EA4 – Éléments d'algorithmique
TP n° 9 : B-arbres

Vous téléchargerez sur Moodle le fichier `tp9.py` à compléter.

On rappelle qu'un *B-arbre d'ordre p* vérifie :

- chaque nœud ou feuille contient au plus $2p$ clés ;
- chaque nœud ou feuille (***sauf la racine***) contient au moins p clés ;
- la racine d'un B-arbre non vide contient au moins une clé ;
- un nœud d'arité $k+1$ contient exactement k clés ;
- ***toutes les feuilles ont la même profondeur.***

La propriété d'ordre des ABR s'étend quant à elle simplement aux nœuds d'arité $k+1$: si un nœud contient les clés $c_0 < c_1 < \cdots < c_{k-1}$ et possède les sous-arbres A_0, A_1... A_k, tous les éléments de A_i sont supérieurs à c_{i-1} (si $i > 0$) et inférieurs à c_i (si $i < k$).

Exercice 1 :

Écrire la méthode `contient` de la classe `Bnoeud` qui, étant donné une clé `x`, retourne `(True, i)` si `x` est dans le tableau de clés du nœud en position `i`. Sinon, `contient` retourne `(False, i)` où `i` est l'indice auquel `x` doit être inséré dans le tableau de clés pour préserver un tableau trié.

Exercice 2 :

Écrire la méthode `cherche` de la classe `Barbre` qui, étant donné une clé `x`, retourne :

- `(True, n, i)` si `x` appartient au B-arbre, où `n` est le nœud dont le tableau de clés contient `x` et `i` est l'indice de `x` dans le tableau de clés,
- `(False, n, i)` si `x` n'appartient pas au B-arbre, où `n` est la feuille à laquelle on aboutit lorsqu'on cherche `x` et `i` est l'indice où l'on doit insérer `x` dans le tableau de clés de la feuille pour garder un tableau trié.

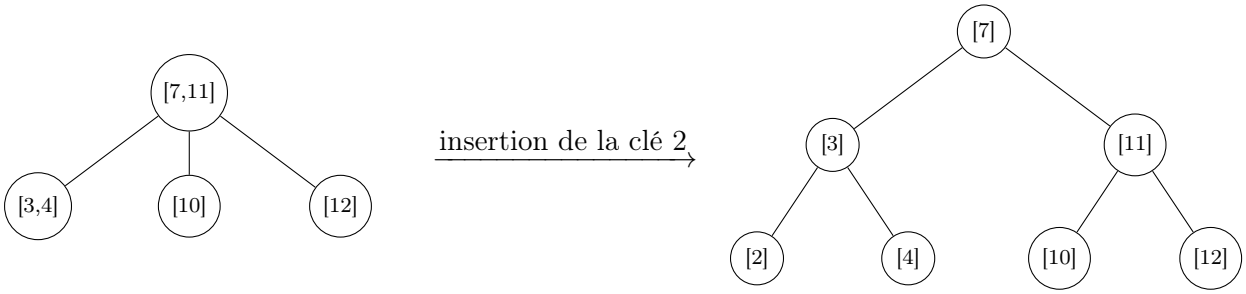
Exercice 3 :

Écrire la méthode `ajoute` de la classe `Barbre` qui ajoute la clé `x` passée en paramètre au B-arbre en préservant la propriété de B-arbre.

Lorsqu'on veut insérer une clé c dans un B-arbre a, il faut procéder de la façon suivante :

1. si la clé c se trouve déjà dans l'arbre, on ne fait rien ;
2. sinon, soit f la feuille à laquelle on aboutit lors de la recherche de c. On souhaite insérer c dans f. Il y a alors deux cas :
   - si le nombre de clés de f est inférieur à $2p$, c'est faisable et on a fini ;
   - sinon, c'est-à-dire si f est saturée, on la scinde en deux feuilles, qui reçoivent chacune p clés parmi les $2p+1$, et la médiane remonte dans l'arbre. Il faut maintenant essayer de l'insérer dans le père g de f, c'est-à-dire qu'il faut reprendre à l'étape 2 où f est maintenant le père g et c la clé médiane.

Par exemple dans l'arbre ci-dessous à gauche, on insère la clé 2 avec $p = 1$: on cherche la feuille où insérer la clé et on trouve la feuille contenant les clés $[3, 4]$. La liste est déja de taille maximale. On insère donc 2 dans $[3, 4]$, puis on scinde la liste en 3 : $[2, 3, 4] \rightarrow [2], [3], [4]$. On attache alors 2 nœuds de tableaux de clés respectifs [2] et [4] au nœud de tableau de clés [3]. Puis on insère 3 dans le tableau de clés de son père. On obtient $[3, 7, 11]$, qu'il faut scinder car on a trop de clés : $[3, 7, 11] \rightarrow [3], [7], [11]$. Et on obtient finalement l'arbre ci-dessous à droite.



Exercice 4 :

Faire des expériences statistiques sur les B-arbres obtenus par ajouts successifs de clés dans un ordre aléatoire : hauteur de l'arbre, nombre de nœuds, taux de remplissage des tableaux de clés...