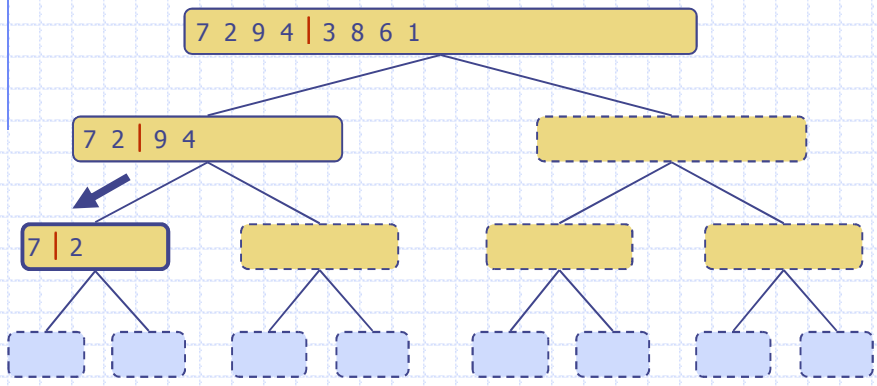
- Chamada recursiva e partição:



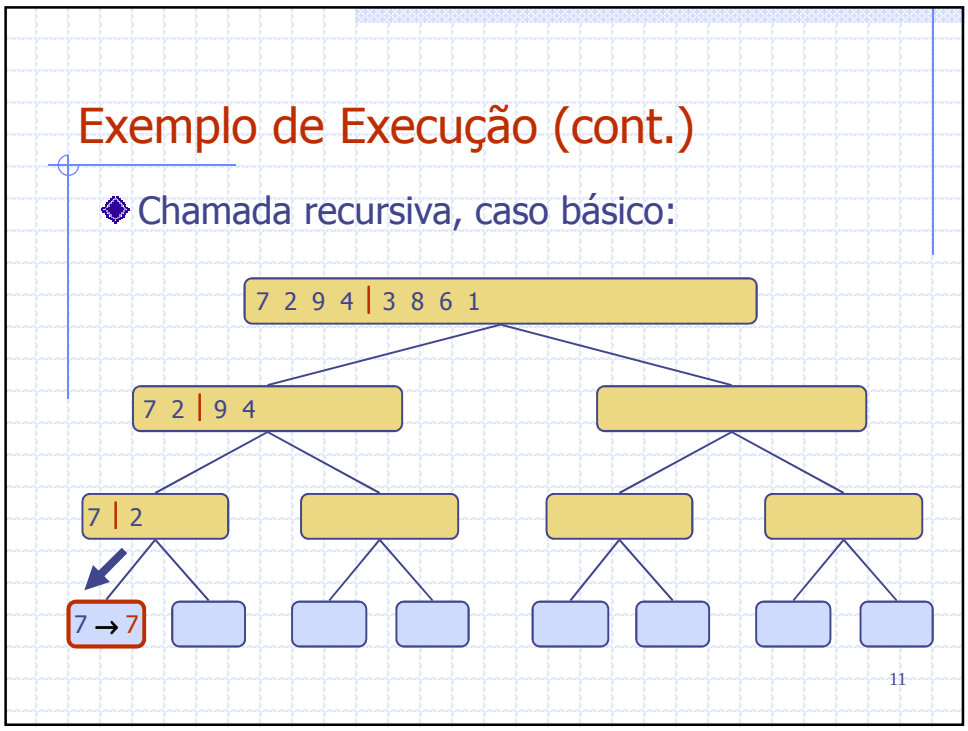
9

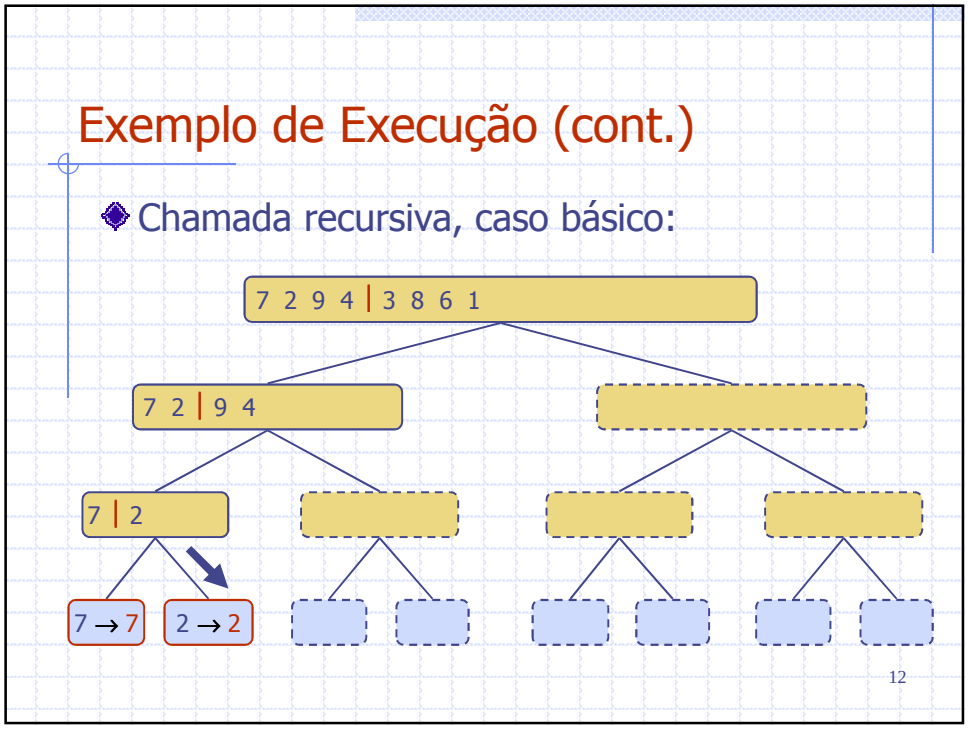
Exemplo de Execução (cont.)

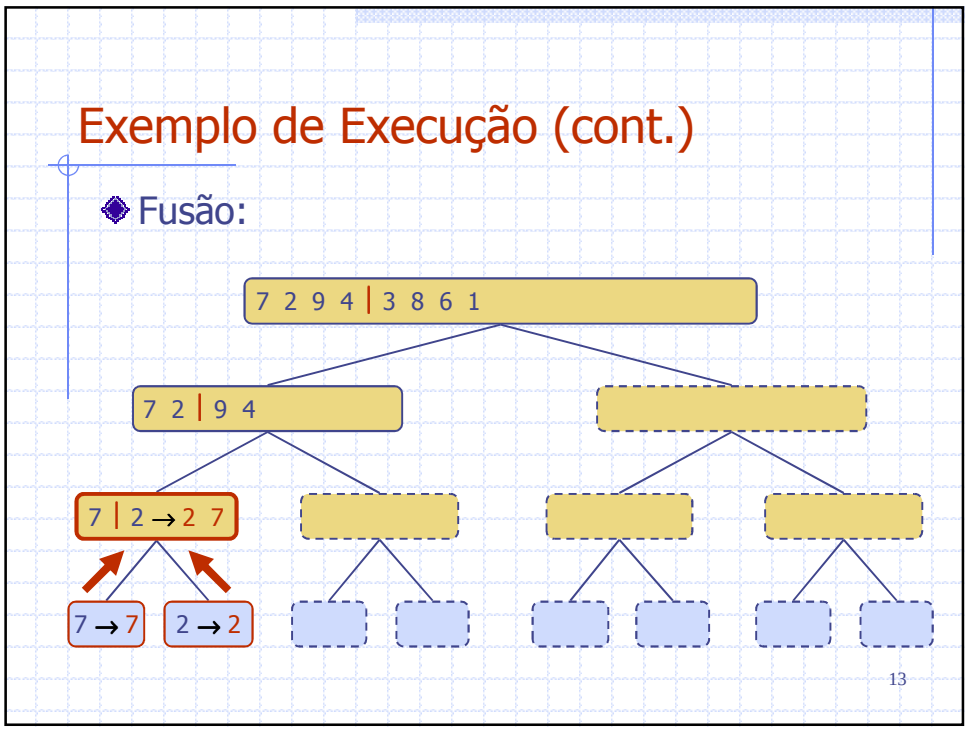
- Chamada recursiva e partição:

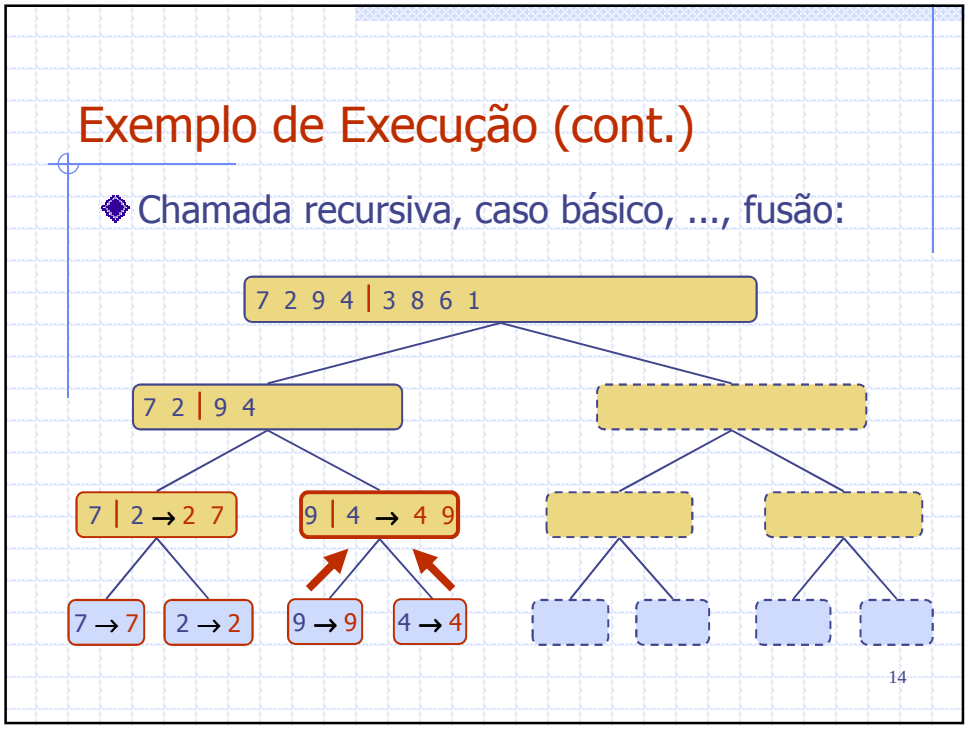


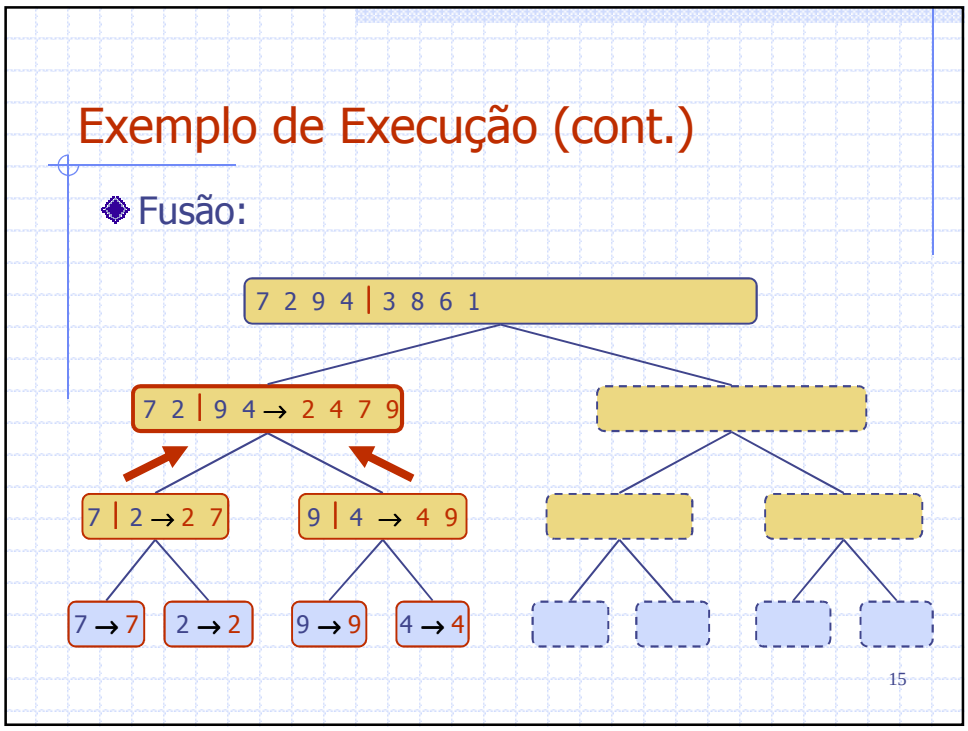
10

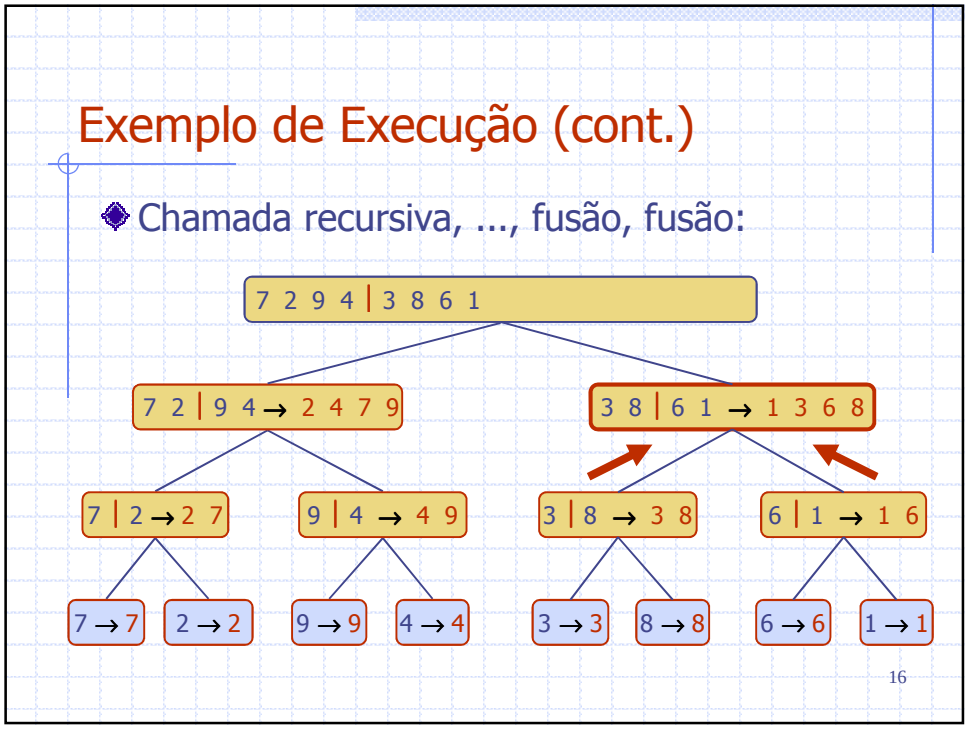

Exemplo de Execução (cont.)
Chamada recursiva, caso básico:
7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2
7 → 7
11


Exemplo de Execução (cont.)
Chamada recursiva, caso básico:
7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2
7 → 7
2 → 2
12


Exemplo de Execução (cont.)
Fusão:
7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2 → 2 7
7 → 7
2 → 2
13

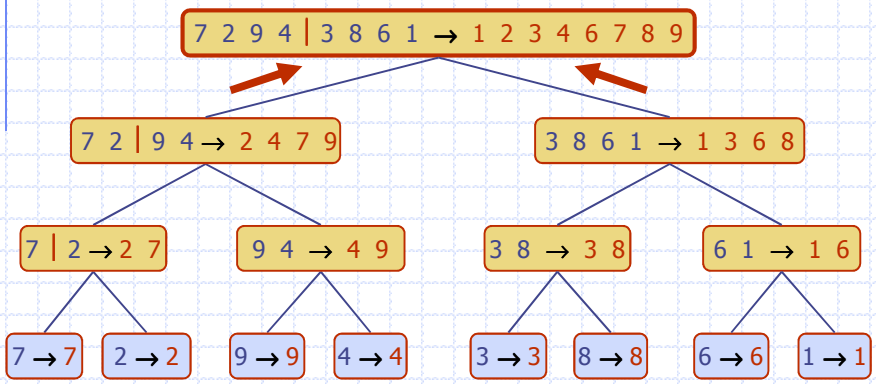

Exemplo de Execução (cont.)
Chamada recursiva, caso básico, ..., fusão:
7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2 → 2 7
9 | 4 → 4 9
7 → 7
2 → 2
9 → 9
4 → 4
14


Exemplo de Execução (cont.)
Fusão:
7 2 9 4 | 3 8 6 1
7 2 | 9 4 → 2 4 7 9
7 | 2 → 2 7
9 | 4 → 4 9
7 → 7
2 → 2
9 → 9
4 → 4
15


Exemplo de Execução (cont.)
Chamada recursiva, ..., fusão, fusão:
7 2 9 4 | 3 8 6 1
7 2 | 9 4 → 2 4 7 9
3 8 | 6 1 → 1 3 6 8
7 | 2 → 2 7
9 | 4 → 4 9
3 | 8 → 3 8
6 | 1 → 1 6
7 → 7
2 → 2
9 → 9
4 → 4
3 → 3
8 → 8
6 → 6
1 → 1
16

Exemplo de Execução (cont.)

- Fusão:

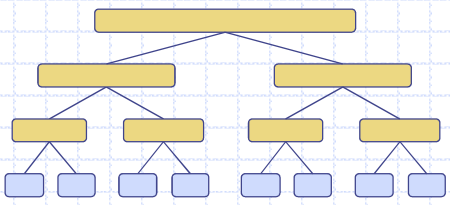


17

Análise do Algoritmo

- Altura da árvore é $O(\log n)$ dado que a cada chamada recursiva (nível / profundidade) divide-se a seqüência pela metade:
  - $n / 2^i \geq 1 \quad \Rightarrow \quad i \leq \log n$

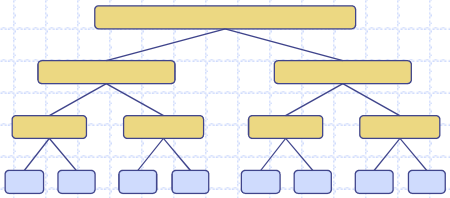
profund.	no. seqs.	tamanho unitário
0	1	n
1	2	$\sim n/2$
i	2^i	$\sim n/2^i$
…	…	…



Análise do Algoritmo

- A quantidade total de elementos nos nós de um dado nível qualquer da árvore é sempre n
- Logo, a complexidade da etapa de fusão em cada nível é $O(n)$

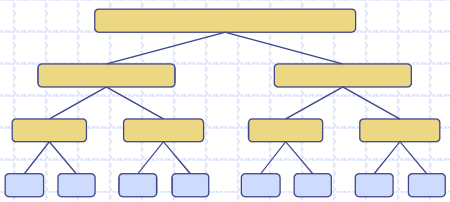
profund.	no. seqs.	tamanho unitário
0	1	n
1	2	$\sim n/2$
i	2^i	$\sim n/2^i$
…	…	…



Análise do Algoritmo

- As fusões são o gargalo computacional
- Como são $O(\log n)$ fusões a um custo $O(n)$ cada, o custo total do algoritmo é $O(n \log n)$
  - **Melhor Caso**, **Pior Caso** e **Caso Médio**

profund.	no. seqs.	tamanho unitário
0	1	n
1	2	$\sim n/2$
i	2^i	$\sim n/2^i$
…	…	…



Implementação Elementar em C

- No quadro...

21

Exercícios

- Analise se o algoritmo Merge-Sort é estável ou não e discuta o porquê, apresentando exemplos
- Ilustre a árvore de recursão binária Merge-Sort para diferentes seqüências (conforme exemplo nos slides)

22

Bibliografia

- M. T. Goodrich & R. Tamassia, *Data Structures and Algorithms in C++/Java*, John Wiley & Sons, 2002/2005
- N. Ziviani, *Projeto de Algoritmos*, Thomson, 2a. Ed, 2004

23