Módulo 8 - Laboratório prático – Exercícios

Em cada módulo, os exercícios assinalados com ♣ são mais importantes. Os exercícios extra apresentam treino extra ou aplicações importantes dos TAD do módulo. São exercícios cuja prática complementa os conceitos e ajuda a perceber como os aplicar.

Exercícios:

1. ♣ Implemente a classe **Árvore Binária de Pesquisa** que obedeça à interface descrita na aula.

2. ♣ Implemente o TAD **Árvore Binária** através da interface abaixo indicado. **Nota**: Deve criar uma classe Element, para armazenar a informação a guardar em cada nó da árvore. Esta classe deve possuir, pelo menos, um método de igualdade para poder comparar duas chaves (keys) usando os operadores __eq__, __ne__, __lt__, __le__, __gt__, __ge__ . Por exemplo, a comparação de "maior que" pode ser implementada por:

```
def __gt__(other):
return self.key > other.key
```

Método - assinatura	Descrição
height()	Devolve a altura da árvore
len()	Devolve a quantidade de nós na árvore
root()	Devolve o nó raiz da árvore
element(p)	Devolve o elemento (valor) armazenado no nó que contêm p (se não existir nó p deve lançar uma exceção)
get_child(p, left)	Devolve um ponteiro para o filho esquerdo (left = True) ou direito (left = False) do nó p (se não existir o nó p deve lançar uma exceção)
add_root (X)	Cria e devolve uma estrutura de árvore, criando o nó raiz com valor X e sem filhos
attach_children(p, lt, rt)	Liga o nó lt como filho esquerdo de p e o nó rt como filho direito de p (se p não existir ou se já tiver um nó num dos filhos, lança uma exceção).
insert_child(X, p, left)	insere X na árvore criando novo nó filho esquerdo (left = True) ou direito (left = False) do nó p (se p não existir ou se já tiver um nó nesse filho, lança uma exceção).
child(p)	Devolve o filho esquerdo (left = True) ou direito (left = False) do nó p (se não existir filho , devolve None; se p for vazio, lança uma exceção)
mostra()	Imprime os valores armazenados na árvore usando uma travessia em ordem
is_node(X)	Indica se o valor X está armazenado num nó da árvore ou não
get_node(X)	Devolve o ponteiro para o nó onde está armazenado X na árvore (se não existir um nó com X devolve None)

3. ♣ Efetue uma nova implementação da classe **Árvore Binária de Pesquisa** (ABP) que obedeça à interface descrita na aula, mas que, agora, seja uma subclasse – ABPH - da classe **Árvore Binária** do exercício anterior.

4. Desenvolva um programa para efetuar os seguintes passos:

a. Cria um conjunto com 1.000.000 de números aleatórios e carrega os elementos em duas estruturas diferentes: numa Árvore Binária e numa ABP.

b. Cronometra uma pesquisa de 100 elementos, retirados aleatoriamente do conjunto em ambas as árvores. Atenção que deve usar o mesmo número em cada uma das pesquisas das duas pesquisas para poder comparar.

c. Analise os resultados e tire conclusões que justifiquem as diferenças encontradas ?

(*Dica*: Não esqueça que pode utilizar o módulo *random* e as operações aí definidas, quer para gerar os valores aleatórios, quer para a amostragem do conjunto.)