

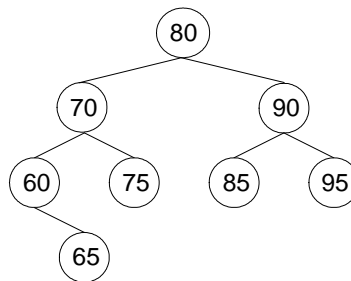
Árvores Binárias de Pesquisa

1. Considere as classes `treeNode`, `bst` e as classes `iterator` segundo as diferentes travessias numa árvore.

a) Defina uma classe `tree` por herança da classe `bst`.

Adicione à classe `tree` os seguintes métodos:

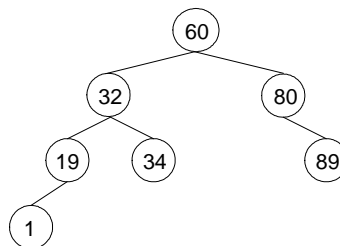
- b) Listar a subárvore esquerda da raiz principal por ordem crescente e a subárvore direita por ordem decrescente: 60, 65, 70, 75, 80, 95, 90, 85



- c) Criar uma árvore a partir de outra com todos os nós excepto as folhas

Árvores AVL

1. Considere a seguinte árvore binária de pesquisa.



Supondo que é usado o mecanismo AVL de balanceamento deste tipo de árvores, mostre a evolução da árvore de acordo com a seguinte sequência de operações. Deve mostrar apenas quatro configurações diferentes, uma por cada alínea.

- | | |
|-------------------|-------------------|
| 1. Remoção do 60 | 2. inserção do 85 |
| 3. Inserção do 61 | 4. inserção do 74 |

Indique qual a sequência de rotações a efetuar para cada uma das alíneas, pela ordem de execução respetiva (basta mencionar os elementos em que as rotações são feitas e indicar o tipo de rotação e o respetivo sentido).

Heaps

1. Ilustre a execução do algoritmo heapsort sobre os valores 10, 14, 19, 2, 30, 1, 20, 22. Considere que o heap está organizado por máximos.
2. O que acontece se o algoritmo heapsort for utilizado num vector ordenado ? E se o vector estiver em ordem inversa ?
3. Adicione à classe template `Heap<T>` o método que retira os elementos repetidos de um heap. Exemplo, heap: 8, 7, 4, 5, 3, 2, 3, 1, 2 heap resultante: 8, 7, 5, 4, 3, 2, 1. Desenvolva o método recorrendo apenas aos métodos da classe template `Heap<T>`.