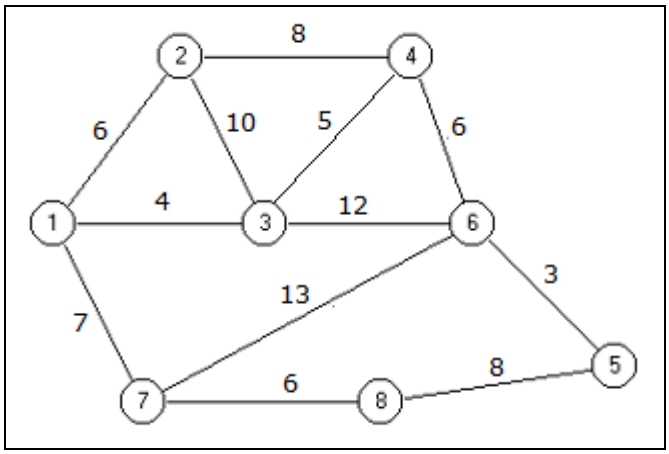
UNIVERSIDADE DA BEIRA INTERIOR

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

Frequência 2 (8 valores) **1:30 h** **01/06/2026**

A. [2,00 val] Grafos

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



1. Determine a **Árvore dos Caminhos Mais Curtos (ACMC)** do **nó 1 (S = 1) para todos os outros nós**, simulando o algoritmo de **DIJKSTRA**. 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** (usar esquema de resposta em baixo).
2. Indique **o caminho mais curto** entre o nó **1** e o nó **5**, assim como o seu **comprimento**. Justifique usando os dados obtidos na simulação.

Esquema proposto para apresentar os dados da simulação (questão 1):

	1	2	3	4	5	6	7	8
R								
Pai								

Permanentes ←
Temporários ←
k ← ?
  j ← …

Algoritmo de DIJKSTRA:

Passo 1.
  $R_S \leftarrow 0$
  $Pai_S \leftarrow S$
  $R_i \leftarrow C_{Si}$, se (S, i) $\in$ A
  $Pai_i \leftarrow S$, se (S, i) $\in$ A
  $R_i \leftarrow \infty$, se (S, i) $\notin$ A
  $Pai_i \leftarrow S$, se (S, i) $\notin$ A
  Permanentes = { S }
  Temporários = N - { S }

Passo 2.
  se (Temporários = $\varnothing$) **então**
    PARAR *(todos os nós têm rótulos permanentes – determinada ACMC)*
  fim_se
  k ← nó de Temporários em que R_k é mínimo (k : R_k = min {R_X , X $\in$ Temporários})
  Permanentes ← Permanentes $\cup$ { k }
  Temporários ← Temporários - { k }

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

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 (pela ordem indicada):

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

  60, 30, 10, 90, 70 e 80 (**NOTA**: ao desenhar as ABP não é necessário colocar as alturas)

  Sempre que **inserir um nodo**, deve-o **acrescentar** à ABP e **verificar se continua 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,

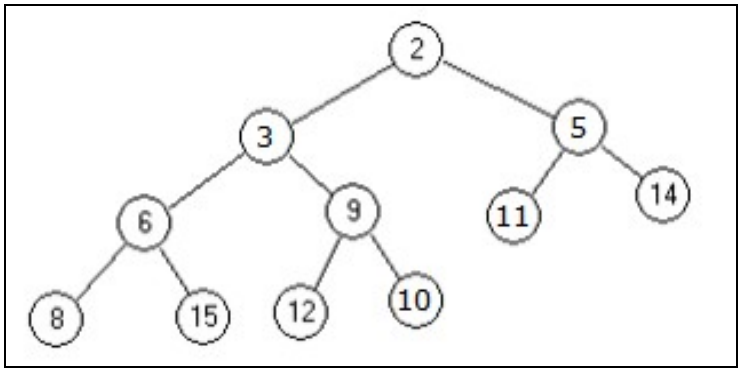
  2º) reequilibrar a árvore e redesenhá-la (apresentando todos os passos efetuados)

- **remova** a **raiz** da ABP obtida antes e **redesenhe** a ABP obtida com a remoção, **verifique** se continua equilibrada e **reequilibre-a**, caso seja necessário. Apresente a árvore final.

NOTA: em caso de rotação dupla, apresentar (desenhar) as duas rotações que lhe estão associadas

C. [1,25 val] Heaps (Filas de Prioridade)

Considere o seguinte **minHeap binário**:



1. **Insira** um nodo com o valor **4** no minHeap dado (ao lado) e desenhe o minHeap obtido. Justifique, apresentando a evolução do minHeap desde a estrutura inicial até à final.
2. Aplique a operação **remover** ao minHeap dado (ao lado) e redesenhe o minHeap obtido. Justifique, apresentando a evolução do minHeap desde a estrutura inicial até à final.

D. [1,0 val + 1,75 val] Árvores Binárias de Pesquisa (ABP)

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

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

1. Implemente **uma** função **recursiva** em C que **receba** uma ABP **T** e um número **inteiro X**, e **devolva** a quantidade de elementos de **T** com valor no campo **Elemento maior que X**.

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

2. Implemente **uma** função em C que **receba** uma ABP **T** e dois números **inteiros X e Y** (X < Y), e **devolva** o número de ligações (arestas) que existem entre X e Y. Caso X ou Y não pertença a T, a função deve devolver um valor negativo.

   Ex: para a ABP dada (ao lado) e para **X = 13** e **Y = 17**, o resultado é **3**.

   Ou seja, existem 3 arestas entre 13 e 17: (10, 13), (15, 10) e (15, 17)

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