

Aula 04

Divisão e Conquista - MergeSort

Prof. Marco Aurélio Stefanos

marco em dct.ufms.br

www.dct.ufms.br/~marco

Aula 04 – p. 1

Divisão e Conquista

Método de Divisão e Conquista

- **Divisão:** Divida o problema em duas ou mais partes, criando subproblemas menores.
- **Conquista:** Os subproblemas são resolvidos recursivamente usando divisão e conquista. Caso os subproblemas sejam suficientemente pequenos resolva-os de forma direta.
- **Combina:** Tome cada uma das partes e junte-as todas de forma a resolver o problema original.

Aula 04 – p. 2

Mergesort

- Mergesort é um algoritmo de ordenação recursivo
- Ele recursivamente ordena as duas metades do vetor
- Usa a estratégia de divisão e conquista
- Mergesort é um algoritmo eficiente
- Tem tempo de execução $O(n \log n)$

Aula 04 – p. 3

Algoritmo Mergesort

- Caso o tamanho do vetor seja maior que 1
 1. divida o vetor no meio
 2. ordene a primeira metade recursivamente
 3. ordene a segunda metade recursivamente
 4. intercale as duas metades
- Senão devolva o elemento

Aula 04 – p. 4

Código do Mergesort

MergeSort(A, p, r)

```
1: if  $p < r$  then  
2:    $q = (p + r) / 2$   
3:   MergeSort( $A, p, q$ )  
4:   MergeSort( $A, q + 1, r$ )  
5:   Merge( $A, p, q, r$ )
```

Chamada Inicial: MergeSort($A, 1, n$)

Análise da intercalação

Problema: Dados $A[p \dots q]$ e $A[q+1 \dots r]$ crescentes, reorganizar $A[p \dots r]$ de modo que ele fique em ordem crescente.

Entrada:

	p			q				r	
A	22	33	55	77	99	11	44	66	88

Saída:

	p	q				r			
A	11	22	33	44	55	66	77	88	99

Intercalação

A intercalação de dois vetores ordenados pode ser feito em tempo linear

- Uma variável em cada vetor indica o próximo elemento a ser inserido a lista intercalada
- Enquanto ambas os vetores tiverem elementos
- Coloque o menor entre os dois elemento indicados no vetor intercalado e incremente índice respectivo
- Quando um dos vetores não tiver mais elementos, concatene o outro no final do vetor intercalado.

Intercalação

Merge(A, p, q, r)

```
1: for  $i = p$  to  $q$  do  
2:    $B[i] = A[i]$   $\triangleright$   $B[p \dots r]$  vetor auxiliar  
3: for  $j = q + 1$  to  $r$  do  
4:    $B[r + q + 1 - j] = A[j]$   
5:  $i = p$   
6:  $j = r$   
7: for  $k = p$  to  $r$  do  
8:   if  $B[i] \leq B[j]$  then  
9:      $A[k] = B[i]$   
10:     $i = i + 1$   
11:  else  
12:     $A[k] = B[j]$   
13:     $j = j - 1$ 
```

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é ($n := r - p + 1$):

linha	todas as execuções da linha
1	$= q - p + 2 = n - r + q + 1$
2	$= q - p + 1 = n - r + q$
3	$= r - (q + 1) + 2 = n - q + p$
4	$= r - (q + 1) + 1 = n - q + p - 1$
5	$= 1$
6	$= 1$
7	$= r - p + 2 = n + 1$
8	$= r - p + 1 = n$
9–13	$= 2(r - p + 1) = 2n$

total $= 8n - 2(r - p + 1) + 5 = 6n + 5$

Aula 04 – p. 9

Consumo de tempo

Quanto tempo consome em função de $n := r - p + 1$?

linha	consumo de todas as execuções da linha
1–4	$O(n)$
5–6	$O(1)$
7	$nO(1) = O(n)$
8	$nO(1) = O(n)$
9–13	$nO(1) = O(n)$

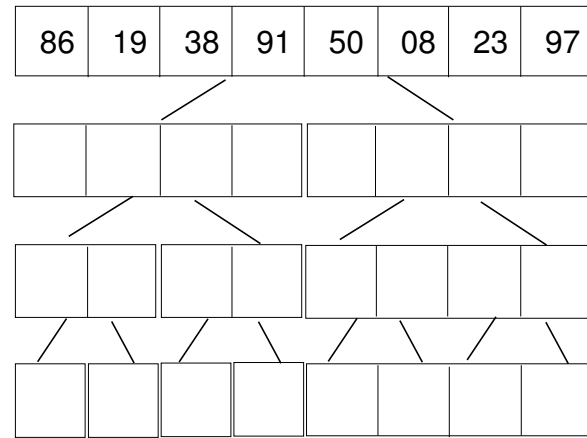
total $O(4n + 1) = O(n)$

Conclusão:

O algoritmo consome $O(n)$ unidades de tempo.

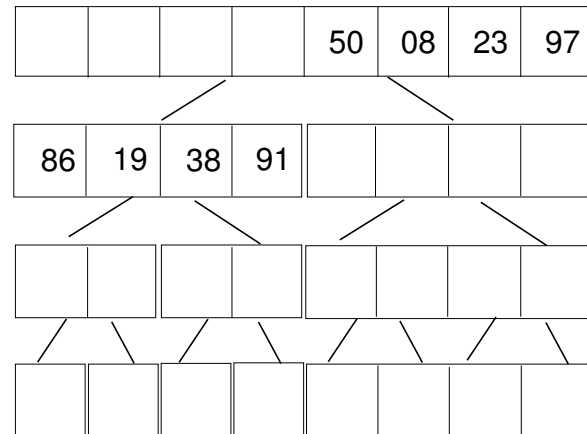
Aula 04 – p. 10

mergesort



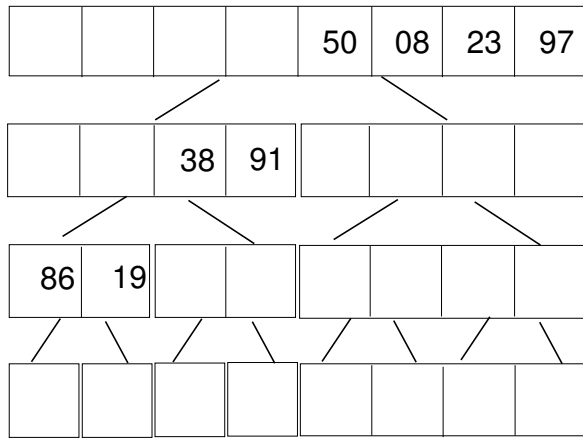
Aula 04 – p. 11

mergesort

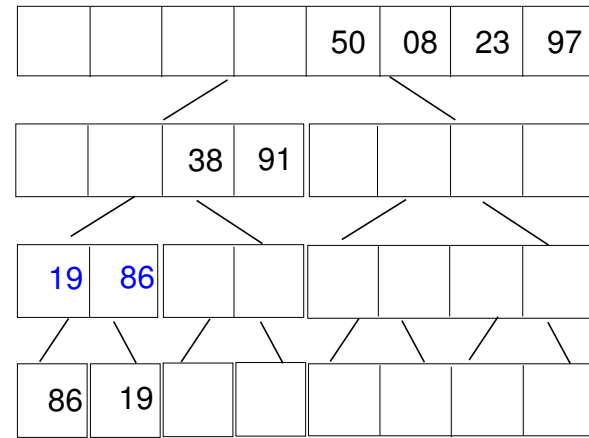


Aula 04 – p. 11

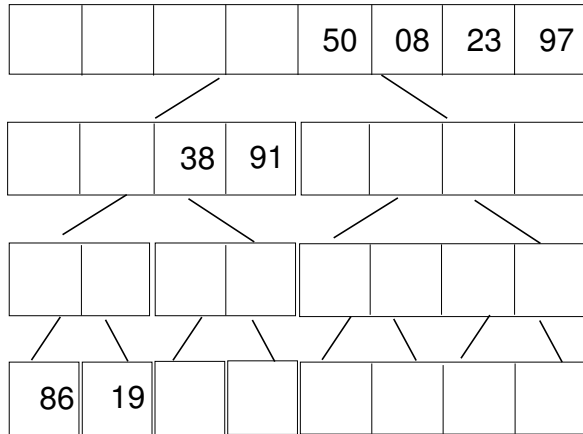
mergesort



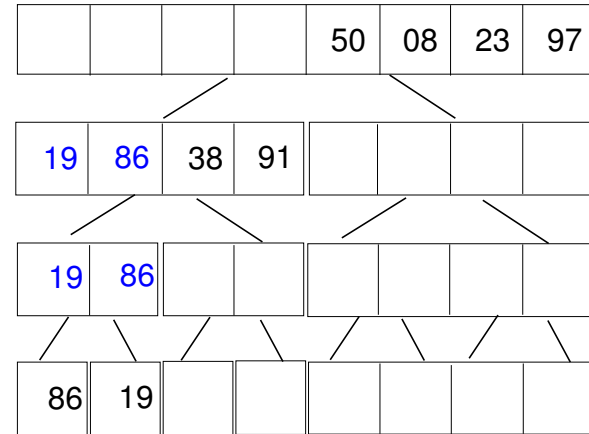
mergesort



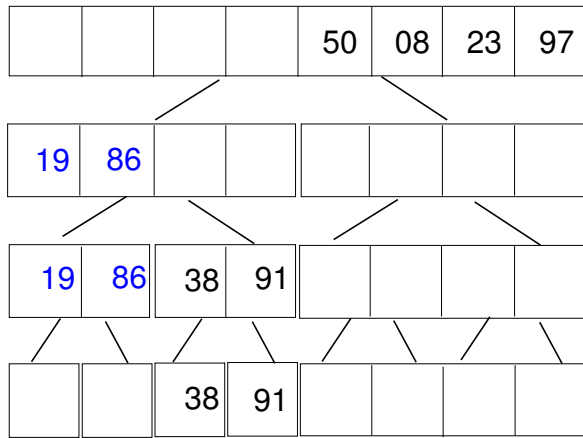
mergesort



mergesort

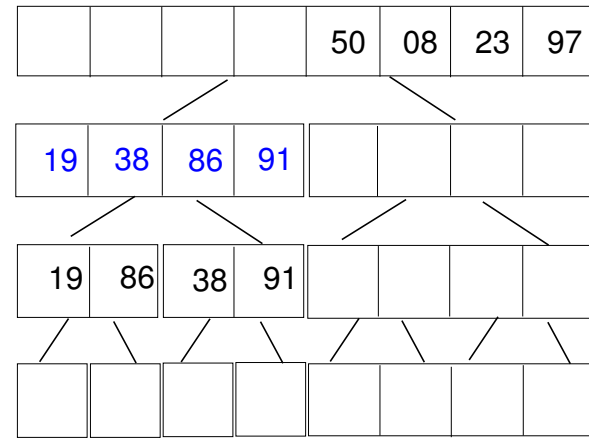


mergesort



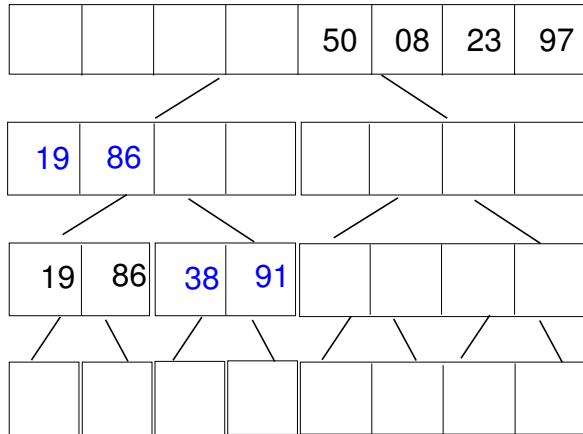
Aula 04 – p. 11

mergesort



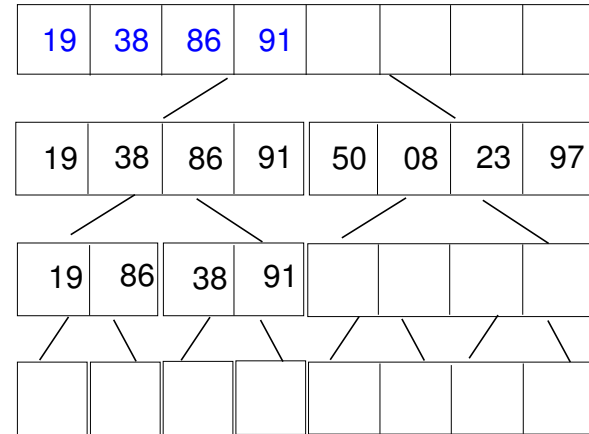
Aula 04 – p. 11

mergesort



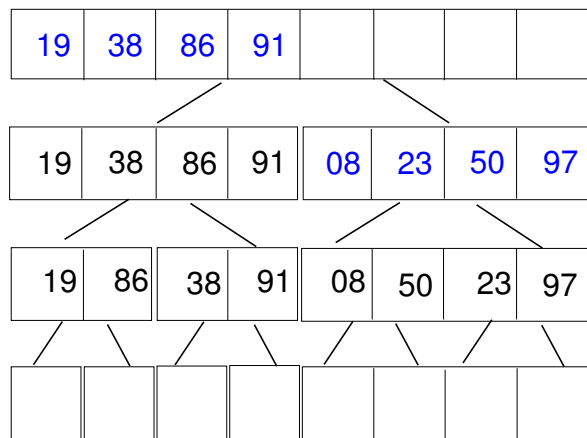
Aula 04 – p. 11

mergesort



Aula 04 – p. 11

mergesort



Aula 04 – p. 11

Correção do Merge-Sort

Teorema O Mergesort está correto.

Prova: Indução em n (tamanho da entrada)

Base $n = 1$, $r - p = 1 \Rightarrow r = p + 1 \Rightarrow q = \lceil \frac{2p+1}{2} \rceil = p$. Então ambas as chamadas recursivas voltam sem alterar o vetor e o Procedimento Merge intercala um elemento. **Ok!**

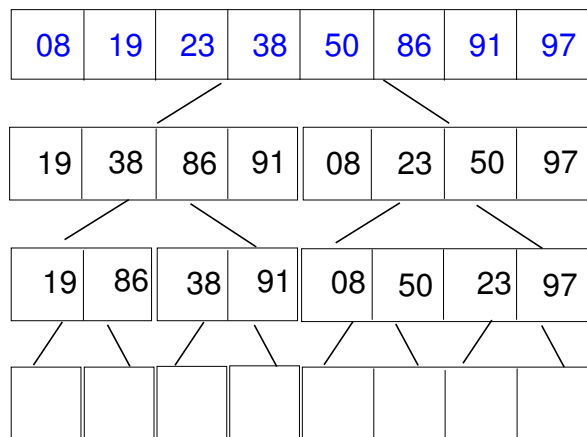
Hipótese de Indução Supor que o algoritmo está correto para toda entrada menor que n .

Passo Indutivo Na Linha 2 q assume o índice do meio do vetor. Na Linha 3 e 4, o Algoritmo é chamado para as duas metades do vetor. Como pela **Hipótese de Indução** ambos resolvem suas respectivas metades, na Linha 5 o Merge intercala duas listas ordenadas e portanto segue o resultado.

(Provar que o Procedimento Merge está correto **Exerc.**)

Aula 04 – p. 12

mergesort



Aula 04 – p. 11

Análise de Divisão e Conquista

- Quando um algoritmo contém chamadas recursivas, o cálculo de seu tempo de execução pode usar recorrências.
- Para o método de Divisão e Conquista
- Seja $T(n)$ o tempo do algoritmo. Suponha que dividimos em a subproblemas de tamanho n/b cada e seja $D(n)$ o tempo para dividir os subproblemas e $C(n)$ o tempo para combiná-los. Então

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{se } n > c \end{cases}$$

Aula 04 – p. 13

Análise do Merge-Sort

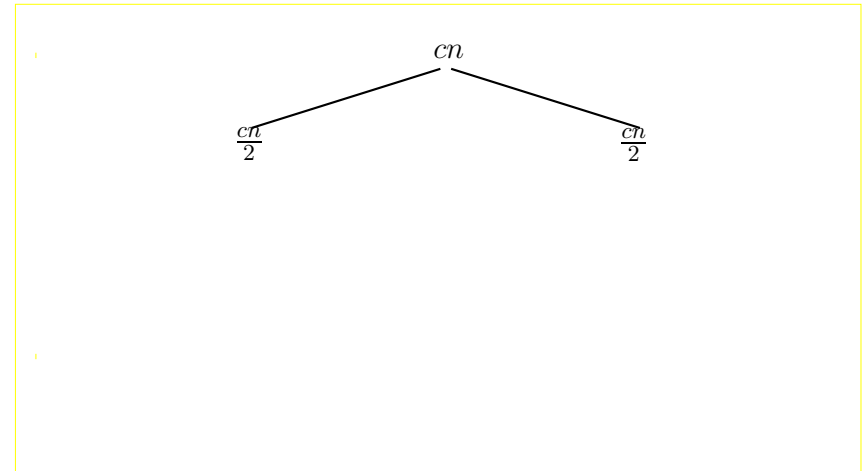
- Dividir Tempo Constante $\Theta(1)$
- Conquista Dois problemas de $n/2$ cada: $2T(n/2)$
- Combina Tempo do Merge $\Theta(n)$

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

Como Resolver?

Aula 04 – p. 14

Árvore de Recursão



Aula 04 – p. 15

Análise do Merge-Sort

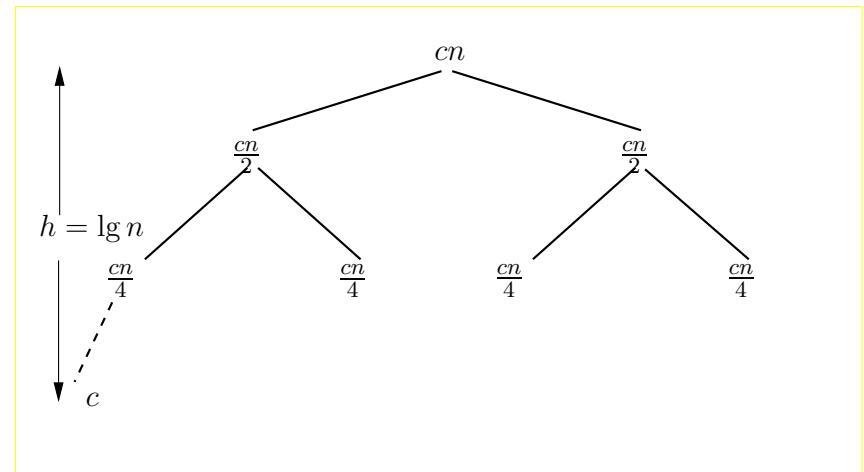
- Dividir Tempo Constante $\Theta(1)$
- Conquista Dois problemas de $n/2$ cada: $2T(n/2)$
- Combina Tempo do Merge $\Theta(n)$

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

Como Resolver? Árvore de Recursão

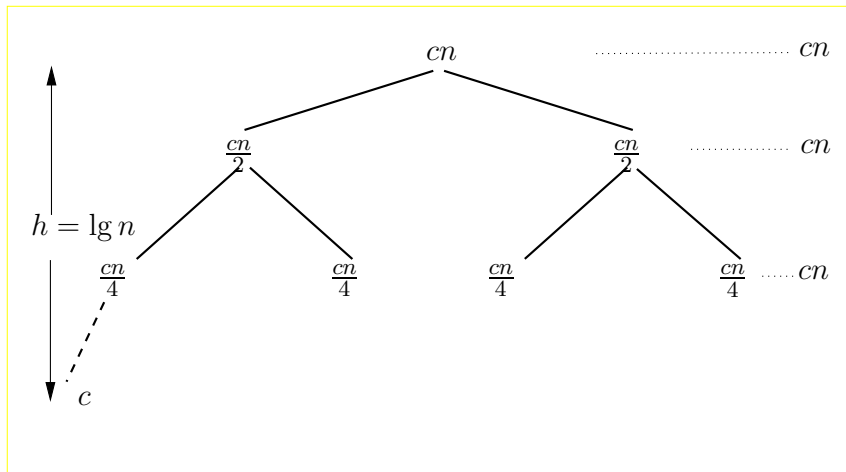
Aula 04 – p. 14

Árvore de Recursão



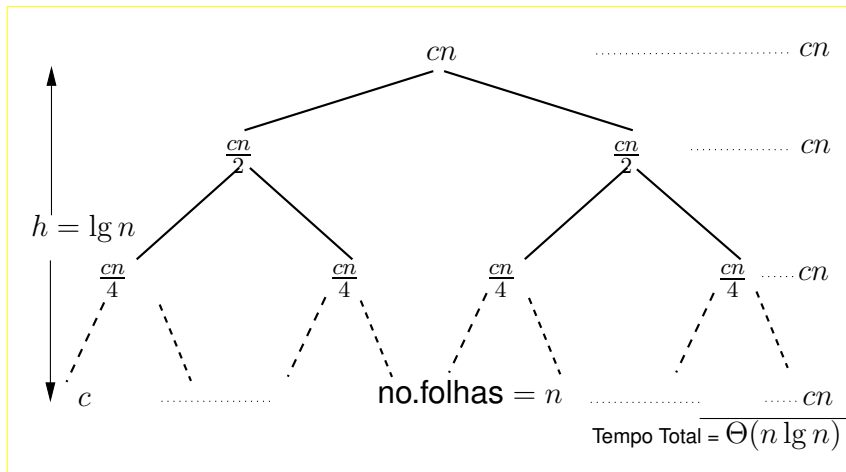
Aula 04 – p. 15

Árvore de Recursão



Aula 04 – p. 15

Árvore de Recursão



Aula 04 – p. 15