UNIVERSIDADE DA BEIRA INTERIOR

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

Frequência 2 (8 val) **1:30 h** **26/05/2025**

A. [1,0 val + 2,0 val] Árvores Binárias de Pesquisa (ABP)

Considere as seguintes declarações associadas de EAD Árvore Binária de Pesquisa:

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

1. Implemente uma função em C que **receba** uma ABP **T** e um número **inteiro X**, e **devolva 1** se existe algum nodo em **T** com valor no campo **Elemento igual a X** ou **0** se não existe.
2. Implemente uma função em C que **receba** uma ABP **T** e um número **inteiro X**, e **devolva** um ponteiro para o **nodo-pai** do **nodo** com o **menor** valor no campo **Elemento** que seja **maior** que **X**; no caso em que **não exista o nodo-pai**, a função deve devolver **NULL**.

Nota: otimize os algoritmos criados (código escrito) tendo em conta a **definição de ABP**.

B. [2,0 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.

Construa uma **ABP do tipo AVL** da seguinte forma:

- **insira** os nodos com os seguintes valores (um de cada vez e pela ordem apresentada):

  30, 70, 50, 20, 40, 25 e 45. (**NOTA**: não é necessário colocar as alturas).

  Sempre que **inserir um nodo**, deve **acrescentar** o novo nodo à árvore e **verificar se continua balanceada/equilibrada**. Caso **não fique equilibrada**, deve

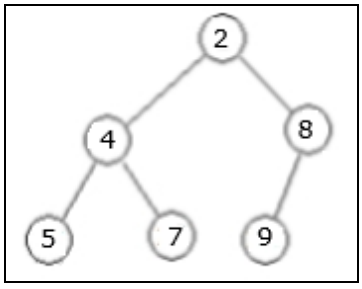
  1º) indicar o nodo onde se deteta o desequilíbrio e o tipo de rotação a aplicar para reequilibrá-la,

  2º) reequilibrar a árvore e redesenhá-la (apresentando todos os passos efetuados)

- **remova** o nodo **com o valor 20** da ABP, **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**:

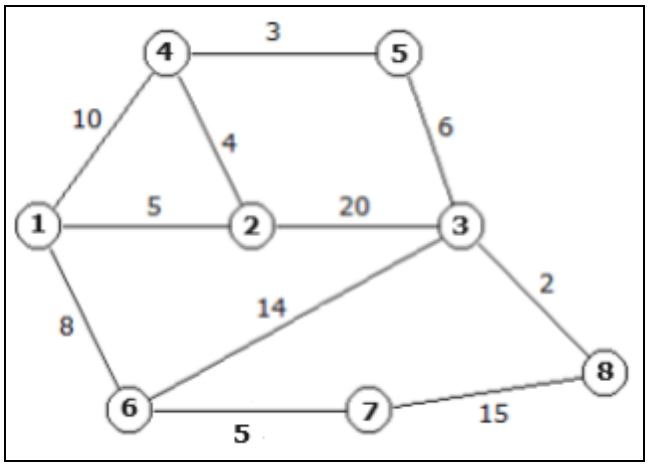


1. Aplique a operação **remover** à minHeap dada e desenhe a minHeap obtida. Justifique, apresentando a evolução da minHeap desde a estrutura inicial até à estrutura final.
2. Insira um nodo com o valor **6** na minHeap obtida em **1.** e desenhe a minHeap obtida. Justifique, apresentando a evolução da minHeap desde a estrutura inicial até à final.

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

D. [2,00 val] Grafos

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



1. Construa a **lista de Adjacência** da rede G.
2. Determine a **Árvore Abrangente Mínima (AAM)** tomando como **nó inicial** o nó **5 (S = 5)**, simulando o algoritmo de **PRIM**. Apresente os dados da simulação: evolução das estruturas de dados (rótulo **R**, **Pai**, **Permanentes** e **Temporários**) e de todos os valores de **k** e de **j**.

   Esquema proposto para apresentar os dados da simulação:

	1	2	3	4	5	6	7	8
R								
Pai								

   Permanentes ←
   Temporários ←
   k ← ?
   j ← …

3. Qual o **custo** da **AAM** determinada em 2? Justifique.

Algoritmo de PRIM:

Passo 1.
Tome-se arbitrariamente um nó S
Atribuir-se ao nó S um rótulo permanente:
$R_S \leftarrow 0$
$Pai_S \leftarrow 0$
Aos restantes nós atribuem-se rótulos temporários:
$R_j \leftarrow C_{Sj}$, se (S, j) $\in$ A
$R_j \leftarrow \infty$, se (S, j) $\notin$ A
$Pai_j \leftarrow$ S , se (S, j) $\in$ A
Pai $\leftarrow$ NULO , se (S, j) $\notin$ A
Temporários = N - { S }
Permanentes = { S }

Passo 2.
k $\leftarrow$ nó com rótulo temporário contendo o **menor valor**
Temporários $\leftarrow$ Temporários - { k }
Permanentes $\leftarrow$ Permanentes $\cup$ { k }
(o arco (Pai_k, k) passa a pertencer à árvore abrangente mínima, com o valor R_k)
se (Temporários = $\varnothing$) **então**
PARAR *(foi determinada uma árvore abrangente mínima)*
fim_se

Passo 3.
para (todo o **j** $\in$ **N** com (k, j) $\in$ A **e** j $\in$ Temporários) **repetir**
se ($C_{kj} < R_j$) **então**
$R_j \leftarrow C_{kj}$
$Pai_j \leftarrow$ k
fim_se
fim_para
regressar ao Passo 2