TD d'Algorithmique et structures de données
Tas binomiaux

Équipe pédagogique Algo SD

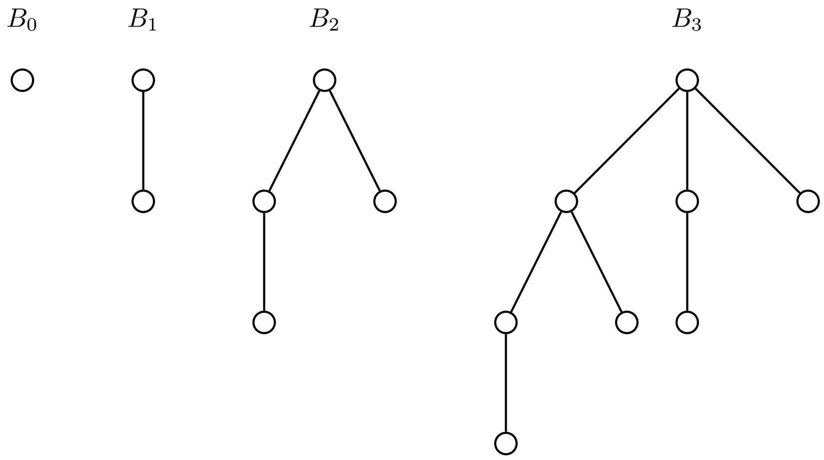
Rappelons qu'un arbre au sens informatique du terme possède une *racine* r, pour un sommet u de l'arbre, tous les sommets w tels que u est le dernier sommet avant w sur l'unique chemin de r à w s'appellent les *fils* de u. L'arbre est dessiné par niveaux, tous les sommets à la même distance de la racine sont sur le même niveau et les fils de chaque sommet sont ordonnés de gauche à droite.

Un arbre binômial B_k est un arbre défini récursivement comme suit :

— B_0 est l'arbre formé d'une racine et pas d'autres sommets.

— B_k est composé de deux arbres B_{k-1} qui sont liés de la façon suivante : la racine de l'un est le fils le plus à gauche de la racine de l'autre.

La figure suivante donne B_0, B_1, B_2 et B_3.



1. Dessiner B_4.
2. (On pourra considérer comme acquises les propriétés suivantes pour la suite du problème si on n'a pas réussi à faire les démonstrations qui sont toutes très simples) Montrer que pour B_k :
   1. Il y a 2^k sommets.
   2. La hauteur (ou profondeur) est de k.
   3. Il y a exactement C_k^i sommets au niveau i. (Les niveaux sont comptés de haut en bas, le niveau 0 contient la racine et le niveau k est celui le plus bas.)
   4. La racine possède k fils. En numérotant les fils de la droite vers la gauche en commençant par 0 (donc le fils le plus à droite est numéroté 0 et le fils le plus à gauche est numéroté $k-1$), le fils numéro i est la racine d'un B_i.
3. Déduire des propriétés précédentes que si n est le nombre de sommets d'un arbre binomial alors aucun sommet n'a plus de $\log n$ fils.

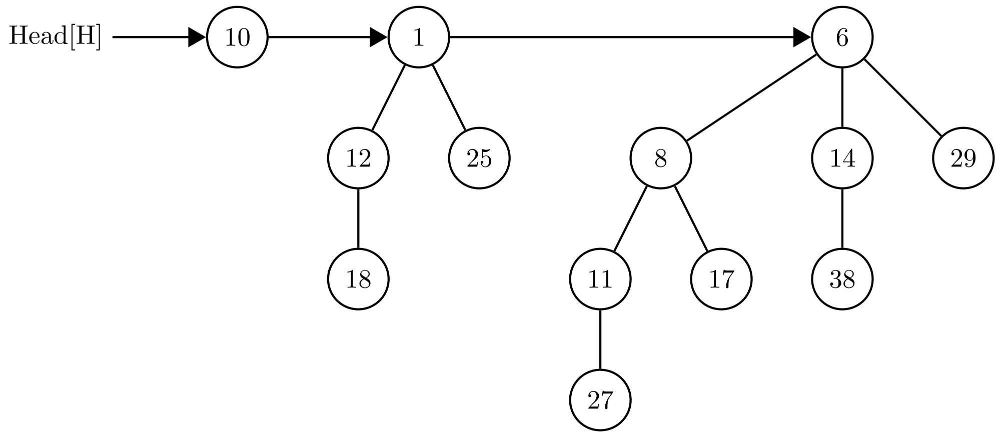
   Un *tas binômial* H (*binomial heap* en anglais) est un ensemble d'arbres binômiaux tels que les propriétés suivantes sont satisfaites :

   1. *Min-ordonné* : Dans chaque arbre binômial de H, la valeur attachée à chaque sommet (appelée dans la suite clé) est inférieure ou égale à celle attachée à ses fils (on aurait pu définir max-ordonné)

2. Pour tout entier k, il y a **au plus un** arbre binômial dans H de racine de degré k. (Au plus un veut dire au maximum un).

On utilisera la propriété suivante, qui est une conséquence de la deuxième condition ci-dessus, sans la démontrer : Dans un tas binômial H contenant n sommets en tout, il y a au plus $\lfloor \log n \rfloor + 1$ arbres. (Pour voir pourquoi, écrivons n sous forme binaire $n = \sum_{i=0}^{\lfloor \log n \rfloor} b_i 2^i$, on sait qu'un arbre binomial B_i possède 2^i sommets, donc l'arbre binomial B_i apparaîtra dans le tas H si et seulement si $b_i = 1$)

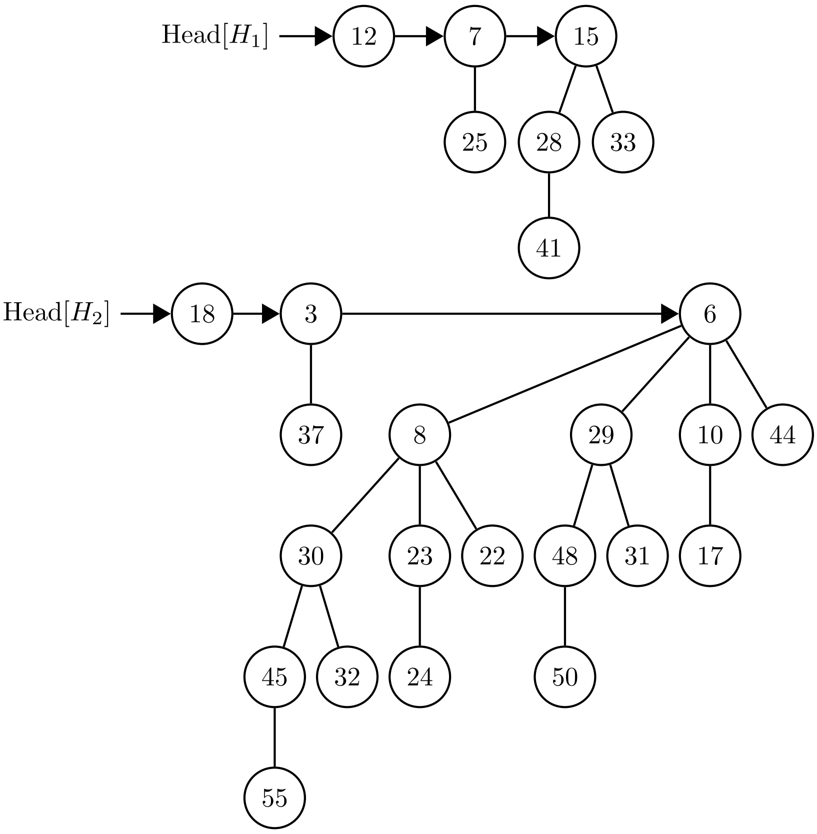
Dans un tas binômial H chaque nœud est une structure avec un lien vers son fils aîné, un lien vers son frère cadet et un autre vers son père. Dans le cas d'absence de tels parents le lien est $NULL$. De plus les racines des arbres sont dans une liste chaînée de gauche à droite (on peut utiliser pour cela le lien frère cadet des racines). La première racine est donnée par $head[H]$. Dans ce qui suit on supposera que les racines sont ordonnées de gauche à droite selon leur degré (nombre de fils), c'est ce qui est fait dans la figure suivante.



4. Ecrire l'algorithme de recherche de la clé minimum et déduire de ce qui précède que trouver la clé minimum se fait en $O(\log n)$.
5. La base de toutes les opérations sur les tas binomiaux est la fusion de deux tas binomiaux. La figure ci-dessous donne deux tas binomiaux H_1 et H_2.

   Pour fusionner deux tas binômiaux on met les arbres des deux tas dans une même liste ordonnée par degré croissant des racines.

   1. Dire pourquoi en général cela ne donne pas un tas binômial.
   2. Dans l'exemple qui vous est proposé, par des opérations que vous décrirez soigneusement vous ramener à un tas binômial. Vous pouvez simplifier les figures dans la mesure où on comprend bien ce que vous faites. Les figures ne sont qu'un support des explications mais ne les remplacent pas. (Aide : essayez de maintenir la propriété que les arbres de la liste sont en ordre croissant des degrés des racines)
   3. Donner les grandes lignes d'un algorithme de fusion de deux tas binômiaux.
   4. Quelle est la complexité de l'algorithme précédent en fonction du nombre de sommets final $n = n_1 + n_2$.
6. En utilisant l'opération de fusion de deux tas comme une fonction, écrire l'algorithme d'insertion d'un nouvel élément dans un tas binômial. Complexité ?
7. Écrire l'algorithme d'extraction (suppression) de l'élément de clé minimum dans un tas binômial. (Cela utilise à nouveau la fusion de deux tas, mais aussi la propriété 4 de la question 2). Complexité ?
8. Ecrire l'algorithme qui permet de diminuer la valeur de la clef d'un élément (racine ou noeud interne) d'un tas binômial. Complexité ?
9. En utilisant les algorithmes des questions précédentes, écrire un algorithme pour supprimer un élément quelconque du tas binômial : on supposera disposer d'une valeur $-\infty$ strictement inférieure à toutes les clefs dans le tas. Complexité ?


Head[H_1]
12
7
15
25
28
33
41
Head[H_2]
18
3
6
37
8
29
10
44
30
23
22
48
31
17
45
32
24
50
55