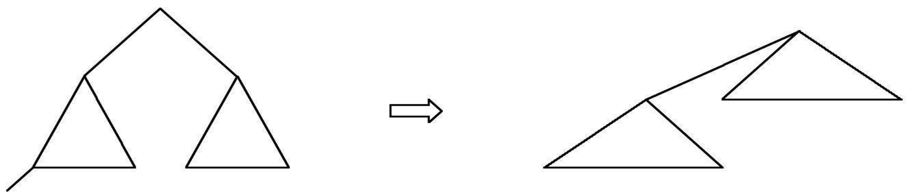
Université Pierre et Marie Curie | **Master Informatique**
23 Novembre 2005 | **MI015-ALGAV**

Devoir sur table

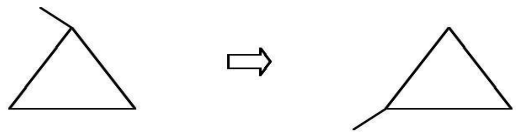
Durée : 2h. Tous documents autorisés.
Le barème donné est indicatif.

1 Tas et arbre binômial [5 points]

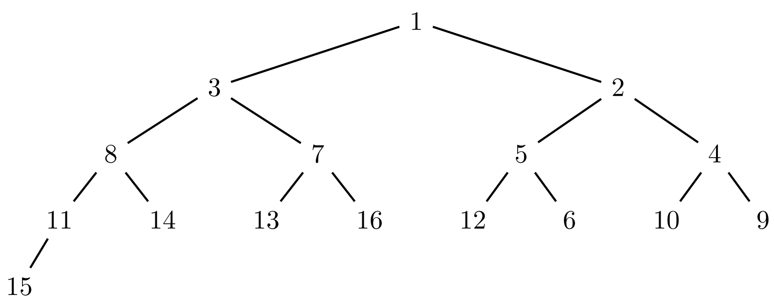
Le but de cet exercice est de convertir un tas binaire (arbre binaire parfait croissant) de 2^k éléments en un tas binômial (arbre binômial croissant), sans faire aucune comparaison.
Un tas binaire avec 1 élément est aussi un tas binômial. Pour $k \geq 1$, un tas binaire avec 2^k éléments est constitué d'un sous-arbre gauche qui est un tas binaire avec 2^{k-1} éléments et d'un sous-arbre droit qui est un tas binaire avec $2^{k-1} - 1$ éléments. On transforme d'abord le sous-arbre droit et la racine en un tas binaire de 2^{k-1} éléments, puis on convertit récursivement ces deux tas en des tas binômiaux de taille 2^{k-1} et on les concatène pour former un tas binômial de taille 2^k, comme le suggère la figure suivante.



1- Décrire un algorithme qui transforme en un tas binaire le sous-arbre droit et la racine d'un tas binaire (en gardant la propriété de tas). Calculer le nombre d'opérations effectuées.



2- Décrire l'algorithme de conversion un tas binaire de taille 2^k en un tas binômial. Et donner le résultat de cette conversion sur l'arbre suivant.



3- Exprimer par une relation de récurrence le nombre de concaténations de tas binômiaux effectués lors de la conversion d'un tas binaire de taille $n = 2^k$. Résoudre cette récurrence.

2 Coût amorti [5 points]

Une file FIFO est une structure de donnée linéaire qui supporte deux opérations
– *ajouter(x,F)*, qui ajoute un élément à la file,
– *enlever(F)*, qui enlève de la file l'élément le plus anciennement présent.
On peut implanter une file F à l'aide de deux piles P_1 et P_2 (et les opérations classiques *empiler(x,P)* et *dépiler(P)*), de la façon suivante :
– *ajouter(x,F)* : *empiler(x,P_1)*

– *enlever(F)* : si P_2 est vide alors dépiler tous les éléments de P_1 et les empiler au fur et à mesure dans P_2. Puis (dans tous les cas) dépiler P_2.

1- Montrer que cette implantation est correcte.

2- On va calculer le coût amorti des opérations *ajouter(x,F)* et *enlever(F)* en essayant différentes fonctions de potentiel. Le coût est mesuré en nombre d'opérations faites sur les piles.

a) essai 1 : la fonction de potentiel vaut deux fois le nombre d'éléments de la pile P_1.

b) essai 2 : la fonction de potentiel est égale au nombre d'éléments de P_1 moins le nombre d'éléments de P_2.

Pour chacune des deux fonctions précédentes, donner le coût amorti des opérations *ajouter(x,F)* et *enlever(F)* et dire s'il est possible que le coût réel total soit supérieur au coût amorti total. Peut-on retenir ces deux fonctions ?

3 Arbres bicolores [5 points]

La suppression dans un arbre bicolore étant délicate, on envisage de faire de la suppression paresseuse : plutôt que de supprimer "physiquement" chaque sommet au fur et à mesure, on se contente de le *marquer* comme étant supprimé, et lorsque la moitié des sommets d'un arbre sont marqués, on reconstruit totalement l'arbre (avec les "bons" sommets, *i.e.* les sommets non supprimés).

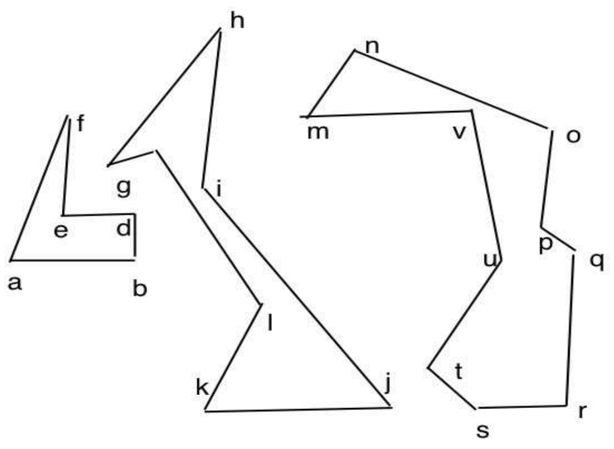
1- Montrer qu'étant donnée une liste triée on peut construire *en temps linéaire* un arbre bicolore contenant les éléments de cette liste.

2- Donner un algorithme qui, étant donné un arbre bicolore de n sommets dont certains sont marqués comme supprimés, reconstruit *en temps* $O(n)$ un arbre bicolore avec les bons sommets.

3- Supposons que l'on part d'un arbre bicolore de n éléments et que l'on effectue $n/2$ opérations de recherche, ajout ou suppression paresseuse. Montrer que le coût total de cette suite d'opérations est en $O(n \log n)$.

4 Géométrie [5 points]

On considère une famille finie $\mathcal{P}$ de polygones simples deux à deux disjoints. On note C l'ensemble des côtés des polygones de $\mathcal{P}$ et S l'ensemble de leurs sommets (avec $|S| = n$). Il s'agit de déterminer le sous-ensemble $A \in C$ des côtés c tels que tous les polygones de $\mathcal{P}$ sont du même côté de la droite de support c. Par exemple pour la famille de polygones ci-dessous A vaut $\{(k,j),(s,r),(r,q),(o,n),(f,a)\}$.



On admettra qu'un côté c appartient à A si et seulement si c est inclus dans un côté de l'enveloppe convexe de l'ensemble S des sommets de $\mathcal{P}$.

1- Étudier une structure de données représentant l'ensemble C des côtés de $\mathcal{P}$, pour pouvoir efficacement rechercher si un couple quelconque de points est un côté de C.

2- En déduire un algorithme de complexité $O(n \log n)$ pour calculer A à partir de $\mathcal{P}$.