TD n°4 - Correction

Tas

Correction : Un tas (*heap* en anglais) est un arbre binaire complet, qui possède en plus la propriété que la clef de tout nœud est supérieure ou égale à celle de ses descendants.

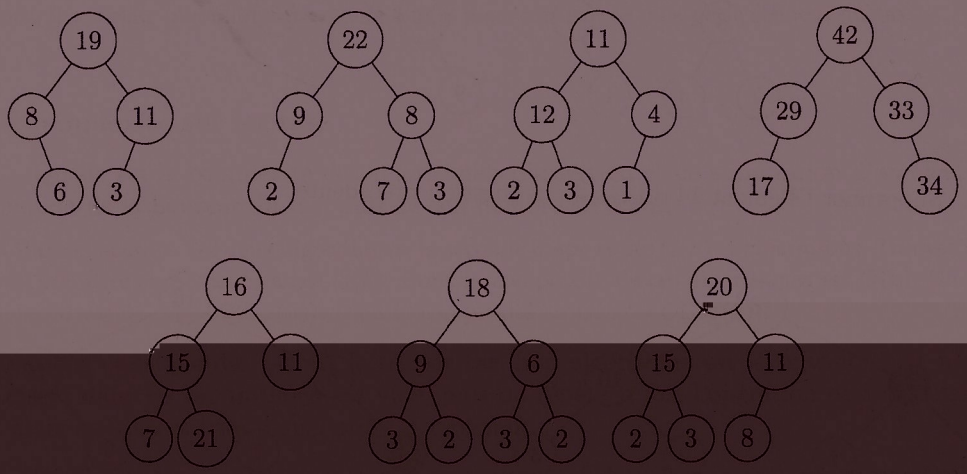
L'insertion dans un tas se fait en 2 étapes. La première consiste à insérer le nouvel élément en gardant la propriété d'arbre complet du tas, c'est-à-dire en créant un nouveau nœud dans le niveau de profondeur le plus élevé de l'arbre, et le plus à gauche possible.
Ensuite, il faut effectuer une opération d'ascension dans l'arbre pour placer le nouveau nœud au bon endroit. Cela se fait en remontant la branche entre le nœud et la racine : on compare le nœud et son nœud père et si le père est plus petit, on échange les nœuds et on continue à remonter. On s'arrête dès que le père est plus grand, ou à la racine.

Comme pour l'insertion, la suppression se fait en 2 étapes. La première est de supprimer le nœud et de le remplacer par le nœud de profondeur le plus élevé de l'arbre, et le plus à droite possible, pour garder la propriété d'arbre complet.
Ensuite, à partir de la position du nœud supprimé, on effectue un parcours descendant dans l'arbre. Si le nœud est plus petit qu'un de ses deux fils, on le permute avec le plus grand des deux, et on s'arrête dès qu'il est plus grand que ses deux fils, ou lorsqu'on est dans une feuille.

1 Exemples

Exercice 1 [Exemples de tas]

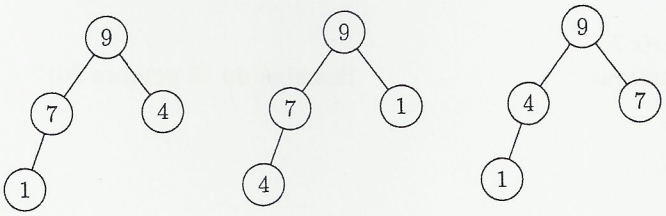
Question 1. Parmi les arbres suivants, lesquels ne sont pas des tas ? Pourquoi ?



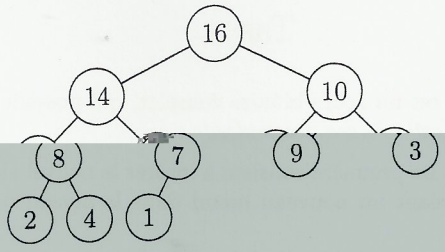
Correction : Les arbres 3, 6 et 7.

Question 2. Dessiner tous les tas possibles avec les éléments suivants : $1, 4, 7, 9$.

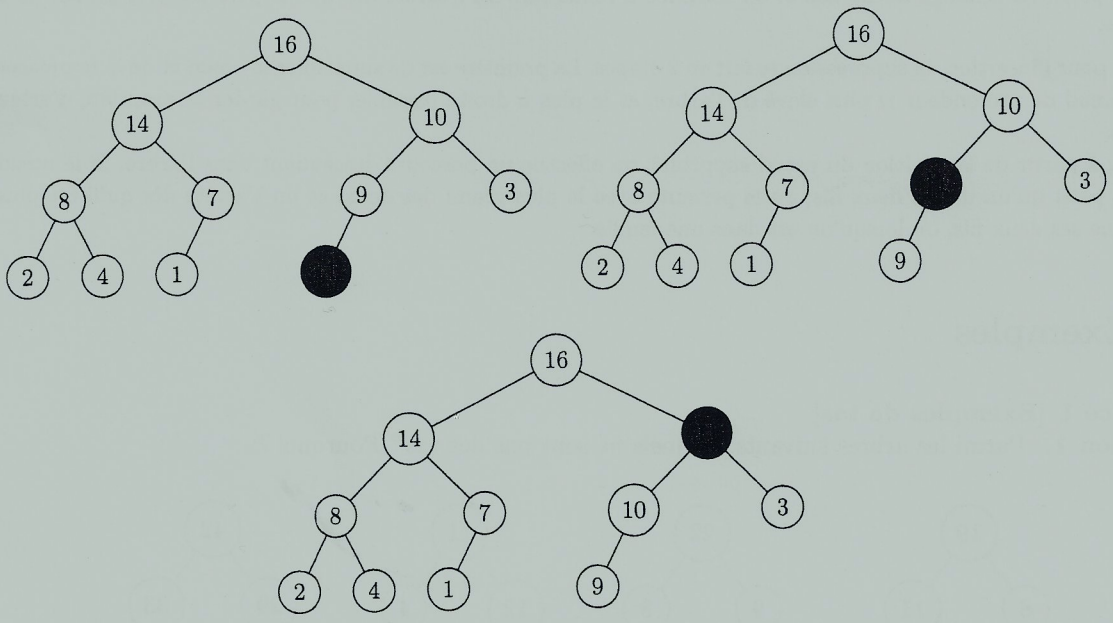
Correction :



Question 3. Entasser l'élément 11 dans le tas suivant :

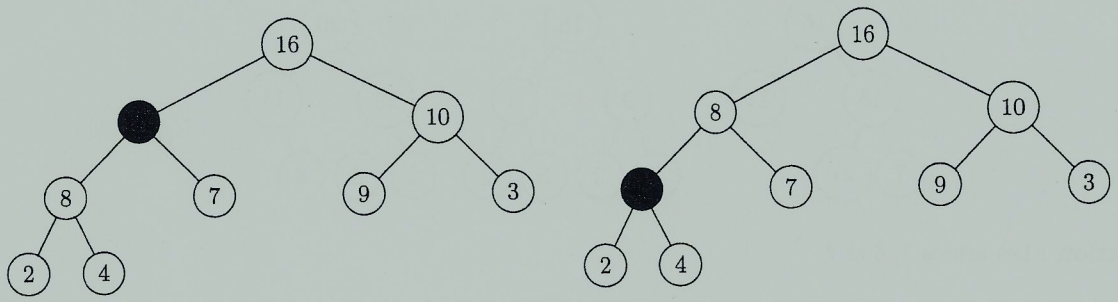


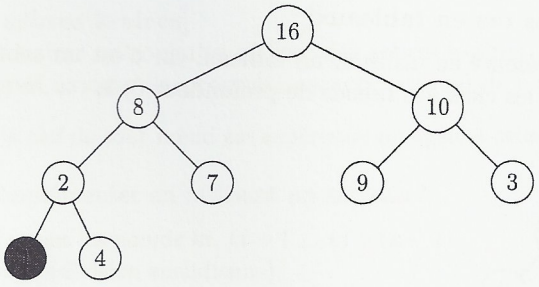
Correction :



Question 4. Supprimer l'élément 14 du tas de la question précédente.

Correction :





Exercice 2 [Propriétés classiques des tas]

Question 1. Quels sont les nombres minimal et maximal de nœuds d'un tas de hauteur h ?

Question 3. Montrer que pour un sous-arbre quelconque d'un tas, la racine du sous-arbre contient la plus grande valeur parmi les nœuds de ce sous-arbre.

Correction : Soit B un sous-arbre d'un tas et c la valeur de sa clef. Supposons qu'il existe un nœud dont la valeur v est strictement plus grande que c. Soit $v_0 = c, \ldots, v_p = n$ les valeurs des clefs des nœuds en suivant la branche reliant ces deux nœuds. Par définition des tas, on a $\forall i \in \{0, \ldots, p-1\}$, $v_i \geq v_{i+1}$, ce qui contredit l'affirmation précédente.

Question 4. Dans un tas où tous les éléments sont distincts, quelles sont les positions possibles pour le plus petit élément ?

Correction : Si l'on considère une branche entre la racine et une feuille f, il découle de la définition des tas et du fait que les clefs sont toutes distinctes que $clef(r) > clef(f)$. Donc le plus petit élément est dans une feuille du tas.

Réciproquement, il est clair que n'importe quelle feuille peut contenir le plus petit élément du tas.

Exercice 3 [Complexité sur les tas]

Question 1. Quelle est la complexité dans le pire des cas des algorithmes d'insertion et de suppression d'un élément dans un tas, en fonction de la hauteur du tas, puis de son nombre de nœuds ?

Correction : Dans chacun de ces deux algorithmes, la seconde étape coûte $O(h)$ comparaisons. Trouver le dernier élément dans la première étape coûte aussi $O(h)$. Donc la complexité de ces algorithmes est $O(h)$. Un tas étant un arbre complet, il vérifie : $h = O(\log(n))$. D'où la complexité en $O(h) = O(\log(n))$.

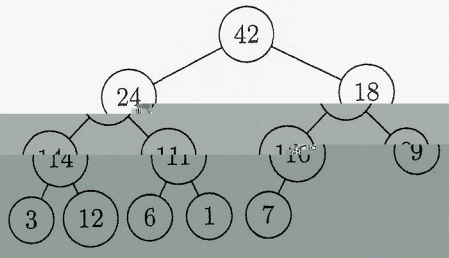
Une des utilisations classique des tas est le tri par tas. Son algorithme est le suivant : soit n entiers à trier, on les insère dans un tas initialement vide, puis on répète n fois l'opération de suppression de la racine. On a alors les entiers triés dans l'ordre décroissant.

Question 2. Quelle est la complexité de ce tri en moyenne et dans le pire des cas, en nombre d'appels aux opérations de base, puis en fonction de n ?

Correction : Dans tous les cas, on est obligé de procéder jusqu'au bout, donc les complexités en moyenne et dans le pire des cas sont les mêmes.
Ce tri coûte n insertions et n suppressions, soit $O(n \log(n))$ opérations.

Exercice 4 [Représentation des tas en tableaux]

Un tas peut être efficacement représenté en utilisant un tableau, car c'est un arbre complet. On remplit le tableau de gauche à droite avec les clefs des nœuds de profondeur 0, 1, ..., h pris de gauche à droite.



42	24	18	14	11	16	9	3	12	6	1	7			

Question 1. Soit un tas représenté sous forme de tableau. Si i est la position d'un nœud ayant un fils droit et un fils gauche, quels sont les indices de ses fils ?
Inversement, soit i l'indice d'un nœud autre que la racine. Quel est l'indice de son père ?

Exercice 5 [Tas avec des arbres k-aires]

Au lieu d'utiliser des arbres binaires on souhaite utiliser des arbres k-aires.

Question 1. Énoncer le nouvel axiome (la propriété caractéristique) de ces nouveaux tas

Correction : C'est la même : la clef de tout nœud est supérieure ou égale à celle de ses descendants.

Question 2. Comment les implémenter en utilisant un tableau ?

Correction : Les fils du nœud i sont les nœuds ki, $ki+1$ ... $ki+(k-1)$.
Le père du nœud i est le nœud k/i (division euclidienne).

Question 3. Donner les algorithmes d'insertion, et de suppression de la racine, avec cette structure de données

Correction :.

insérer(x :nombre) (T et n variables globales);
début
  $n \leftarrow n+1$ (augmenter la taille courante);
  $T[n] = x$ (insérer à la dernière feuille);
  $i = n$ (début de la remontée, i est l'indice du nœud courant);
  tant que $i > 1$ *et* $T[i] > T[i/k]$ **faire**
    Échanger $T[i]$ et $T[i/k]$;
    $i \leftarrow i/k$;
fin
Extraire racine();
début
  $r = T[n]$ (stocke la racine pour retour);
  Échanger $T[n]$ et $T[1]$ (échange tête et dernière feuille);
  $n \leftarrow n-1$ (supprime la dernière feuille);
  $i = 1$ (début de la descente);
  $d \leftarrow vrai$ (drapeau pour arêter la descente);
  tant que $i \leq n/k$ *et* d **faire**
    $j \leftarrow$ indiceMax($T, ki, max(n, ki+k-1)$);
    (indiceMax(T, a, b) retourne l'indice du max de T dans $[a, b]$);
    si $T[i] < T[j]$ **alors**
      Échanger $T[i]$ et $T[j]$;
      $i \leftarrow j$;
    sinon
      $d \leftarrow faux$
  retourner r;
fin