Document de travail pour le TD4

Question 1- arbre binaire

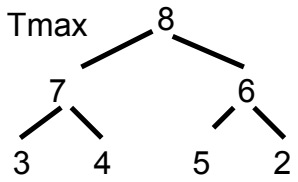
Nous allons travailler sur des arbres binaires contenant des entiers.

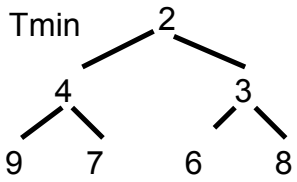
Pour tout arbre binaire non vide, il est nécessaire :

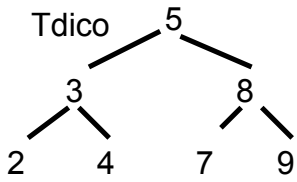
- d'accéder à sa racine (qui est un entier) ;
- d'accéder à ses fils gauche et droit qui sont des arbres binaires, éventuellement vides ;

Pour tout arbre binaire, il faut savoir s'il est vide ou non. Vous devez décrire une classe arbreB, comme indiqué ci-dessous, avec ces opérations de base :

```
public class arbreB
{
      private arbreB fg, fd;
      private int rac;

        // constante arbre vide
  public static final arbreB Avide= null;

        // pour la génération aléatoire de valeurs entre 0 et 9
  public static final int maxAleat=10;

         // dans les tableaux d'initialisation le premier element est -2
         // pour travailler entre les indices 1 et max
  public static final int Tmax[]={-2,8,7,6,3,4,5,2};
  public static final int Tmin[]={-2,2,4,3,9,7,6,8};
  public static final int Tdico[]={-2,5,3,8,2,4,7,9};
```

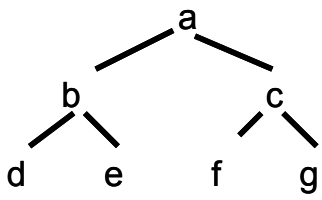






Ajoutez à cette classe des méthodes **récursives** permettant :

- de calculer la **hauteur** de l'arbre ;
- de tester si cet arbre est **équilibré** ;
- de **parcourir** un arbre binaire en affichant chacune des valeurs trouvées :



préfixe *a b d e c f g*

infixe *d b e a f c g*

postfixe *d e b f g c a*

largeur *a b c d e f g*

  - parcours prefixe (racine, fils gauche, fils droit),
  - parcours postfixe (fils gauche, fils droit, racine),
  - parcours infixe (fils gauche, racine, fils droit).

  Que pensez-vous du parcours en largeur d'abord ?

- de **rechercher** un entier dans l'arbre (on recherche une des occurrences, sans contrainte particulière) :

- sans aucune hypothèse,
- en supposant que l'arbre est un tas maximum,
- en supposant que l'arbre est un tas minimum,
- en supposant que l'arbre est arbre dictionnaire (équilibré et ordonné) ;

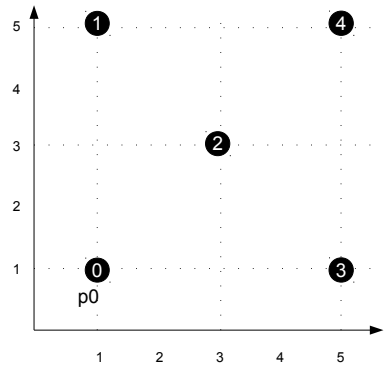
Ajoutez à cette classe des méthodes permettant d'**initialiser** un arbre binaire :

- la hauteur de l'arbre est donnée, l'arbre sera équilibré, les valeurs entières sont tirées de façon aléatoire (entre à et `maxAleat`) ;
- à partir d'un tableau d'entiers (en respectant la convention vue au TP3, à savoir que les fils gauche et droit d'un élément placé à l'indice *i* du tableau se trouvent respectivement aux indices *2i* et *2i+1*) ;

Question 2- enveloppe convexe et gift wrapping
-marche de Jarvis (version 2)-

Le calcul « alternatif » consiste à :

- *choisir comme point p* le point le point le plus en bas à gauche (p0)
- répéter jusqu'à retrouver p0 :
  - choisir le point q suivant de p (en boucle sur les points)
  - pour tous les points (i) :
    remplacer q par i si l'orientation de p-q-i est à droite
  - le point q en fin de boucle entre dans l'enveloppe et remplace q



C'est le signe du déterminant du produit vectoriel de PQ et QI qui détermine l'orientation de p-q-i. Dans le cas de points p-q-i alignés (orientation 0), le point le plus éloigné de P est conservé (dans le tableau donné en exemple ci-dessous, c'est la différence des distances PQ et PI qui détermine le choix entre Q et I).

Sur l'exemple du carré avec un point au centre, on obtient :

étape 1
- initialisation de p avec le point 0 (en bas à gauche)
- initialisation de q avec le suivant, point 1
- boucle (i de 0 à 4) : le point q en sortie de boucle (point 3) est conservé pour l'enveloppe convexe

étape 2
- p remplacé par q (3)
- initialisation de q avec le suivant, point 4

- boucle (i de 0 à 4) : le point q

p	q	i	orientation	distance	choix
					0
0	1	0	0	-16	
0	1	1	0	0	
0	1	2	- (droite)	-8	
0	2	3	- (droite)	8	
0	3	4	+ (gauche)	16	3
3	4	0	+ (gauche)	0	
3	4	1	+ (gauche)	16	
3	4	2	+ (gauche)	-8	
3	4	3	0	-16	
3	4	4	0	0	4
4	0	0	0	0	
4	0	1	- (droite)	-16	
4	1	2	+ (gauche)	-8	
4	1	3	+ (gauche)	0	
4	1	4	0	-16	1
1	2	0	- (droite)	8	
1	0	1	0	-16	
1	0	2	+ (gauche)	-8	
1	0	3	+ (gauche)	16	
1	0	4	+ (gauche)	0	0

en sortie de boucle (point 4)
est conservé pour l'enveloppe
convexe

étape 3
- p remplacé par q (4)
- initialisation de q avec le suivant, point 0
- boucle (i de 0 à 4) :
  le point q en sortie de boucle (point 1) est conservé pour l'enveloppe convexe

étape 4
- p remplacé par q (1)
- initialisation de q avec le suivant, point 2
- boucle (i de 0 à 4) : le point q en sortie de boucle (point 0) termine l'enveloppe convexe

L'algorithme correspondant est le suivant :

```
gift ( PTS, max, EC, nbEC ) :
   p0 ← minimal (PTS,max)
   copier p0 en première position dans EC
   p ← p0
   répéter
      q ← p+1 modulo max
      pour i de 0 à max
      faire t ← orientation (p, q, i)
            si t < 0 ou ( t=0 et I plus éloigné de P que Q)
            alors q ← i
            fsi
      fpour
   enregistrer q dans EC
   p ← q
   jusqu'à p=p0
```

avec :
- `PTS` le tableau des points, `max` le nombre de points du nuage,
- `EC` l'enveloppe convexe et `nbEC` son nombre de points,
- `minimal (PTS, max)` le point le plus bas (et s'il y en a plusieurs, le plus à gauche en bas)
- `orientation (p, q, i)` le déterminant du produit vectoriel de PQ et QI[1]

Attention : le nuage de points ne doit pas contenir deux fois le même point. Quel problème peut-on avoir en générant aléatoirement des points distincts ?

Pour chercher le plus long des segments colinéaires PI et PQ (ou celui des points I ou Q qui éloigne le plus de P), utilisez leur produit scalaire ou la différence de leurs distances à P.

Vous trouverez des explications et discussions de cet algorithme sur :
https://interstices.info/jcms/c_16014/un-joli-algorithme-geometrique-et-ses-vilains-problemes-numeriques/

http://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/
http://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/
http://tomswitzer.net/2009/12/jarvis-march/

1 Seul son signe est indiqué dans l'exemple.