Entraînement : Arbres en structure de tas

Pour tous les exercices, la grille d'évaluation est la suivante.

Structure de données tas.

A (20)	Exemple tas / tableau correct
D (8)	Structure comprise mais erreurs (d'inattention ?)
E (1)	Exemples faux

Algorithme.

A (20)	L'algorithme répond correctement au problème posé, il est écrit de façon claire et la complexité est correcte.
B (16)	L'algorithme contient quelques erreurs mais reste globalement juste et la complexité est correcte.
C (11)	L'algorithme fonctionne globalement mais complexité non optimale ou complexité calculée fausse.
D (8)	L'algorithme ne fonctionne pas.
E (1)	Algorithme quasi inexistant ou ne répondant pas du tout au problème posé.

Exercice 1.

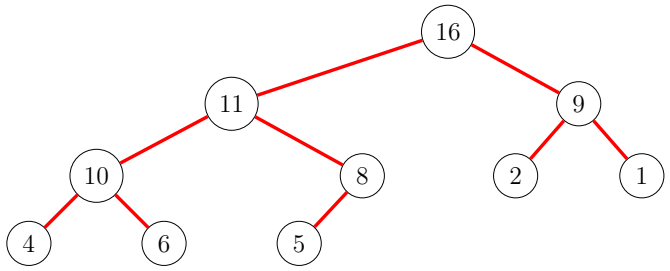
On rappelle qu'un tas est un **arbre binaire parfait décroissant** :

— tous les niveaux sauf le dernier sont remplis au maximum et les feuilles du dernier niveaux sont alignées à gauche,

— la valeur d'un nœud est toujours supérieure égale aux valeurs de ses fils.

La structure d'arbre binaire parfait permet au tas d'être stocké dans un tableau.

(1) Donner le tableau correspondant au tas suivant



(2) Dessiner l'arbre correspondant au tableau suivant

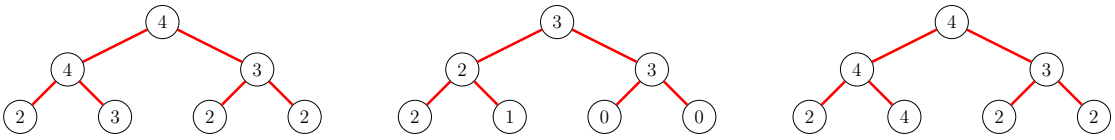
indice	0	1	2	3	4	5	6	7	8
valeur	5	4	2	3	2	1	1	2	1

On se place dans le contexte suivant : on possède un ensemble de serveurs qui gèrent des processus. Chaque serveur possède un certain nombre de *cœurs* libres pour traiter de nouveaux processus. On stocke la disponibilité des serveurs dans un tableau sous forme de tas, par exemple

0	1	2	3
4	2	2	0

signifie que j'ai 4 serveurs. Le serveur 0 possède 4 cœurs disponibles, les serveurs 1 et 2 en possèdent chacun 2 et le serveur 3, 0. Le serveur 0 sera toujours celui avec le plus de cœurs disponibles.

(3) Considérer les 3 exemples suivants :



Pour chacun des exemples,

(a) Diminuer la valeur de `T[0]` de 1 (on lui affecte un processus) et regarder si le résultat obtenu est un tas.

(b) Si ce n'est pas le cas, décrire le processus à suivre pour *corriger* le résultat et obtenir un tas.

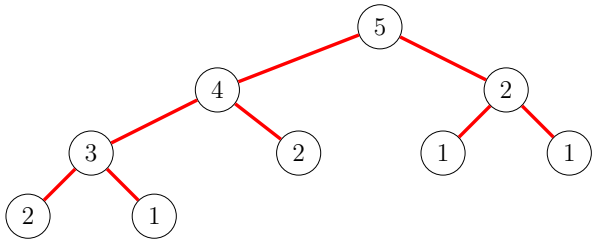
(4) Implanter une fonction qui réalise de façon générale l'algorithme décrit sur les exemples précédents. Quelle est sa complexité ?

Solution

(1)

0	1	2	3	4	5	6	7	8	9
16	11	9	10	8	2	1	4	6	5

(2)



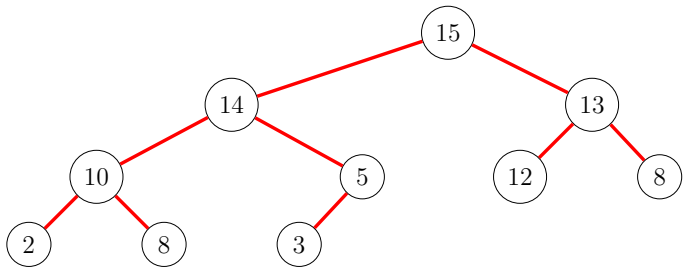
(3) Aucun des trois ne reste un tas : il faut échanger la valeur avec son fils maximal puis recommencer récursivement sur le fils échangé.

(4) Une solution possible

```
Input : Tableau T
Procédé :
    i <- 0
    Tant que 2*i+1 < T.taille :
        filsMax <- 2*i+1
        Si 2*i +2 < T.taille et T[2*i+2] > T[2*i+1]:
            filsMax <- 2*i+2
        Si T[i] < T[filsMax]:
            Echanger T[i], T[filsMax]
            i <- filsMax
```

Exercice 2 (Partiel 2018)**.**
On rappelle qu'un tas est un **arbre binaire parfait décroissant** :
— tous les niveaux sauf le dernier sont remplis au maximum et les feuilles du dernier niveaux sont alignées à gauche,
— la valeur d'un nœud est toujours supérieure égale aux valeurs de ses fils.
La structure d'arbre binaire parfait permet au tas d'être stocké dans un tableau.

(1) Donner le tableau correspondant au tas suivant

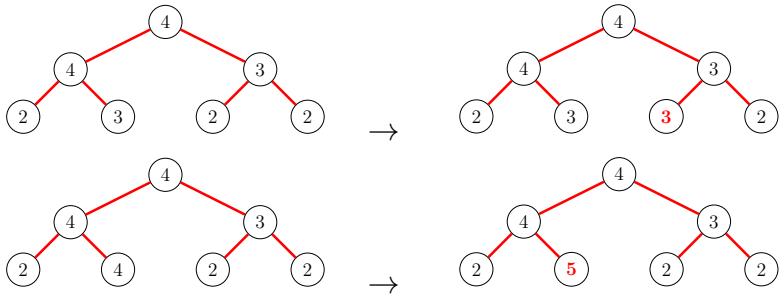


(2) Dessiner l'arbre correspondant au tableau suivant

indice	0	1	2	3	4	5	6	7	8
valeur	5	4	2	3	2	1	1	2	1

On utilise un tableau organisé en tas pour représenter les disponibilités dans des salles de jeu en ligne : chaque case du tableau correspond à une salle et sa valeur correspond aux nombres de places disponibles pour accueillir un nouveau joueur. La salle avec le plus grand nombre de places disponibles est au sommet du tas.

(3) Un joueur peut quitter une salle à tout moment et dans ce cas, la valeur dans le tableau augmente. Observer les 2 exemples suivants :



Dans quel cas le résultat obtenu est-il un tas ? Proposer une solution pour l'autre cas : décrire le principe et dessiner l'arbre obtenu.

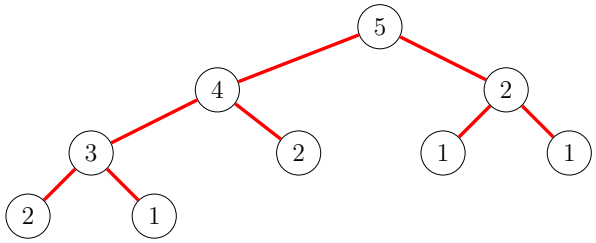
(4) Implanter la fonction `QuitteSalle(T,i)` qui augmente la valeur de `T[i]` pour signifier qu'une place s'est libérée et modifie le tableau pour que la structure de tas soit conservée. Quelle est sa complexité ?

Solution

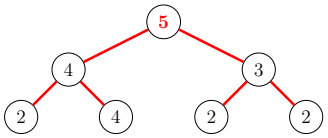
(1)

0	1	2	3	4	5	6	7	8	9
15	14	13	10	5	12	8	2	8	3

(2)



(3) Dans le deuxième cas, l'arbre obtenu n'est plus un tas. Il faut échanger le nœud 5 avec son père, puis à nouveau récursivement avec la racine.



(4)

```
QuitteSalle
Input :
    - un tableau T de taille n
    - un entier i
Procédé :
    Si i < n:
        T[i] <- T[i] - 1
        pere <- (i-1)/2
        Tant que pere >= 0 et T[i] > T[pere]:
            Echanger T[pere],T[i]
            i <- pere
            pere <- (i-1)/2
```

Complexité : $O(\log(n))$ (hauteur de l'arbre)