Mergesort

Algoritmos e Estruturas de Dados

Prof. Dr. Luciano Demétrio Santos Pacífico
{luciano.pacifico@ufrpe.br}

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Conteúdo

- **Introdução**
- **Paradigma Dividir-para-Conquistar**
- ***Mergesort***

Introdução

UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Recursividade

- Muitos algoritmos úteis possuem natureza recursiva, ou seja, para que um dado problema seja resolvido, o algoritmo chama a si mesmo diversas vezes, solucionando a cada etapa um subproblema do problema global.

- A solução final é obtida por uma combinação dos resultados dos subproblemas.

Problema da Ordenação

- Em muitos problemas práticos, a solução seria mais facilmente encontrada se os dados armazenados na estrutura adotada estivessem ordenados.

- Exemplo: a busca em uma lista ordenada pode ter custo médio menor do que a busca em uma lista não ordenada.

- Os dados podem ser ordenados de forma crescente ou decrescente.

Problema da Ordenação

- Como todas as aplicações estudadas até o momento consideram a existência de ao menos um atributo numérico (o atributo ***chave***), podemos considerar que as estruturas de dados vistas, em sua maioria, poderiam suportar a ideia de ordenação.

- Deve-se levar em consideração que nem todos os tipos de dados possuem a relação de ordenação intrinsecamente definida.

Problema da Ordenação

- Definição: dado um vetor **x** de *n* números, ordenar seus elementos (em ordem crescente, ou decrescente).

- Ex.: Sendo **x** = [32, 14, 59, 1, 4, 98, 15], e se o problema for de ordenação crescente, a saída **y** esperada é tal que **y** = [1, 4, 14, 15, 32, 59, 98].

Problema da Ordenação

- Estudaremos três novas abordagens para a solução do problema de ordenação:
  - *Mergesort*;
  - *Quicksort*;
  - *Heapsort*.
- Lembrando: na aula 1, vimos o algoritmo ***Insertion-Sort***, que também executa a tarefa de ordenação.

Paradigma Dividir-para-Conquistar

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Dividir-para-Conquistar

- Muitos problemas computacionais apresentam alto grau de dimensionalidade.

- Em alguns casos, o problema global pode ser dividido em subproblemas.

- A solução para o problema global é obtida a partir das soluções encontradas para esses subproblemas.

Dividir-para-Conquistar

- No modelo dividir-para-conquistar o problema global é dividido em subproblemas menores e independentes.
- Os subproblemas são iguais ao problema global.
- As divisões ocorrem até que tais subproblemas possam ser resolvidos.
- Após a solução dos subproblemas, os mesmos são combinados para a obtenção de uma solução definitiva para o problema inicial.

Dividir-para-Conquistar

- Existem três condições que indicam que a estratégia de divisão e conquista pode ser utilizada com sucesso em um problema:
  - Deve ser possível decompor o problema em subproblemas;
  - A combinação dos resultados dever ser eficiente (trivial, se possível);
  - Os subproblemas devem ser mais ou menos do mesmo tamanho.

Dividir-para-Conquistar

- Etapas:
  - Dividir: Divide o problema global em subproblemas menores e que são instâncias do problema global.
  - Conquistar: Os subproblemas são solucionados recursivamente. Se os subproblemas já forem pequenos o suficiente, uma resposta direta é encontrada.
  - Combinar: combina as soluções dos subproblemas para a formação da solução global para o problema original.
- O algoritmo *Mergesort* segue este paradigma.

Mergesort

UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Mergesort

- *Mergesort*

  - Dividir: divide a sequência de n elementos em duas subsequência de $n/2$ elementos cada.

  - Conquistar: Ordena as subsequências recursivamente usando o próprio *Mergesort*.

  - Combinar: combina as duas subsequências ordenadas para formar uma resposta ordenada (procedimento `merge`).

Mergesort

- A recursão será executada até que uma sequência de **tamanho 1** seja encontrada, pois uma sequência com apenas um elemento **já está ordenada**.

- A operação chave do *Mergesort* é a combinação de duas sequências ordenadas na etapa de combinação.

- O procedimento `merge(A, p, q, r)` executa essa combinação, onde A é a sequência de elementos, p, q e r são índices desta sequência tais que $p \leq q < r$.

- O procedimento *Merge* assume que as subsequências A[p..q] e A[q+1..r] **já estão ordenadas**, sendo responsável pela formação da **nova subsequência ordenada** A[p..r].

Mergesort

- Para os exemplos a seguir, não haverá necessidade da definição de novas estruturas básicas para a execução do *Mergesort*, pois precisamos apenas de um vetor numérico, que será representado por um **Array**<**inteiro**>.
- Para os problemas de ordenação numérica, será permitida a inserção de valores iguais no vetor (**Array**<**inteiro**>) a ser ordenado.

```
//A -> Array<inteiro> contendo os dados
//p -> índice da posição mais à esquerda considerada
//r -> índice da posição mais à direita considerada
procedimento mergesort(A, p, r)
    se p < r
       q = floor((p+r) / 2) //floor é a função piso (arredondamento para baixo)
       mergesort(A, p, q)
       mergesort(A, q + 1, r)
       merge(A, p, q, r)
```

```
//A -> Array<inteiro> contendo os dados
//p -> índice da posição mais à esquerda considerada
//q -> valor intermediário
//r -> índice da posição mais à direita considerada
procedimento merge(A, p, q, r)
    n1 = p - q + 1
    n2 = r - q
    L = Array<inteiro>[n1 + 1]
    R = Array<inteiro>[n2 + 1]
    para i = 1 até n1
        L[i] = A[p + i - 1]
    para j = 1 até n2
        R[j] = A[q + j]
    L[n1 + 1] = Inf            //Inf é o valor infinito positivo
    R[n2 + 1] = Inf
    i = 1
    j = 1
    para k = p até r
        se L[i] <= R[j]
            A[k] = L[i]
            i = i + 1
        senão
            A[k] = R[j]
            j = j + 1
```

Mergesort

- A seguir será apresentado um exemplo completo da execução do *Mergesort*.

- Todas as posições do vetor a ser ordenado já estão preenchidas.

- A chamada ao procedimento *Mergesort* deve ser feita com valor de *p* igual à primeira posição válida do vetor, e *r* igual à última posição válida do mesmo (ou seja, seu tamanho).

- Exemplo: ordenar a sequência A = {5, 2, 4, 7, 1, 3, 2, 6}.

***Mergesort*(*A*, 1, 8)**

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

***q* = 4**

***Mergesort*(*A*, 1, 4)**

***Mergesort*(*A*, 5, 8)**

***Merge*(*A*, 1, 4, 8)**

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 8)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

***q* = 4**

***Mergesort*(*A*, 1, 4)**

***Mergesort*(*A*, 5, 8)**

***Merge*(*A*, 1, 4, 8)**

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 4)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

***q* = 2**

***Mergesort*(*A*, 1, 2)**

***Mergesort*(*A*, 3, 4)**

***Merge*(*A*, 1, 2, 4)**

Mergesort(A, 1, 4)

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 4)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

q = 2

Mergesort(*A*, 1, 2)

Mergesort(*A*, 3, 4)

Merge(*A*, 1, 2, 4)

Mergesort(A, 1, 4)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 2)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

$q = 1$

***Mergesort*(*A*, 1, 1)**

***Mergesort*(*A*, 2, 2)**

***Merge*(*A*, 1, 2, 4)**

Mergesort(A, 1, 2)

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 2)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

q **= 1**

***Mergesort*(*A*, 1, 1)**

***Mergesort*(*A*, 2, 2)**

***Merge*(*A*, 1, 2, 4)**

Mergesort(A, 1, 2)

↑

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 1)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

***Mergesort*(*A*, 1, 1)**

***Mergesort*(*A*, 1, 2)**

***Mergesort*(*A*, 1, 4)**

***Mergesort*(*A*, 1, 8)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 2)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

q = 1

Mergesort(*A*, 1, 1)

Mergesort(*A*, 2, 2)

Merge(*A*, 1, 2, 4)

Mergesort(A, 1, 2)

↑

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 2, 2)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

***Mergesort*(*A*, 2, 2)**

***Mergesort*(*A*, 1, 2)**

***Mergesort*(*A*, 1, 4)**

***Mergesort*(*A*, 1, 8)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 2)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

q = 1

Mergesort(*A*, 1, 1)

Mergesort(*A*, 2, 2)

Merge(*A*, 1, 1, 2)

Mergesort(*A*, *1*, *2*)

↑

Mergesort(*A*, *1*, *4*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Merge(A, 1, 1, 2)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	5	$+\infty$

	1	2
R	2	$+\infty$

$i = 1$

$j = 1$

$k = 1$

Mergesort(A, 1, 2)

↑

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
  Se $L[i] \leq R[j]$
    $A[k] = L[i]$; i =i +1
  Senão $A[k] = R[j]$
    $j = j + 1$

Merge(A, 1, 1, 2)

1	2	3	4	5	6	7	8
5	2	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	5	+∞

	1	2
R	2	+∞

$i = 1$

$j = 1$

$k = 1$

Mergesort(A, 1, 2)

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Merge(A, 1, 1, 2)

1	2	3	4	5	6	7	8
2	**2**	**4**	**7**	**1**	**3**	**2**	**6**

$n_1 = n_2 = 1$

	1	2
L	**5**	**+∞**

	1	2
R	**2**	**+∞**

$i = 1$

$j = 2$

$k = 1$

Mergesort(A, 1, 2)

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Merge(*A*, 1, 1, 2)

1	2	3	4	5	6	7	8
2	2	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	5	+∞

	1	2
R	2	+∞

$i = 1$

$j = 2$

$k = 2$

***Mergesort*(*A*, 1, 2)**

↑

***Mergesort*(*A*, 1, 4)**

↑

***Mergesort*(*A*, 1, 8)**

↑

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Merge(A, 1, 1, 2)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	5	+∞

	1	2
R	2	+∞

$i = 2$

$j = 2$

$k = 2$

Mergesort(A, 1, 2)

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Mergesort(*A*, 1, 2)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

***q* = 1**

***Mergesort*(*A*, 1, 1)**

***Mergesort*(*A*, 2, 2)**

***Merge*(*A*, 1, 1, 2)**

***Mergesort*(*A*, *1, 2*)**

***Mergesort*(*A*, *1, 4*)**

***Mergesort*(*A*, *1, 8*)**

Pilha das Chamadas Recursivas

Como o *Mergesort*(*A*, 1, 2) já executou todos os comandos, saída da recursão

Mergesort(*A*, 1, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

q = 2

Mergesort(*A*, 1, 2)

Mergesort(*A*, 3, 4)

Merge(*A*, 1, 2, 4)

Mergesort(A, 1, 4)

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Mergesort(*A*, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

***q* = 3**

***Mergesort*(*A*, 3, 3)**

***Mergesort*(*A*, 4, 4)**

***Merge*(*A*, 3, 3, 4)**

Mergesort(A, 3, 4)

↑

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

q = 3

Mergesort(*A*, 3, 3)

Mergesort(*A*, 4, 4)

Merge(*A*, 3, 3, 4)

Mergesort(*A*, *3*, *4*)

↑

Mergesort(*A*, *1*, *4*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 3, 3)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

***Mergesort*(*A*, 3, 3)**

***Mergesort*(*A*, 3, 4)**

***Mergesort*(*A*, 1, 4)**

***Mergesort*(*A*, 1, 8)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

q = 3

Mergesort(*A*, 3, 3)

Mergesort(*A*, 4, 4)

Merge(*A*, 3, 3, 4)

Mergesort(*A*, *3*, *4*)

↑

Mergesort(*A*, *1*, *4*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Mergesort(A, 4, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

Mergesort(A, 4, 4)

Mergesort(A, 3, 4)

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Mergesort(*A*, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

q = 3

Mergesort(*A*, 3, 3)

Mergesort(*A*, 4, 4)

Merge(*A*, 3, 3, 4)

Mergesort(*A*, *3*, *4*)

Mergesort(*A*, *1*, *4*)

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Merge(*A*, 3, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	4	+∞

	1	2
R	7	+∞

$i = 1$

$j = 1$

$k = 3$

Mergesort(A, 3, 4)

↑

***Mergesort*(*A*, 1, 4)**

↑

***Mergesort*(*A*, 1, 8)**

↑

Pilha das Chamadas Recursivas

Para *k* = *p* até *r* **então**
    Se L[*i*] ≤ R[*j*]
        A[*k*] = L[*i*]; i =i +1
    Senão A[*k*] = R[*j*]
        j = *j* + 1

Merge(A, 3, 3, 4)

1	2	3	4	5	6	7	8
2	**5**	**4**	**7**	**1**	**3**	**2**	**6**

$n_1 = n_2 = 1$

	1	2
L	**4**	**+∞**

	1	2
R	**7**	**+∞**

$i = 1$

$j = 1$

$k = 3$

Mergesort(A, 3, 4)

↑

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Merge(A, 3, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	4	$+\infty$

	1	2
R	7	$+\infty$

$i = 2$

$j = 1$

$k = 3$

Mergesort(A, 3, 4)

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Merge(A, 3, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	4	+∞

	1	2
R	7	+∞

$i = 2$

$j = 1$

$k = 4$

***Mergesort(A*, 3, 4)**

↑

***Mergesort(A*, 1, 4)**

↑

***Mergesort(A*, 1, 8)**

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(*A*, 3, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	4	$+\infty$

	1	2
R	7	$+\infty$

$i = 2$

$j = 2$

$k = 4$

***Mergesort*(*A*, 3, 4)**

↑

***Mergesort*(*A*, 1, 4)**

↑

***Mergesort*(*A*, 1, 8)**

↑

Pilha das Chamadas Recursivas

```
Para k = p até r então
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        j = j + 1
```

Mergesort(*A*, 3, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

q = 3

Mergesort(*A*, 3, 3)

Mergesort(*A*, 4, 4)

Merge(*A*, 3, 3, 4)

Mergesort(A, 3, 4)

Mergesort(*A, 1, 4)*

Mergesort(*A, 1, 8*)

Pilha das Chamadas Recursivas

Como o *Mergesort*(*A*, 3, 4) já executou todos os comandos, saída da recursão

Mergesort(*A*, 1, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

q = 2

Mergesort(*A*, 1, 2)

Mergesort(*A*, 3, 4)

Merge(*A*, 1, 2, 4)

Mergesort(*A*, *1*, *4*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	2	5	+∞

	1	2	3
R	4	7	+∞

$i = 1$

$j = 1$

$k = 1$

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[*i*] ≤ R[*j*]
        A[*k*] = L[*i*]; i =i +1
    Senão A[*k*] = R[*j*]
        j = *j* + 1

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	**5**	**4**	**7**	**1**	**3**	**2**	**6**

$n_1 = n_2 = 2$

	1	2	3
L	**2**	**5**	**+∞**

	1	2	3
R	**4**	**7**	**+∞**

$i = 1$

$j = 1$

$k = 1$

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	**5**	**4**	**7**	**1**	**3**	**2**	**6**

$n_1 = n_2 = 2$

	1	2	3
L	**2**	**5**	**+∞**

	1	2	3
R	**4**	**7**	**+∞**

$i = 2$

$j = 1$

$k = 1$

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	2	5	+∞

	1	2	3
R	4	7	+∞

$i = 2$

$j = 1$

$k = 2$

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	4	4	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	2	5	+∞

	1	2	3
R	4	7	+∞

$i = 2$

$j = 2$

$k = 2$

Mergesort(A, 1, 4)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	4	4	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	2	5	$+\infty$

	1	2	3
R	4	7	$+\infty$

***i* = 2**

***j* = 2**

***k* = 3**

Mergesort(A, 1, 4)

↑

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	2	5	+∞

	1	2	3
R	4	7	+∞

$i = 3$

$j = 2$

$k = 3$

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

```
Para k = p até r então
   Se L[i] ≤ R[j]
      A[k] = L[i]; i =i +1
   Senão A[k] = R[j]
      j = j + 1
```

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	2	5	+∞

	1	2	3
R	4	7	+∞

$i = 3$

$j = 2$

$k = 4$

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
  Se L[i] ≤ R[j]
    A[k] = L[i]; i =i +1
  Senão A[k] = R[j]
    $j = j + 1$

Merge(A, 1, 2, 4)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	2	5	+∞

	1	2	3
R	4	7	+∞

$i = 3$

$j = 3$

$k = 4$

Mergesort(A, 1, 4)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Mergesort(*A*, 1, 4)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 2

Mergesort(*A*, 1, 2)

Mergesort(*A*, 3, 4)

Merge(*A*, 1, 2, 4)

Mergesort(A, 1, 4)

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Como o *Mergesort*(*A*, 1, 4) já executou todos os comandos, saída da recursão

Mergesort(*A*, 1, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 4

Mergesort(*A*, 1, 4)

Mergesort(*A*, 5, 8)

Merge(*A*, 1, 4, 8)

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Mergesort(*A*, 5, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

***q* = 6**

***Mergesort*(*A*, 5, 6)**

***Mergesort*(*A*, 7, 8)**

***Merge*(*A*, 5, 6, 8)**

Mergesort(A, 5, 8)

↑

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 5, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 6

Mergesort(*A*, 5, 6)

Mergesort(*A*, 7, 8)

Merge(*A*, 5, 6, 8)

Mergesort(A, 5, 8)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 5, 6)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 5

Mergesort(*A*, 5, 5)

Mergesort(*A*, 6, 6)

Merge(*A*, 5, 5, 6)

Mergesort(*A*, 5, 6)

↑

Mergesort(*A*, 5, 8)

↑

Mergesort(*A*, 1, 8)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 5, 6)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 5

Mergesort(*A*, 5, 5)

Mergesort(*A*, 6, 6)

Merge(*A*, 5, 5, 6)

Mergesort(*A*, *5*, *6*)

↑

Mergesort(*A*, *5*, *8*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 5, 5)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

***Mergesort*(*A*, 5, 5)**

***Mergesort*(*A*, 5, 6)**

***Mergesort*(*A*, 5, 8)**

***Mergesort*(*A*, 1, 8)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 5, 6)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 5

Mergesort(*A*, 5, 5)

Mergesort(*A*, 6, 6)

Merge(*A*, 5, 5, 6)

Mergesort(*A*, *5, 6)*

↑

Mergesort(*A*, *5, 8)*

↑

Mergesort(*A*, *1, 8*)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 6, 6)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

***Mergesort*(*A*, 6, 6)**

***Mergesort*(*A*, 5, 6)**

***Mergesort*(*A*, 5, 8)**

***Mergesort*(*A*, 1, 8)**

Pilha das Chamadas Recursivas

Mergesort(*A*, 5, 6)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 5

Mergesort(*A*, 5, 5)

Mergesort(*A*, 6, 6)

Merge(*A*, 5, 5, 6)

Mergesort(*A*, *5*, *6*)

↑

Mergesort(*A*, *5*, *8*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Merge(A, 5, 5, 6)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	1	+∞

	1	2
R	3	+∞

$i = 1$

$j = 1$

$k = 5$

***Mergesort(A*, 5, 6)**

***Mergesort(A*, 5, 8)**

***Mergesort(A*, 1, 8)**

Pilha das Chamadas Recursivas

Para *k* = *p* até *r* **então**
    Se L[*i*] ≤ R[*j*]
        A[*k*] = L[*i*]; i =i +1
    Senão A[*k*] = R[*j*]
        j = *j* + 1

Merge(A, 5, 5, 6)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	1	+∞

	1	2
R	3	+∞

$i = 1$

$j = 1$

$k = 5$

Mergesort(A, 5, 6)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Merge(A, 5, 5, 6)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	1	+∞

	1	2
R	3	+∞

$i = 2$

$j = 1$

$k = 5$

Mergesort(A, 5, 6)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

```
Para k = p até r então
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        j = j + 1
```

Merge(A, 5, 5, 6)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	1	$+\infty$

	1	2
R	3	$+\infty$

$i = 2$

$j = 1$

$k = 6$

Mergesort(A, 5, 6)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Merge(*A*, 5, 5, 6)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	1	$+\infty$

	1	2
R	3	$+\infty$

$i = 2$

$j = 2$

$k = 6$

***Mergesort*(*A*, 5, 6)**

***Mergesort*(*A*, 5, 8)**

***Mergesort*(*A*, 1, 8)**

Pilha das Chamadas Recursivas

```
Para k = p até r então
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        j = j + 1
```

Mergesort(*A*, 5, 6)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 5

Mergesort(*A*, 5, 5)

Mergesort(*A*, 6, 6)

Merge(*A*, 5, 5, 6)

Mergesort(*A*, *5*, *6*)

Mergesort(*A*, *5*, *8*)

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Como o *Mergesort*(*A*, 5, 6) já executou todos os comandos, saída da recursão

Mergesort(*A*, 5, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 6

Mergesort(*A*, 5, 6)

Mergesort(*A*, 7, 8)

Merge(*A*, 5, 6, 8)

Mergesort(*A*, *5*, *8*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 7, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 7

***Mergesort*(*A*, 7, 7)**

***Mergesort*(*A*, 8, 8)**

***Merge*(*A*, 7, 7, 8)**

***Mergesort*(*A*, 7, 8)**

↑

***Mergesort*(*A*, 5, 8)**

↑

***Mergesort*(*A*, 1, 8)**

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 7, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

***q* = 7**

***Mergesort*(*A*, 7, 7)**

***Mergesort*(*A*, 8, 8)**

***Merge*(*A*, 7, 7, 8)**

Mergesort*(*A*, *7, 8)

↑

Mergesort*(*A*, *5, 8)

↑

***Mergesort*(*A*, *1, 8*)**

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 7, 7)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

Mergesort(A, 7, 7)

Mergesort(A, 7, 8)

Mergesort(A, 5, 8)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Mergesort(*A*, 7, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

$q = 7$

Mergesort(*A*, 7, 7)

Mergesort(*A*, 8, 8)

Merge(*A*, 7, 7, 8)

Mergesort(*A*, *7*, *8*)

↑

Mergesort(*A*, *5*, *8*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 8, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

Como p < r não é verdadeiro, saída da recursão

Mergesort(A, 8, 8)

Mergesort(A, 7, 8)

Mergesort(A, 5, 8)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Mergesort(*A*, 7, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 7

Mergesort(*A*, 7, 7)

Mergesort(*A*, 8, 8)

Merge(*A*, 7, 7, 8)

Mergesort(A, 7, 8)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Merge(A, 7, 7, 8)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	2	$+\infty$

	1	2
R	6	$+\infty$

$i = 1$

$j = 1$

$k = 7$

Mergesort(A, 7, 8)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

```
Para k = p até r então
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        j = j + 1
```

Merge(A, 7, 7, 8)

1	2	3	4	5	6	7	8
2	**3**	**5**	**7**	**1**	**3**	**2**	**6**

$n_1 = n_2 = 1$

	1	2
L	**2**	**+∞**

	1	2
R	**6**	**+∞**

***i* = 1**

***j* = 1**

***k* = 7**

Mergesort(A, 7, 8)

Mergesort(A, 5, 8)

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Merge(A, 7, 7, 8)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	2	+∞

	1	2
R	6	+∞

$i = 2$

$j = 1$

$k = 7$

Mergesort(A, 7, 8)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(A, 7, 7, 8)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	2	$+\infty$

	1	2
R	6	$+\infty$

$i = 2$

$j = 1$

$k = 8$

Mergesort(A, 7, 8)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(A, 7, 7, 8)

1	2	3	4	5	6	7	8
2	3	5	7	1	3	2	6

$n_1 = n_2 = 1$

	1	2
L	2	$+\infty$

	1	2
R	6	$+\infty$

$i = 2$

$j = 2$

$k = 8$

Mergesort(A, 7, 8)

↑

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Mergesort(*A*, 7, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 7

Mergesort(*A*, 7, 7)

Mergesort(*A*, 8, 8)

Merge(*A*, 7, 7, 8)

Mergesort(*A*, *7*, *8*)

↑

Mergesort(*A*, *5*, *8*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

Como o *Mergesort*(*A*, 7, 8) já executou todos os comandos, saída da recursão

Mergesort(*A*, 5, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	3	2	6

q = 6

Mergesort(*A*, 5, 6)

Mergesort(*A*, 7, 8)

Merge(*A*, 5, 6, 8)

***Mergesort*(*A*, *5*, *8*)**

↑

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Merge(*A*, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	1	3	+∞

	1	2	3
R	2	6	+∞

$i = 1$

$j = 1$

$k = 5$

Mergesort(*A*, *5*, *8*)

↑

Mergesort(*A*, *1*, *8*)

↑

Pilha das Chamadas Recursivas

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Merge(*A*, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	1	3	+∞

	1	2	3
R	2	6	+∞

$i = 1$

$j = 1$

$k = 5$

***Mergesort*(*A*, 5, 8)**

↑

***Mergesort*(*A*, 1, 8)**

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(A, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

$n_1 = n_2 = 2$

	1	2	3
L	1	3	+∞

	1	2	3
R	2	6	+∞

$i = 2$

$j = 1$

$k = 5$

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Merge(A, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	3	2	6

***n_1 = n_2* = 2**

	1	2	3
L	1	3	+∞

	1	2	3
R	2	6	+∞

***i* = 2**

***j* = 1**

***k* = 6**

Mergesort(A, 5, 8)

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Merge(*A*, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	2	2	6

$n_1 = n_2 = 2$

L

1	2	3
1	3	$+\infty$

R

1	2	3
2	6	$+\infty$

$i = 2$

$j = 2$

$k = 6$

Mergesort(*A*, *5*, *8*)

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Merge(A, 5, 6, 8)

1	2	3	4	5	6	7	8
2	**5**	**4**	**7**	**1**	**2**	**2**	**6**

$n_1 = n_2 = 2$

	1	2	3
L	**1**	**3**	**+∞**

	1	2	3
R	**2**	**6**	**+∞**

$i = 2$

$j = 2$

$k = 7$

Mergesort(A, 5, 8)

↑

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Para *k* = *p* até *r* **então**
    Se L[*i*] ≤ R[*j*]
        A[*k*] = L[*i*]; i =i +1
    Senão A[*k*] = R[*j*]
        j = *j* + 1

Merge(*A*, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	2	3	6

$n_1 = n_2 = 2$

	1	2	3
L	1	3	+∞

	1	2	3
R	2	6	+∞

***i* = 3**

***j* = 2**

***k* = 7**

***Mergesort*(*A*, *5*, *8*)**

↑

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

```
Para k = p até r então
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        j = j + 1
```

Merge(A, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	2	3	6

$n_1 = n_2 = 2$

L

1	2	3
1	3	+∞

R

1	2	3
2	6	+∞

***i* = 3**

***j* = 2**

***k* = 8**

Mergesort(A, 5, 8)

↑

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Merge(A, 5, 6, 8)

1	2	3	4	5	6	7	8
2	5	4	7	1	2	3	6

$n_1 = n_2 = 2$

	1	2	3
L	1	3	+∞

	1	2	3
R	2	6	+∞

***i* = 3**

***j* = 3**

***k* = 8**

Mergesort(A, 5, 8)

↑

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Mergesort(*A*, 5, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	2	3	6

q = 6

Mergesort(*A*, 5, 6)

Mergesort(*A*, 7, 8)

Merge(*A*, 5, 6, 8)

Mergesort(*A*, *5*, *8*)

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Como o *Mergesort*(*A*, 5, 8) já executou todos os comandos, saída da recursão

Mergesort(*A*, 1, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	2	3	6

***q* = 4**

***Mergesort*(*A*, 1, 4)**

***Mergesort*(*A*, 5, 8)**

***Merge*(*A*, 1, 4, 8)**

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 1$

$j = 1$

$k = 1$

```
Para k = p até r então
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        j = j + 1
```

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	**2**	4	5	7	+∞

	1	2	3	4	5
R	**1**	2	3	6	+∞

***i* = 1**

***j* = 1**

***k* = 1**

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Merge(A, 1, 4, 8)

1	2	3	4	5	6	7	8
1	4	5	7	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 1$

$j = 2$

$k = 1$

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	4	5	7	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	$+\infty$

	1	2	3	4	5
R	1	2	3	6	$+\infty$

***i* = 1**

***j* = 2**

***k* = 2**

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

Merge(A, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	5	7	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

***i* = 2**

***j* = 2**

***k* = 2**

Para *k* = *p* até *r* **então**

Se L[*i*] ≤ R[*j*]

A[*k*] = L[*i*]; i =i +1

Senão A[*k*] = R[*j*]

j = *j* + 1

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	5	7	1	2	3	6

$n_1 = n_2 = 4$

L

1	2	3	4	5
2	4	5	7	$+\infty$

R

1	2	3	4	5
1	2	3	6	$+\infty$

$i = 2$

$j = 2$

$k = 3$

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	7	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 2$

$j = 3$

$k = 3$

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Merge(A, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	7	1	2	3	6

$n_1 = n_2 = 4$

L

1	2	3	4	5
2	4	5	7	+∞

R

1	2	3	4	5
1	2	3	6	+∞

$i = 2$

$j = 3$

$k = 4$

Para *k* = *p* até *r* **então**
    Se L[*i*] ≤ R[*j*]
        A[*k*] = L[*i*]; i =i +1
    Senão A[*k*] = R[*j*]
        j = *j* + 1

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Merge(A, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 2$

$j = 4$

$k = 4$

Para $k = p$ até r **então**
  Se L[i] ≤ R[j]
    A[k] = L[i]; i =i +1
  Senão A[k] = R[j]
    $j = j + 1$

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	1	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 2$

$j = 4$

$k = 5$

Para $k = p$ até r **então**
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        $j = j + 1$

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Merge(A, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	4	2	3	6

$n_1 = n_2 = 4$

L

1	2	3	4	5
2	4	5	7	+∞

R

1	2	3	4	5
1	2	3	6	+∞

$i = 3$

$j = 4$

$k = 5$

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Mergesort(A, 1, 8)

↑

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	4	2	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

***i* = 3**

***j* = 4**

***k* = 6**

Para *k* = *p* até *r* **então**
    Se L[*i*] ≤ R[*j*]
        A[*k*] = L[*i*]; i =i +1
    Senão A[*k*] = R[*j*]
        j = *j* + 1

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	4	**5**	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	**5**	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

i = 4
j = 4
k = 6

1. **Para** *k* = *p* até *r* **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

Mergesort(*A*, *1*, *8*)

Pilha das Chamadas Recursivas

Merge(A, 1, 4, 8)

1	2	2	3	4	5	3	6
1	**2**	**2**	**3**	**4**	**5**	**3**	**6**

$n_1 = n_2 = 4$

L

1	2	3	4	5
2	**4**	**5**	**7**	**+∞**

R

1	2	3	4	5
1	**2**	**3**	**6**	**+∞**

***i* = 4**

***j* = 4**

***k* = 7**

```
Para k = p até r então
    Se L[i] ≤ R[j]
        A[k] = L[i]; i =i +1
    Senão A[k] = R[j]
        j = j + 1
```

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Merge(A, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	4	5	6	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 4$

$j = 5$

$k = 7$

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

Mergesort(A, 1, 8)

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	4	5	3	6

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 4$

$j = 5$

$k = 8$

1. **Para** $k = p$ até r **então**
2. **Se** L[i] ≤ R[j]
3. A[k] = L[i]; i =i +1
4. **Senão** A[k] = R[j]
5. $j = j + 1$

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

Merge(*A*, 1, 4, 8)

1	2	3	4	5	6	7	8
1	2	2	3	4	5	6	7

$n_1 = n_2 = 4$

	1	2	3	4	5
L	2	4	5	7	+∞

	1	2	3	4	5
R	1	2	3	6	+∞

$i = 5$

$j = 5$

$k = 8$

1. **Para** $k = p$ até r **então**
2. **Se** L[*i*] ≤ R[*j*]
3. A[*k*] = L[*i*]; i =i +1
4. **Senão** A[*k*] = R[*j*]
5. *j* = *j* + 1

***Mergesort*(*A*, *1*, *8*)**

↑

Pilha das Chamadas Recursivas

Mergesort(*A*, 1, 8)

1	2	3	4	5	6	7	8
2	4	5	7	1	2	3	6

***q* = 4**

***Mergesort*(*A*, 1, 4)**

***Mergesort*(*A*, 5, 8)**

***Merge*(*A*, 1, 4, 8)**

Como o *Mergesort*(*A*, 1, 8) já executou todos os comandos, saída da recursão

***Mergesort*(*A*, *1*, *8*)**

Pilha das Chamadas Recursivas

FIM!

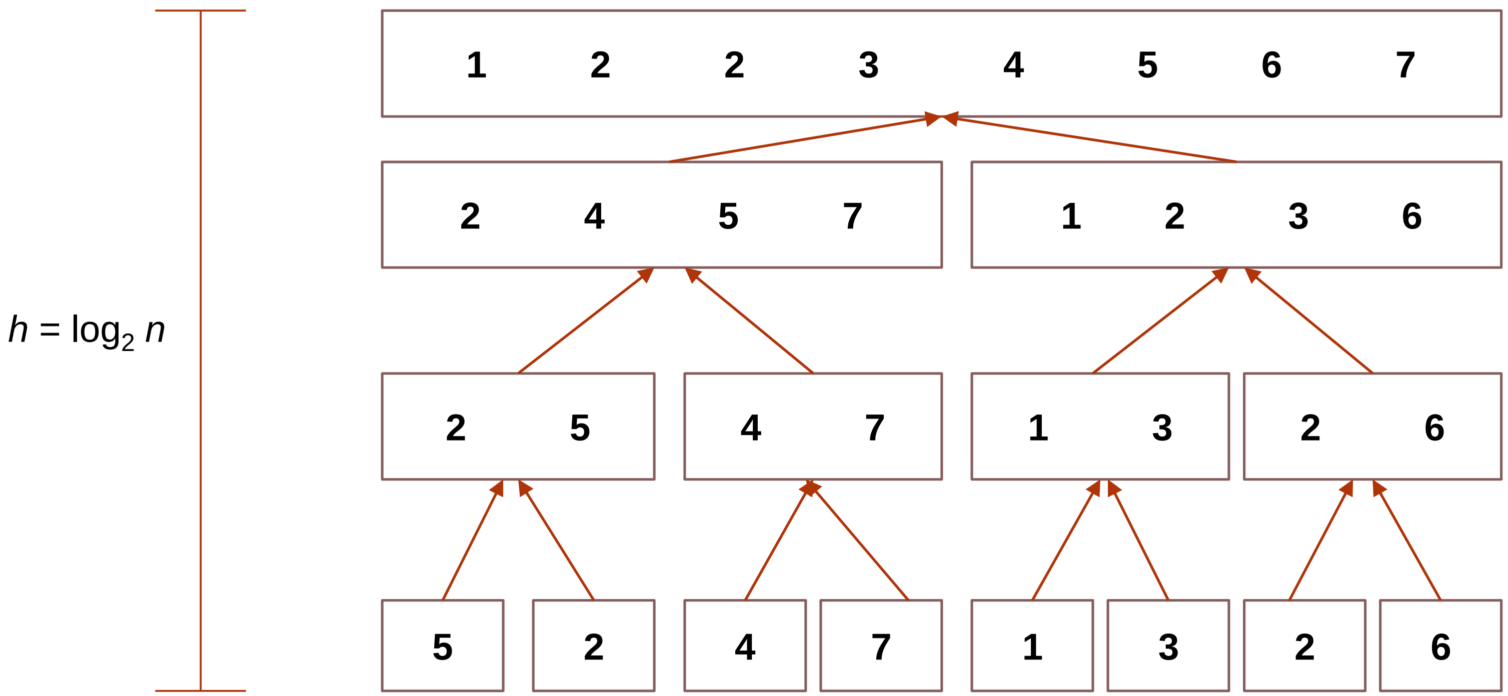
Mergesort

- Procedimento *Merge*:

  - Considerar que linhas 1-2 tenham custo O(1);
  - Linhas 3-6: $O(n)$;
  - Linhas 7-8: O(1);
  - Linha 9: $O(n)$;
  - Linhas 10-13: $nO(1)=O(n)$.

- O custo final do procedimento *Merge* vai ser de: $T(n)=2O(1)+4O(n)+2O(1)+O(n)+4O(n)=O(n)$.

Mergesort



Mergesort

- O custo final do *Mergesort* será então:

$$\Theta(n\log_2 n)$$

Referências

- CORMEN, H. T.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introduction to Algorithms, 3rd ed., *Boston: MIT Press*, 2009.

- FEOFILOFF, Paulo. Algoritmos em Linguagem C. Editora Campus/Elsevier, 2009.

Mergesort

Algoritmos e Estruturas de Dados
Prof. Dr. Luciano Demétrio Santos Pacífico
{luciano.pacifico@ufrpe.br}

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO