

Programação avançada

Algoritmos de ordenação e grafos

- Leia o documento inteiro antes de começar a trabalhar.
- Esboce sua solução antes de iniciar a programação.
- O código deve compilar e rodar sem erros.
- Não há necessidade de apresentar um relatório.
- Os arquivos devem ser enviados para o Classroom.

Para os exercícios 1, 2 e 3, consideramos o seguinte arranjo de números inteiros positivos: `int t[] = {31 9 4 7 3 18 23 54 99 87 54 35 90 1 22 30 49 77 5 76};`

Exercício 1: Bubble-Sort

Implementar o algoritmo Bubble-Sort. É sugerido propor uma implementação próxima ao seguinte pseudo-código:

```
BUBBLESORT(A)
1  for i = 1 to A.length - 1
2      for j = A.length downto i + 1
3          if A[j] < A[j - 1]
4              exchange A[j] with A[j - 1]
```

Figura 1: Pseudo código de Bubble-Sort

Exercício 2: Quick-Sort

Implementar o algoritmo Quick-Sort. É sugerido propor uma implementação próxima ao seguinte pseudo-código:

	QUICKSORT (A, p, r)	PARTITION (A, p, r)
1	if $p < r$	1 $x = A[r]$
2	$q = \text{PARTITION}(A, p, r)$	2 $i = p - 1$
3	QUICKSORT ($A, p, q - 1$)	3 for $j = p$ to $r - 1$
4	QUICKSORT ($A, q + 1, r$)	4 if $A[j] \leq x$
		5 $i = i + 1$
		6 trocar $A[i]$ por $A[j]$
		7 trocar $A[i + 1]$ por $A[r]$
		8 return $i + 1$
	(a) Quick Sort	(b) Partition

Figura 2: Pseudo código de Quick-Sort

Exercício 3: Merge-Sort

Implementar o algoritmo Merge-Sort. É sugerido propor uma implementação próxima ao seguinte pseudo-código:

MERGE (A, p, q, r)	
1 $n_1 = q - p + 1$	
2 $n_2 = r - q$	
3 sejam $L[1..n_1 + 1]$ e $R[1..n_2 + 1]$ novos arranjos	
4 for $i = 1$ to n_1	
5 $L[i] = A[p + i - 1]$	
6 for $j = 1$ to n_2	
7 $R[j] = A[q + j]$	
8 $L[n_1 + 1] = \infty$	
9 $R[n_2 + 1] = \infty$	
10 $i = 1$	
11 $j = 1$	
12 for $k = p$ to r	MERGE-SORT (A, p, r)
13 if $L[i] \leq R[j]$	1 if $p < r$
14 then $A[k] = L[i]$	2 then $q = \lfloor (p + r) / 2 \rfloor$
15 $i = i + 1$	3 MERGE-SORT (A, p, q)
16 else $A[k] = R[j]$	4 MERGE-SORT ($A, q + 1, r$)
17 $j = j + 1$	5 MERGE (A, p, q, r)
(a) Merge	(b) Merge-Sort

Figura 3: Pseudo código de Merge-Sort

Exercício 4: Dois arranjos

A e B são dois arranjos classificados, onde A tem um buffer no final suficientemente grande para armazenar B. Escreva um método para mesclar B em A em ordem ordenada sem usar memória adicional.