Gdalia Thibault
Sup D2

Devoir Maison

13 Mai 2014

Epita Promo 2019

Exercice 1

question 1 : représentation par tas de la figure 1.

5	8	12	9	11	20	15	18	10	13

question 2 :

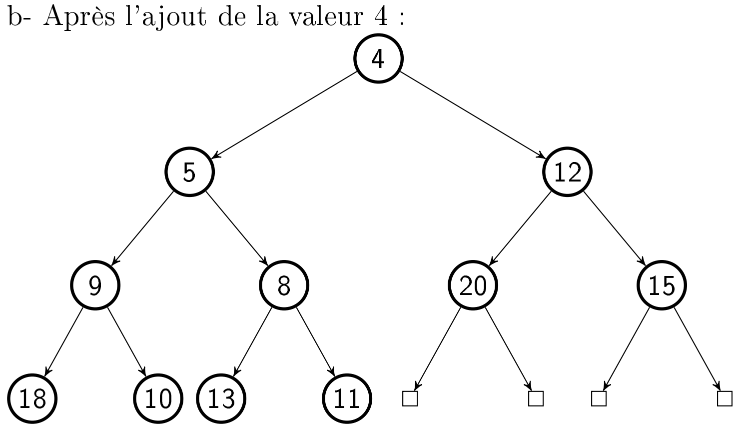
a- La racine se trouve a place 1 du vecteur.
b- Les fils d'un nœud se trouve à 2x(place du nœud) et à 2x(position du nœud) + 1
c- Pour retrouver le père d'un noeud il faut faire (position du nœud) div 2. Où div est la division entière
d- Le nœud est une feuille dans le cas ou les cases du vecteur 2x(position du nœud) et 2x(position du nœud) + 1 sont vide ou si elle n'existe pas
e- Un nœud est un point simple si l'une des cases, correspondant a ses fils, dans le vecteur est vide.

Exercice 2 :

question 1 : Ajout

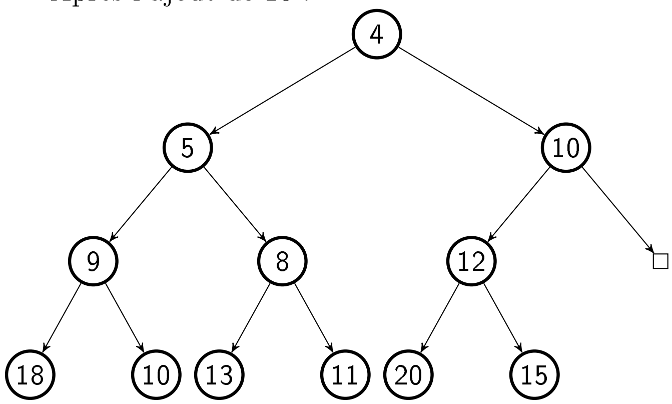
a- On ajoute l'élément en feuille, puis on le fais remonter en l'échangeant avec son père jusqu'a ce qu'il se trouve a sa place. c'est-à-dire lorsque son père est inférieur au nouvel élément.

b- Après l'ajout de la valeur 4 :



K	J	I	B	Q	C	G	D	F	E	H
4	5	12	9	8	20	15	18	10	13	11

Après l'ajout de 10 :



K	J	L	B	Q	I	D	F	E	H	C	G
4	5	10	9	8	12	18	10	13	11	20	15

c- algorithme d'ajout dans un tas :

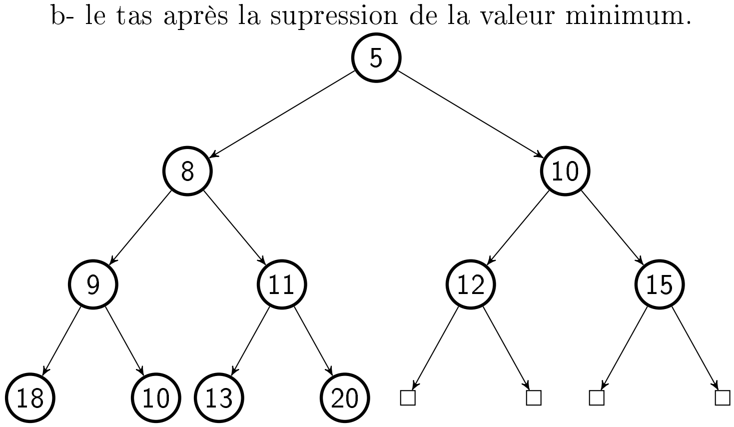
```
algorithme procedure ajout
    parametres globaux
        t_tas tas
    parametres locaux
        t_element elt
        reel val
    variables
        t_tas tmp
debut
    taille <- taille + 1
    place <- taille
    tas[place] <- (elt,val)
    tant que tas[place] < tas[place div 2] faire
        tmp[1] <- tas[place div 2]
        place <- place div 2
        tas[place] <- tas[taille]
        tas[taille] <- tmp[1]
    fin tant que
fin algorithme procedure ajout
```

d- la complexité de cet algo est O(log(n))

2-supression

a- La valeur minimum est la première valeur du tas, donc la racine de l'arbre. Pour la supprimer on la passe en feuille, pour cela on l'échange avec le dernier élément du tas. Puis on remet la nouvelle racine a sa place dans le tas pour que l'on continue a respecter la relation d'ordre.

b- le tas après la supression de la valeur minimum.



J	Q	F	B	H	I	G	D	L	E	C
5	8	10	9	11	12	15	18	10	13	20

c- Alorithme de suppression

```
algorithme procedure suppression
    parametres globaux
        t_tas tas
    variables
        entier place
debut
    swap(tas.elt[1], tas.elt[taille])
    taille <- taille - 1
    place <- 1
    tant que (tas[place] > tas[2*place] ou tas[place] > tas[2*place + 1]) faire
        si tas[2*place] < tas[2*place + 1] alors
            swap(tas[place], tas[2*place])
            place <- 2*place
        sinon
            swap(tas[place], tas[2*place + 1])
            place <- 2*place + 1
```

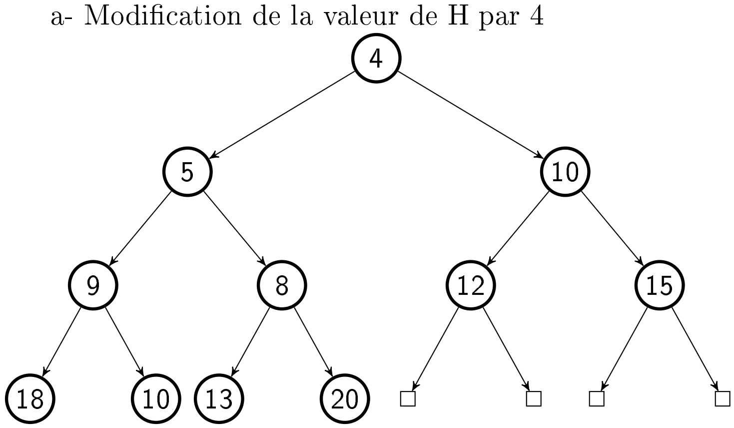
```
        fin si
    fin tant que
fin algorithme procedure suppression
```

d- la complexite de cette algorithme est de l'autre de O(log(taille du tas))

Exercice 3 :

question 1 : Minimisation

a- Modification de la valeur de H par 4



H	J	F	B	Q	I	G	D	L	E	C
4	5	10	9	8	12	15	18	10	13	20

b- algorithme modif_tas :

```
algorithme procedure modif_tas
    parametres globaux
        t_tas t
    parametres locaus
        t_element elt
        reel x
    variables
        entier r
debut
    r <- 1
    tant que tas.elt[r] <> elt faire
```

```
        r <- r+1
    fin tant que
    tas[r] <- (elt, x)
    tant que tas[r] < tas[r div 2]
        swap(tas[r], tas[r div 2])
        r <- r div 2
    fin tant que
fin algorithme modif_tas
```

Question 2 : Optimisation

a-La COmplexité d'un algorithme de recherche est O(taille du tas).

b- Si on avait l'ancienne valeur de l'élément à modifié on pourrait le trouvait plus rapidement et la complexité passerait à O(log(taille du tas)).