Ordenação

Algoritmos e Estrutura de Dados - IF672

João Nascimento <jvsn2>
Mateus Freire <mfvd>

Centro de Informática UFPE

Universidade Federal de Pernambuco

Sorting

- Algoritmos muito usados em toda ciência da computação;
- Usos práticos em diversos setores, como bancos;
- Diferentes tipos e complexidades;

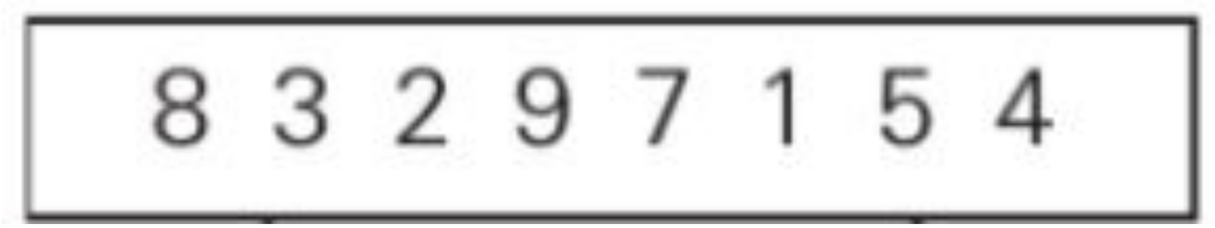
Sorting - Complexidade

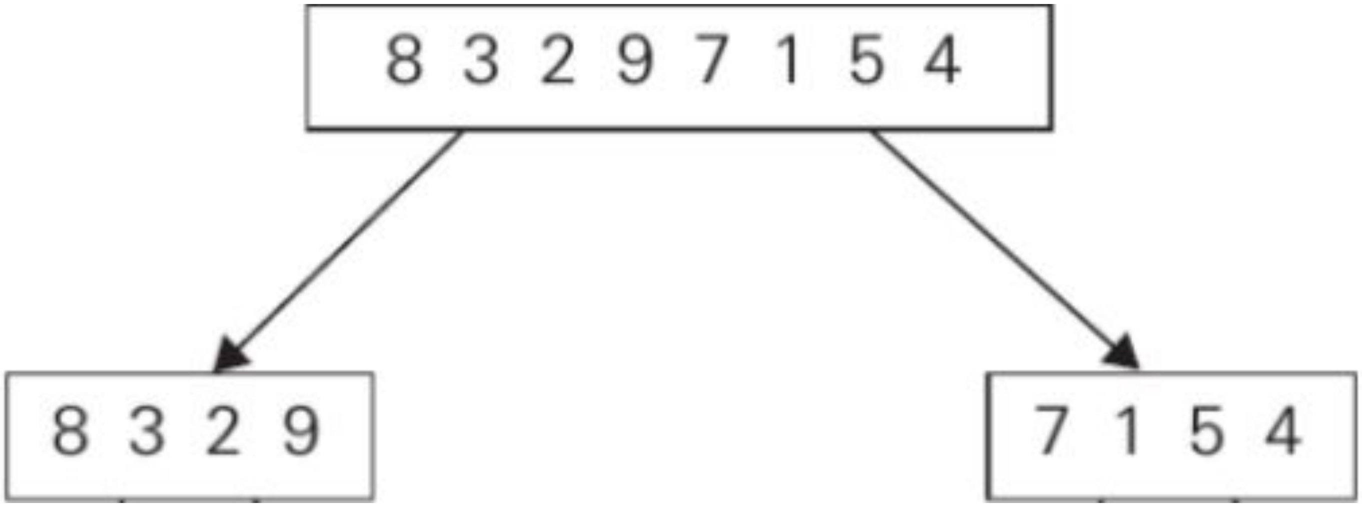
Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

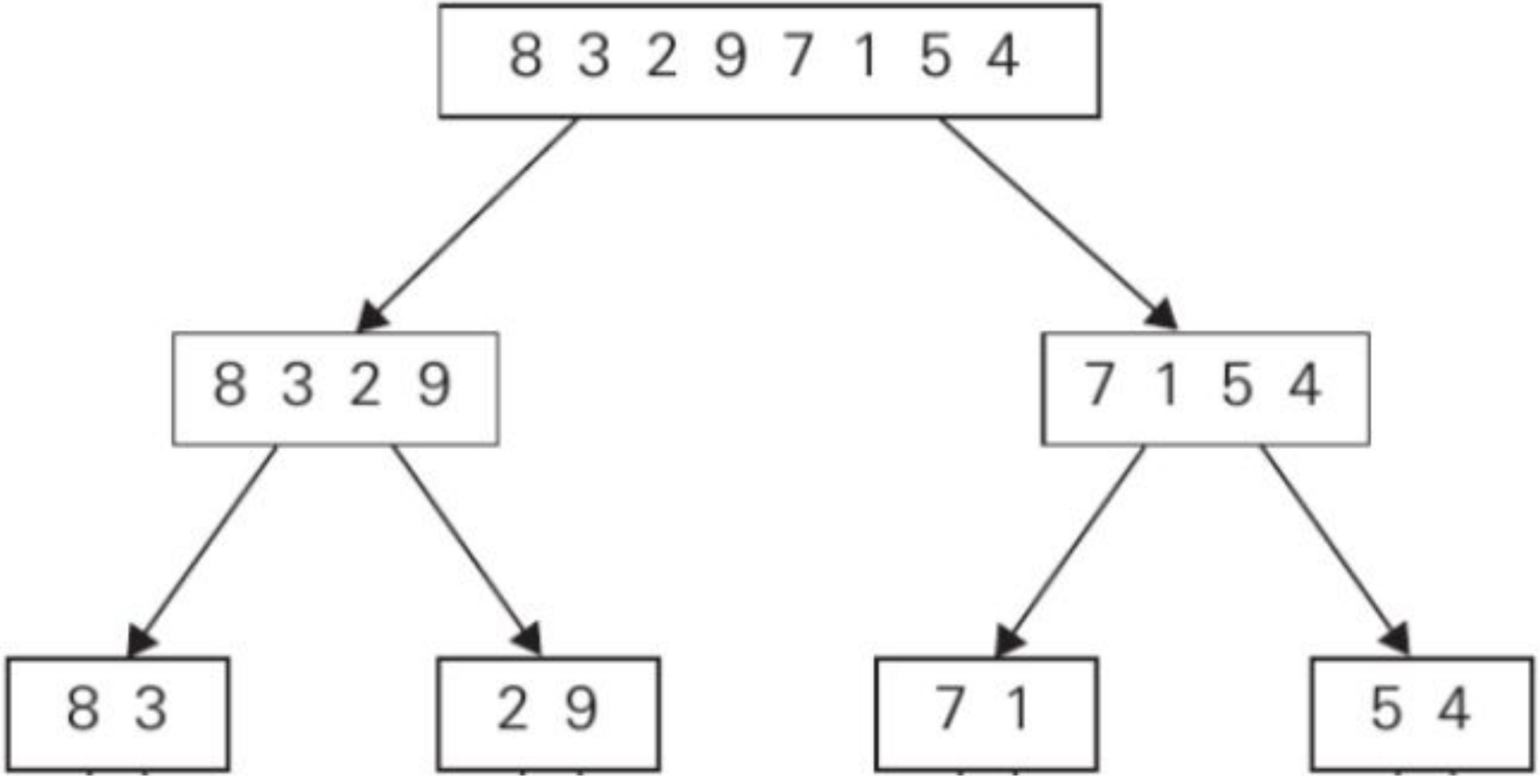
Mergesort

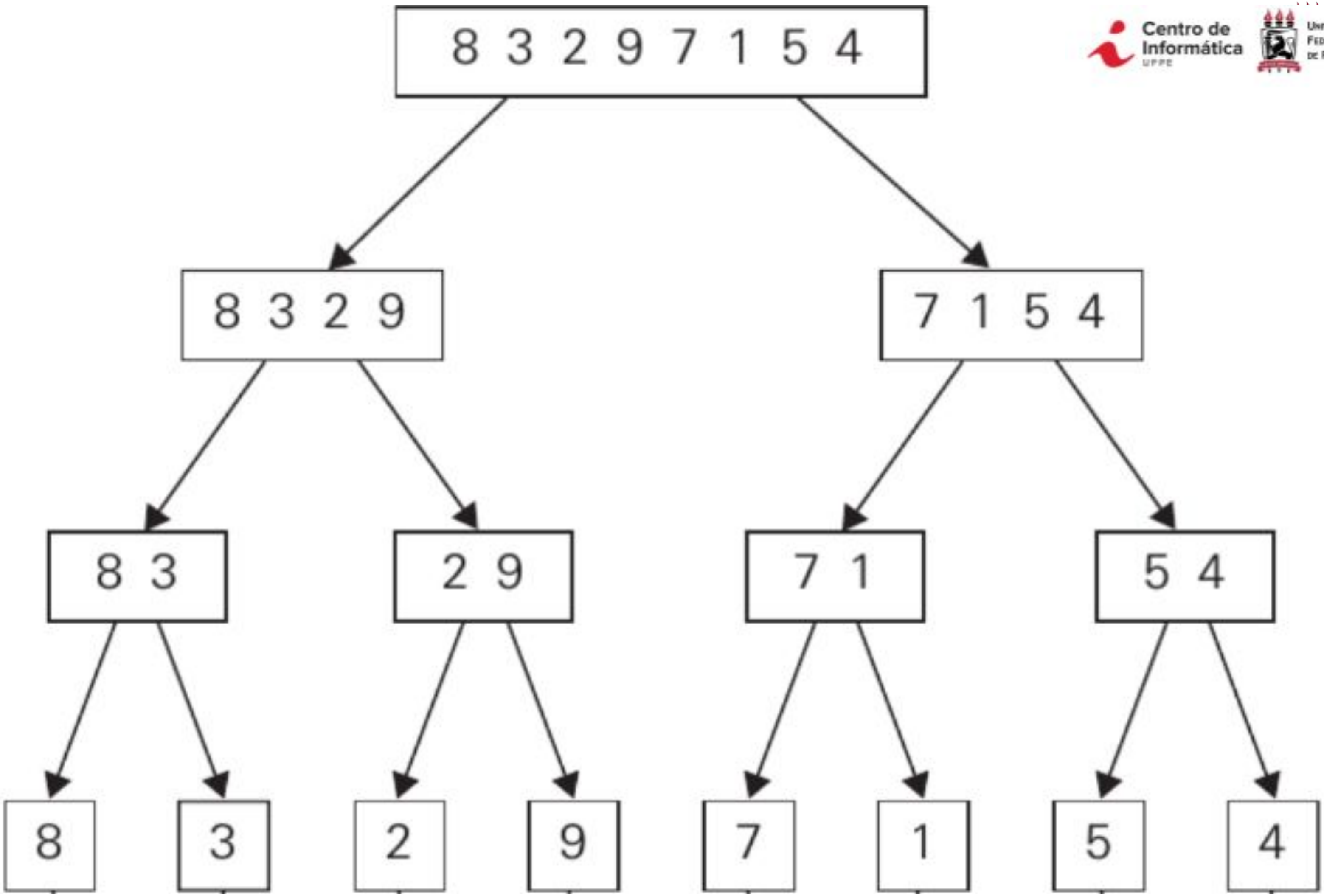
- **Divide and conquer:**
  - A problem is divided into several subproblems of the same type (ideally of about equal size)
  - The subproblems are solved
  - If necessary, the solutions to the subproblems are combined
  - Uses more space
  - Stable algorithm

In practice

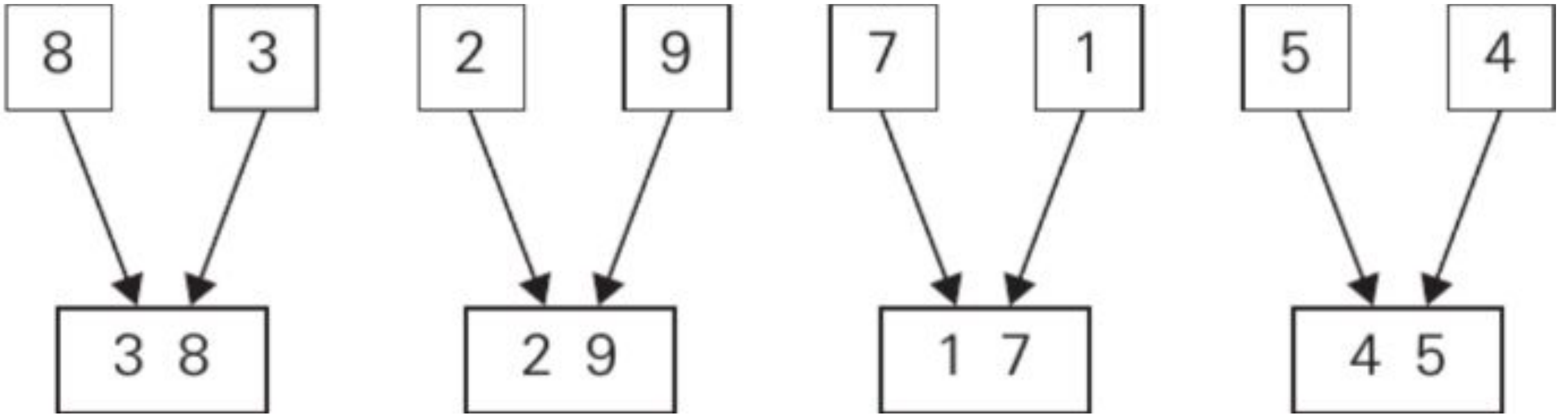


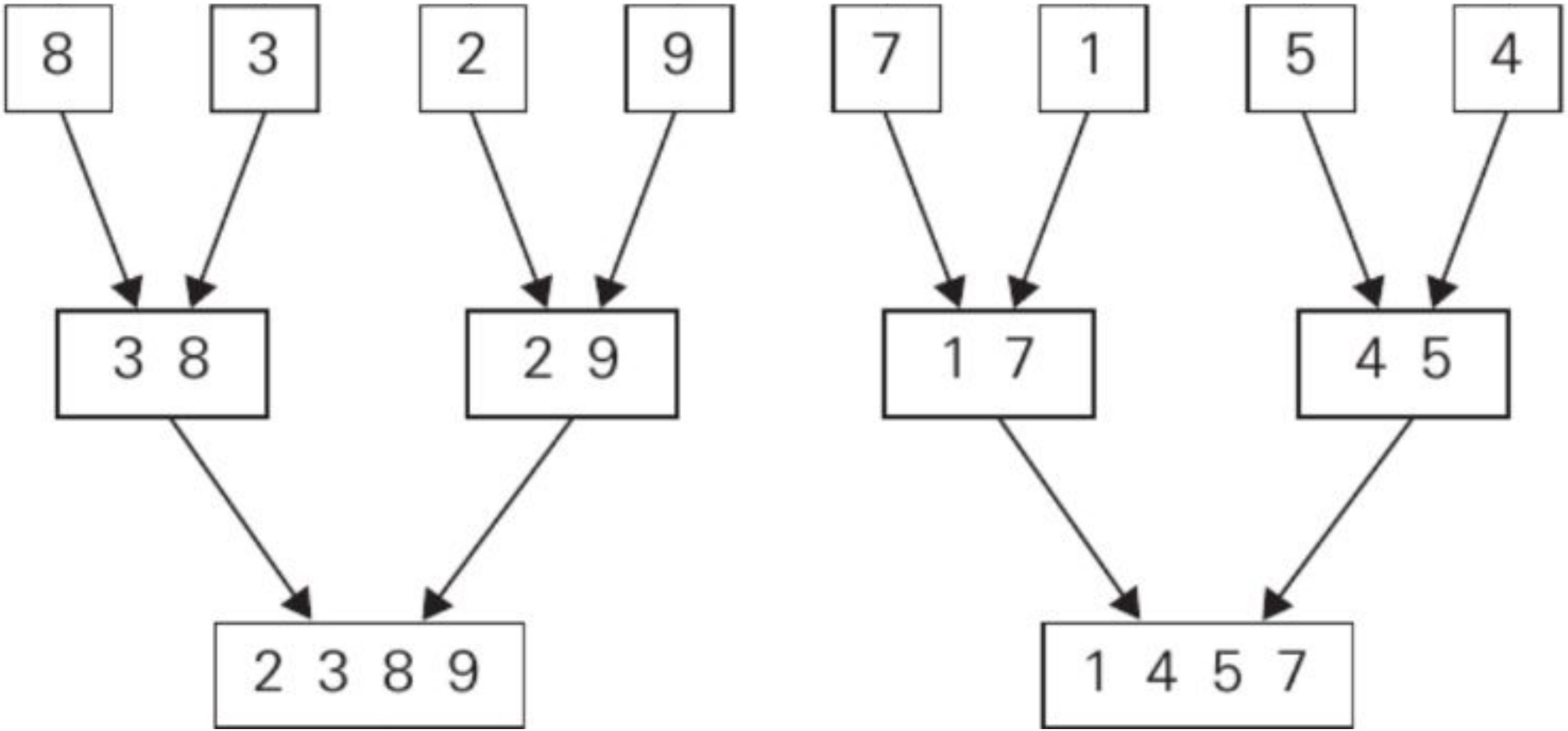

8 3 2 9 7 1 5 4
8 3 2 9
7 1 5 4

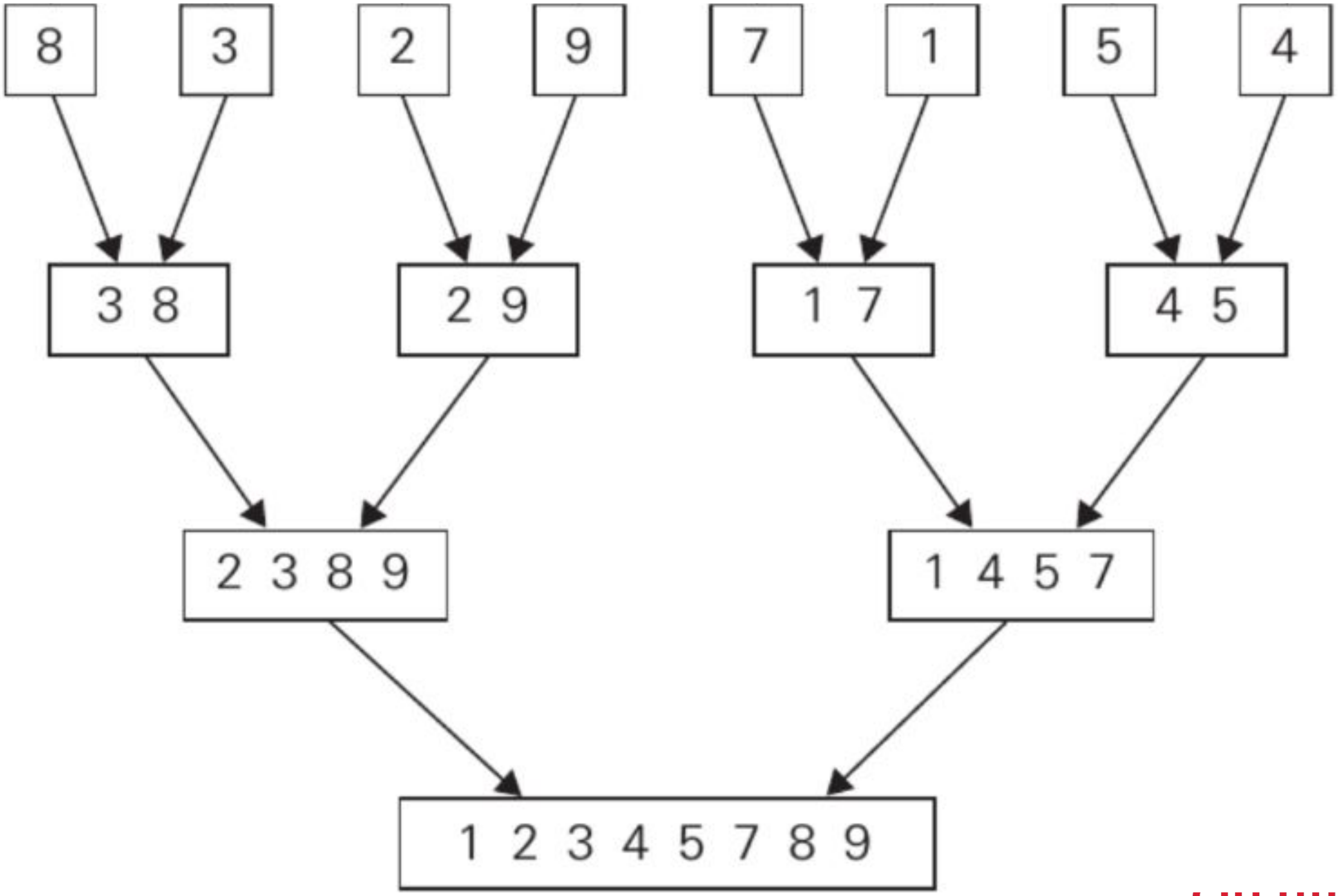

8 3 2 9 7 1 5 4
8 3 2 9
7 1 5 4
8 3
2 9
7 1
5 4


8 3 2 9 7 1 5 4
8 3 2 9
7 1 5 4
8 3
2 9
7 1
5 4
8
3
2
9
7
1
5
4

8	3	2	9	7	1	5	4


8
3
2
9
7
1
5
4
3 8
2 9
1 7
4 5


8
3
2
9
7
1
5
4
3 8
2 9
1 7
4 5
2 3 8 9
1 4 5 7

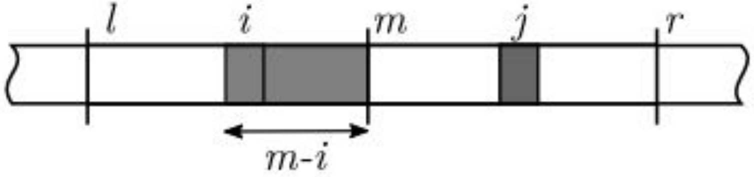

8 3 2 9 7 1 5 4
3 8
2 9
1 7
4 5
2 3 8 9
1 4 5 7
1 2 3 4 5 7 8 9

1	2	3	4	5	7	8	9

■ QUESTÃO 3 (2,0pt)

Num vetor de inteiros $V = (v_0, \ldots, v_{n-1})$, temos uma ***inversão*** quando um valor maior está à esquerda de um valor menor, isto é quando um par (i, j) é tal que $i < j$ e $v_i > v_j$. Podemos contar a quantidade de inversões de um array com uma variação do ***Mergesort***, como explicado a seguir.

A função ***merge*** combina dois segmentos adjacentes ordenados $V[l : m] = (v_l, \ldots, v_{m-1})$ e $V[m : r] = (v_m, \ldots, v_{r-1})$. Se, ao comparar um elemento v_i do primeiro segmento com um elemento v_j do segundo segmento, tivermos $v_j < v_i$, então v_j está invertido com relação a v_i e todos os demais elementos do primeiro segmento à direita da posição i, ou seja, devemos contabilizar mais $m - i$ inversões.



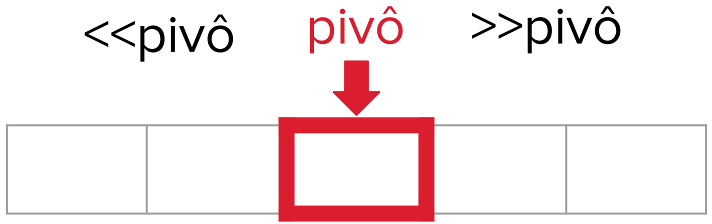
Ilustre a execução deste ***Mergesort*** modificado sobre o vetor de entrada

$$V = (4, 0, 6, 2, 7, 3, 1, 5)$$

exibindo o vetor V imediatamente após cada execução da função ***merge***, juntamente com o número de inversões contabilizadas até então.

Quicksort (PT-BR)

- **Funcionamento básico:**
  - Seleciona um pivô: elementos (chaves) a esquerda vão ser menores que o pivô, elementos a direita maiores
  - Particiona o array pelo pivô, a esquerda vão ser elementos menores que o pivô e a direita maiores;
  - Ordena recursivamente os “sub”vetores



Quicksort (PT-BR)

- **Características:**
  - Também depende da partição de um vetor;
    - Dividir e conquistar
  - Usa recursão;
  - A escolha da partição afeta o desempenho;
  - Não utiliza espaço adicional como o merge;
  - Algoritmo *não-estável;*

Quicksort (PT-BR)

- **Características:**
  - Melhor caso (e na média): O(n * log n)
    - Pivô balanceado = mediana do vetor
  - Pior caso: $O(n^2)$:
    - Partições nos extremos = desbalanceadas;
    - Vetor ordenado;

Quicksort (PT-BR)

- Exemplo: Utilizar o quicksort para ordenar o vetor:

V = [4, 3, 5, 7, 2, 1, 8, 6]

Escolhendo V[0] = 4 como pivô;

Escolhendo pivô como V[(length(V)/2)-1] = 7

Quicksort

Questão:

O Professor Saulo Doce propôs o seguinte algoritmo enigma:

```
Algoritmo enigma
Entrada head: ptr p/ cabeça de lista com
              sentinela
              tail: ptr p/ último elto. da lista
Saída   ???
se head = tail or head→next = tail então
    devolva
fim se
hp ← func (head, tail)
enigma (head, hp)
enigma (hp→next, tail)
fim
```

```
Algoritmo func
Entrada head, tail
Saída   ???
pv ← tail→val
lp ← head
cur ← head
enquanto cur→next ≠ tail faça
    se cur→next→val < pv então
        tmp ← lp→next→val
        lp→next→val ← cur→next→val
        cur→next→val ← tmp
        lp ← lp→next
    fim se
    cur ← cur→next
fim faça
tail→val ← lp→next→val
lp→next→val ← pv
devolva lp
fim
```

Quicksort

Questão:

Após o algoritmo enigma do Saulo Doce responda:

a) Ilustre a execução do algoritmo sobre a lista

$$\backslash \to 6 \to 8 \to 1 \to 3 \to 7 \to 5 \to 2 \to 4$$

exibindo a lista após cada chamada à função *func* (linha 4 do Alg. *enigma*)

▷ *0,5 pt*

b) O que o algoritmo faz? (máx. 02 linhas)

▷ *0,5 pt*

Quicksort

Questão:

O Professor Saulo Doce propôs uma seguinte variação do algoritmo Quicksort:

Algoritmo *qsort*
Entrada $V = (v_0, \ldots, v_{n-1})$; $0 \leq l \leq r \leq n-1$

se $r - l > 1$ **então**
    $p \leftarrow \text{partition}(V, l, r)$
    $pred \leftarrow p$
    enquanto $l \leq pred$ **e** $V[pred] = V[p]$ **faça**
        $pred \leftarrow pred - 1$
    fim faça
    $suc \leftarrow p$
    enquanto $suc \leq r$ **e** $V[suc] = V[p]$ **faça**
        $suc \leftarrow suc + 1$
    fim faça
    $qsort(V, l, pred)$
    $qsort(V, suc, r)$
fim se
fim

Supondo que o pivô escolhido é o elemento mais à esquerda do trecho particionado, responda:

a) Qual o custo assintótico do algoritmo no *pior* caso? Dê um exemplo de entrada de pior caso para $n = 10$.

b) Qual o custo assintótico do algoritmo no *melhor* caso? Dê um exemplo de entrada de pior caso para $n = 10$.

Centro de Informática UFPE

Universidade Federal de Pernambuco

Obrigado !!!

E como diria o mito...

nós tentamos.