UNIVERSIDADE DA BEIRA INTERIOR

Algoritmos e Estruturas de Dados **2º Semestre**

Frequência 2 (8 val) **1:15 h** **2023/2024**

A. [2,00 val] Árvores Binárias de Pesquisa (ABP)

Considere a seguinte declaração de EAD Árvore Binária:

```
struct NodoABP {
   int    Elemento;
   struct  NodoABP  *Esquerda;
   struct  NodoABP  *Direita;
};
typedef  struct  NodoABP *PNodoABP;
```

Implemente uma função em C que

- **receba** uma ABP **T**,
- **devolva** o menor elemento positivo (inteiro > 0) entre todos os elementos da ABP T (se T está vazia ou se todos os elementos de T são não positivos (≤ 0), então a função deve devolver **-1**).

Nota: otimize o algoritmo criado (código escrito) tendo em conta a **definição de ABP**.

B. [2,50 val] ABP Balanceadas/Equilibradas (AVL)

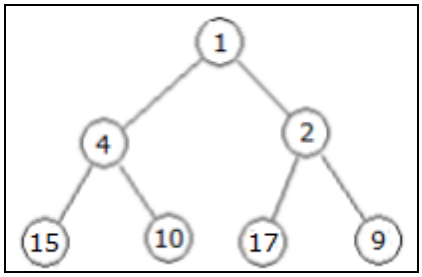
Equilibrar uma **ABP do tipo AVL**, consiste em aplicar uma das 4 operações de **rotação**: simples à esquerda, simples à direita, dupla à esquerda e dupla à direita.

a) **Construa** uma **ABP do tipo AVL inserindo** os nodos com os seguintes valores (um a um e pela ordem apresentada): **50, 40, 30, 60, 55 e 65**. (**NOTA**: não é necessário colocar as alturas).
Sempre que **inserir um nodo**, deve **redesenhar** a **ABP** obtida (acrescentar o nodo à árvore) e **verificar se continua balanceada/equilibrada**. Caso **não fique equilibrada**, deve
1º) indicar o nodo onde se deteta o desequilíbrio e o tipo de rotação a aplicar para reequilibrá-la,
2º) reequilibrar a árvore e redesenhá-la (apresentando todos os passos efetuados)

b) **Remova** o nodo **raiz** da ABP **obtida em a)**, **redesenhe** a árvore obtida, **verifique** se continua balanceada/equilibrada e **reequilibre-a**, caso seja necessário. Apresente a árvore final.

C. [1,00 val] Heaps

Considere a seguinte **minHeap**:



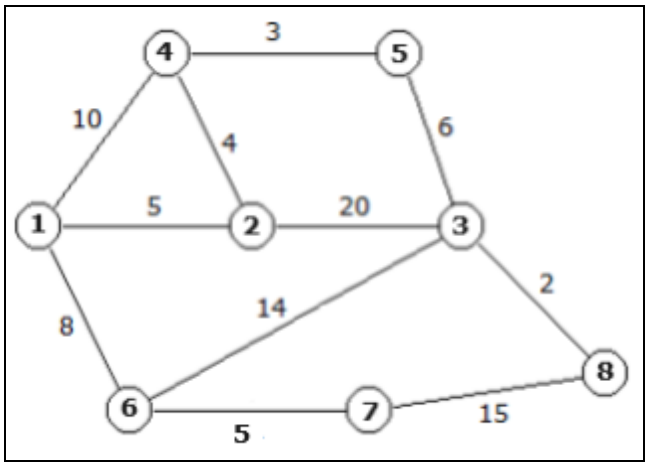
a) Insira um nodo com o valor **3** na minHeap dada (inicial) e desenhe a minHeap obtida. Jusstifique, apresentando a evolução da minHeap desde a estrutura inicial até à final.

b) Aplique a operação **remover** à minHeap dada (inicial) e desenhe a minHeap obtida. Justifique, apresentando a evolução da minHeap desde a estrutura inicial até à estrutura final.

NOTA: Descreva os processos aplicados para obter as **minHeaps** finais nas duas questões.

D. [2,50 val] Grafos

Considere a seguinte rede G = (A, N, C), onde o valor de cada arco corresponde ao seu custo (C_{ij}):



1. Construa a **lista de Adjacência** da rede G.
2. Determine a Árvore dos Caminhos Mais Curtos **do nó 1 (S = 1) para todos os outros nós**, simulando o algoritmo de **Dijkstra**. Apresente os dados da simulação, em especial: evolução dos rótulos, conjuntos dos Permanentes e dos Temporários, evolução dos valores de **k** e de **j**.

   Esquema proposto para apresentar os dados da simulação:

	1	2	3	4	5	6	7	8
R_i								
Q_i								

   Permanentes ←
   Temporários ←
   k ← ?
   j ← …

3. Qual o C.M.C. entre o nó 1 e o nó 8 ? E qual o custo deste caminho ? Justifique.

Algoritmo de Dijkstra:

Passo 1.
[Q_S, R_S] = [S, 0]
[Q_i, R_i] = [S, C_{Si}], $\forall$ i $\in$ N - {S} e (S, i) $\in$ N
[Q_i, R_i] = [S, ∞], $\forall$ i $\in$ N - {S} e (S, i) $\notin$ N
Temporários = N - { S }
Permanentes = { S }

Passo 2.
se Temporários = $\varnothing$ **então**
PARAR
fim_se
k = nó de Temporários tal que R_k é o mínimo
($k : R_k = \min \{ R_x , x \in$ Temporários $\}$)
Temporários = Temporários - { k }
Permanentes = Permanentes $\cup$ { k }

Passo 3.
para todo o **j** $\in$ **N** com (k, j) $\in$ A **e** j $\in$ Temporários **fazer**
se ($R_k + C_{kj} < R_j$) **então**
$R_j = R_k + C_{kj}$
Q_j = k
fim_se
fim_para
Regressar ao Passo 2