INF2010 - Structures de données et algorithmes
Hiver 2018
Travail Pratique 5
Monceaux

Objectifs :

Implémenter une structure d'arbre binomial
Implémenter un monceau binomial

Instructions générales

Dans ce TP, vous aurez à implémenter un monceau binomial max à l'aide de 2 classes : Node et Monceau.
Un fichier main vous est fourni afin de pouvoir tester vos classes ainsi qu'un fichier ResultatAttendu.txt qui correspond à l'affichage désiré.
Notez que les tests dans le fichier main ne sont pas complètement exhaustifs, pensez aussi à tester les cas extrêmes.

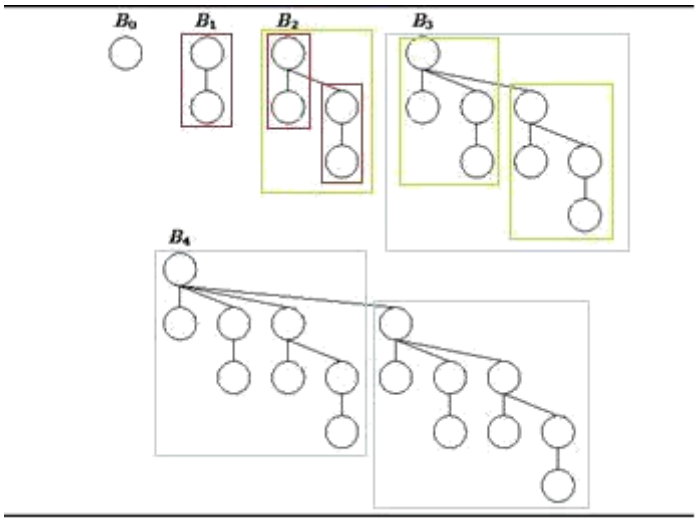
Monceau binomial

Vous avez vu en cours comment créer un monceau à partir d'un arbre binaire. Un inconvénient majeur d'un tel monceau est que la fusion de deux monceaux est une opération longue (O(n), où n est le nombre d'éléments). Nous allons ici nous intéresser à une autre implémentation possible, le monceau binomial, permettant une fusion beaucoup plus rapide.

Pour rappel, un monceau min (respectivement max) respecte la propriété que les enfants d'un nœud ont une valeur plus grande (resp. petite) que celle du parent. Ainsi, le nœud racine contient la valeur la plus petite (resp. grande) de l'arbre. Nous allons ici implémenter un monceau max.

Un monceau binomial est en fait une collection d'**arbres binomiaux**. Un arbre binomial peut se définir par récurrence :

- Un arbre binomial d'ordre 0 est constitué uniquement de la racine.
- Un arbre binomial d'ordre k est constitué de deux arbres d'ordre k-1, la racine de l'un étant l'enfant de la racine de l'autre.



Source : http://www.brpreiss.com/books/opus4/html/page371.html

Pour pouvoir construire un monceau binomial, ces arbres doivent en plus respecter la propriété d'ordre des monceaux énoncée plus haut.

La première étape de ce TP est d'implémenter une telle classe. Pour simplifier le travail, on ne va travailler ici qu'avec des arbres d'entiers.

Exercice 1 : Création d'une classe d'arbre binomial

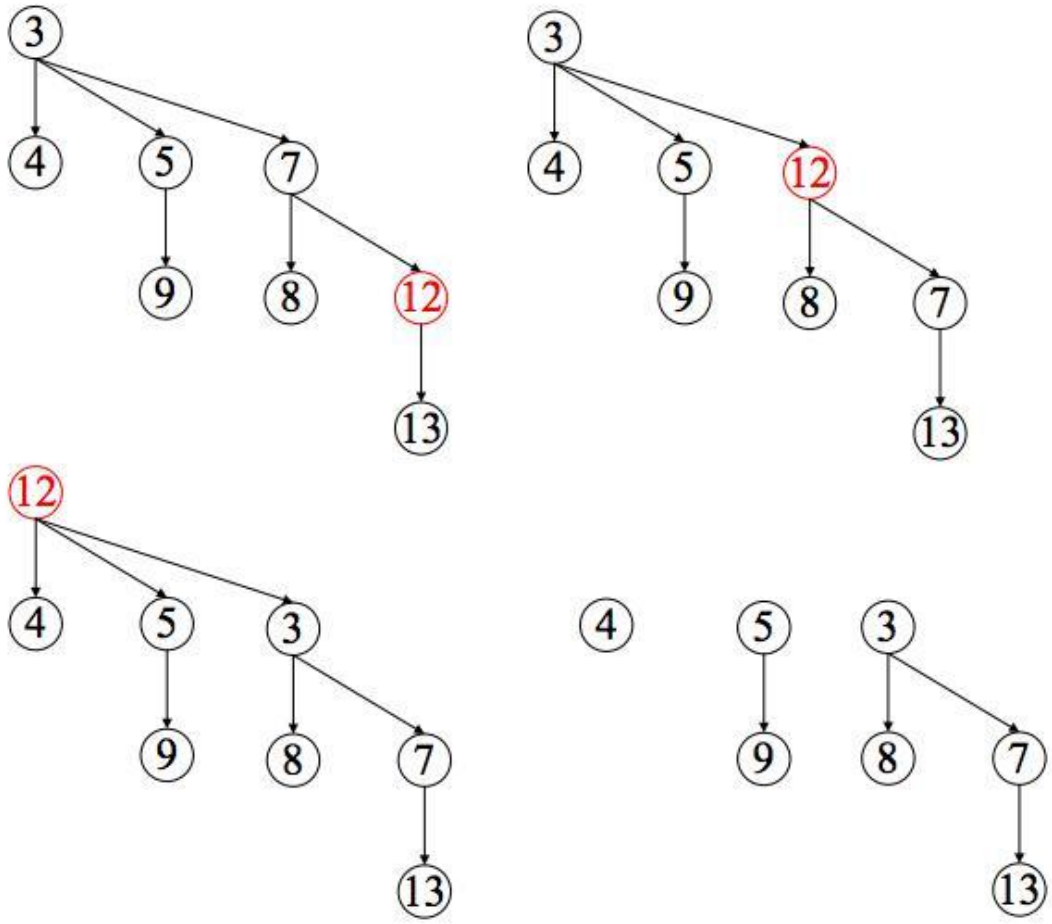
Les propriétés de l'arbre binomial font qu'il ne peut avoir que n = 2^k éléments, k étant son ordre. On ne peut donc pas insérer ou supprimer les éléments un à un. Par contre, il est très facile de fusionner deux arbres d'ordre k pour former un arbre d'ordre k+1.

On vous fournit une classe Node, représentant un nœud d'arbre binomial. L'arbre lui-même est tout simplement représenté par le nœud racine. Plusieurs méthodes sont déjà implémentées. Vous devez compléter les méthodes suivantes :

- `public Node fusion(Node autre)` : fusionne les deux arbres donnés. Vous devez vérifier que les arbres sont de même ordre et que les nœuds sont bien des racines (parent == null). Faites-en sorte de respecter la condition d'ordre du monceau (valeur du parent > valeur des enfants) !
- `public Node findValue(Node n, int valeur)` : renvoie le premier nœud contenant la valeur demandée, s'il existe, parmi les descendants (enfants, petits-enfants, etc.) de ce nœud, null sinon. Pensez à vérifier la valeur des enfants pour ne pas parcourir tout l'arbre inutilement, i.e. si on recherche 12 et qu'un enfant a pour valeur 8, on sait que ce n'est pas la peine d'aller explorer vers cet enfant.
- `private void moveUp()` : dans l'arbre, échange la position du nœud et de son parent. Attention, ceci brise la condition d'ordre du monceau : il ne faut utiliser cette fonction que dans `delete`. Pensez aussi à modifier la valeur d'ordre de tous les nœuds dont vous avez changé la position.

- `public ArrayList<Node> delete()` : supprime le nœud de l'arbre. Comme un arbre binaire ne peut contenir que 2^k éléments, l'opération de suppression consiste en fait à faire remonter le nœud jusqu'à ce qu'il soit racine de l'arbre (à l'aide de `moveUp`), et à renvoyer ses k - 1 enfants, chacun étant un arbre binomial.
- `public ArrayList<Node> getElementsSorted()` : tri les éléments de l'arbre en ordre décroissant et retourne ces éléments triés dans une ArrayList.

Les graphes suivants montrent le processus complet pour supprimer l'élément 12 d'un arbre binomial d'ordre 3 (attention, il s'agit d'un monceau min dans le cas ci-dessous)

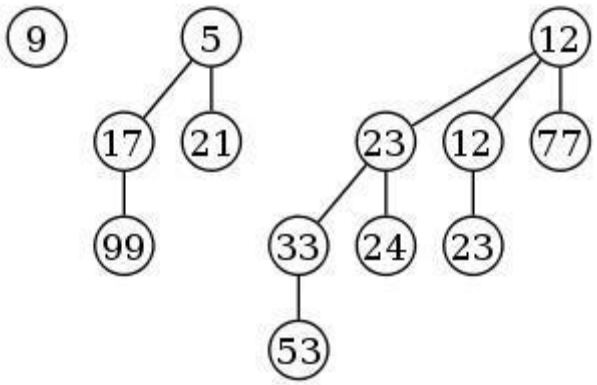


Une fonction `print()` est déjà implémentée et vous permet d'afficher votre arbre binomial.

Exercice 2 : Création du monceau binomial

Un monceau binomial est un ensemble d'arbres binomiaux. Cet ensemble peut contenir exactement 0 ou 1 arbre binomial de chaque ordre : il ne peut pas y avoir 2 arbres

d'ordre 3 par exemple. Comme le nombre d'éléments contenu dans un arbre ne peut être qu'une puissance de deux, les arbres présents dans le monceau dépendent de l'écriture en binaire du nombre d'éléments. Par exemple, si un monceau a 13 éléments, 13 en binaire s'écrit 1101. Il y aura donc un arbre d'ordre 3 (contenant 8 éléments), d'ordre 2 (contenant 4 éléments) et d'ordre 0 (contenant 1 élément).



Exemple de monceau binomial min contenant 13 éléments (source : Wikipedia)

L'opération de base est, comme pour l'arbre binomial, la fusion de deux monceaux binomiaux. Cette fusion se fait de la façon suivante :

On a la possibilité de garder un arbre temporaire, la « retenue ».

Pour tout j, en commençant par j = 0, on regarde le nombre d'arbres d'ordre j dans les deux monceaux plus la retenue. Il peut donc y en avoir entre 0 et 3.

- S'il n'y en a pas, le monceau final n'aura pas d'arbre d'ordre j.
- S'il y en a un, on place cet arbre dans le monceau final.
- S'il y en a deux, on les fusionne et place l'arbre résultat, d'ordre j+1, dans la retenue, pour l'itération suivante. Le monceau final n'aura pas d'arbre d'ordre j.
- S'il y en a trois, on en fusionne deux, on place le troisième dans le monceau final, et place le résultat de la fusion, d'ordre j+1, dans la retenue.

Implémentez cette opération en complétant la méthode `public void fusion(Monceau autre)` dans la classe Monceau.

Implémentez ensuite la méthode `public void insert(int Val)` permettant d'insérer une valeur dans le monceau. Cette méthode doit utiliser `fusion`.

À l'aide des fonctions `Node.findValue` et `Node.delete`, implémentez la méthode `public boolean delete(int val)` permettant de supprimer tous les nœuds ayant une certaine valeur. Attention, `Node.findValue` ne renvoie qu'un seul nœud, il faut donc potentiellement l'appeler plusieurs fois.
Vous pouvez afficher votre monceau à tout moment à l'aide de `Node.print()`.

Instructions pour la remise

Le travail doit être fait par équipe de 2 personnes et doit être remis via Moodle :

Jeudi B1 : 11 Avril 2018
Jeudi B2 : 04 Avril 2018

Veuillez envoyer vos fichiers .java seulement dans une archive de type *.zip qui portera le nom inf2010_lab5_MatriculeX_MatriculeY (de sorte que MatriculeX < MatriculeY). Les travaux en retard seront pénalisés de 20 % par jour de retard. Aucun travail ne sera accepté après 4 jours de retard.