

## Séance de TD 03

### Exercice 1

#### Fonction EntasserMax(A, i)

Soit un tableau A, et un indice i, A[i] est la clé associée au  $i^{\text{ième}}$  élément de A. Quand la fonction EntasserMax est appelée, on suppose que les arbres binaires enracinés en Gauche(i) et Droite(i) sont des tas max. La fonction EntasserMax(A, i) consiste à faire “descendre” la valeur A[i] dans le tas max de manière que le sous-arbre enraciné en i devienne un tas max.

1. Donner l’algorithme correspondant à la fonction EntasserMax
2. Illustrer l’action de EntasserMax(A,2), avec A = [16,4,10,14, 7,9,3,2,8,1].
3. Idem pour EntasserMax(A,3) avec A = [27,17,3,16,13,10,1,5,7,12,4,8,9,0].

#### Code de l’algorithme

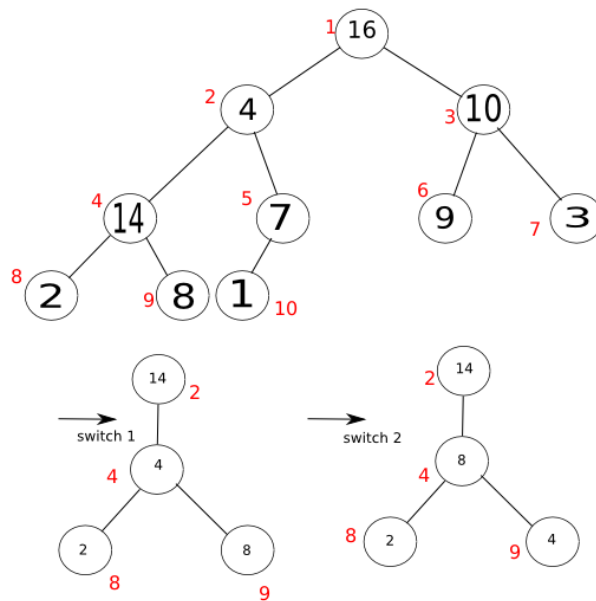
```
Si A[i] > A[j] et j <= taille(A)
  max <- i
Sinon
  max <- j
Finsi

Si A[k] > A[max] et k <= taille(A)
  max <- k
Finsi

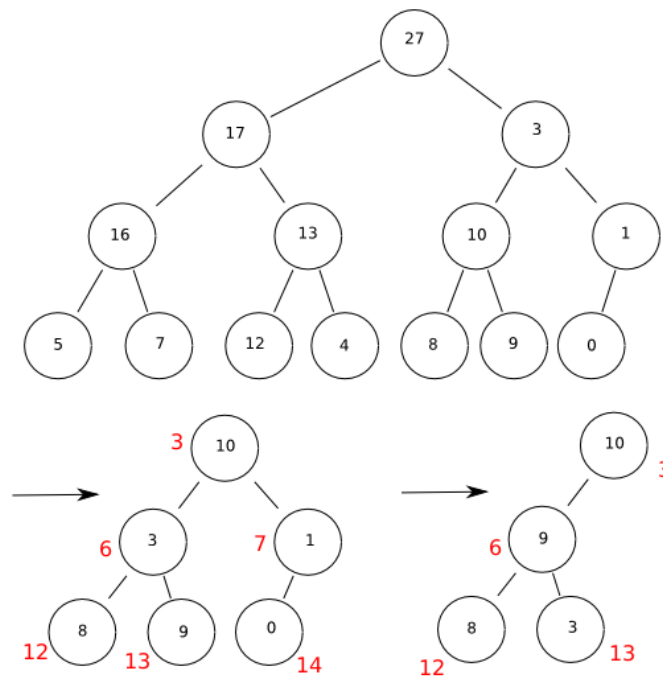
Si max != i
  échanger A[i] et A[max]
  EchangerMax(A, max)
Finsi
```

#### Applications

A = [16,4,10,14, 7,9,3,2,8,1]



$A = [27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$



**Exercice 2**

Fonction ConstruireTasMax(A, i)

Dans la construction d'un tas, la fonction `EntasserMax` est utilisée à l'envers pour convertir un tableau  $A[1, \dots, n]$ , avec  $n = \text{longueur}(A)$ , en tas max.

Supposant que dans un arbre binaire, qui représente un tableau à  $n$  éléments, les feuilles sont les noeuds indexés par  $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \dots, n$ . Ces feuilles sont donc des tas à 1 élément.

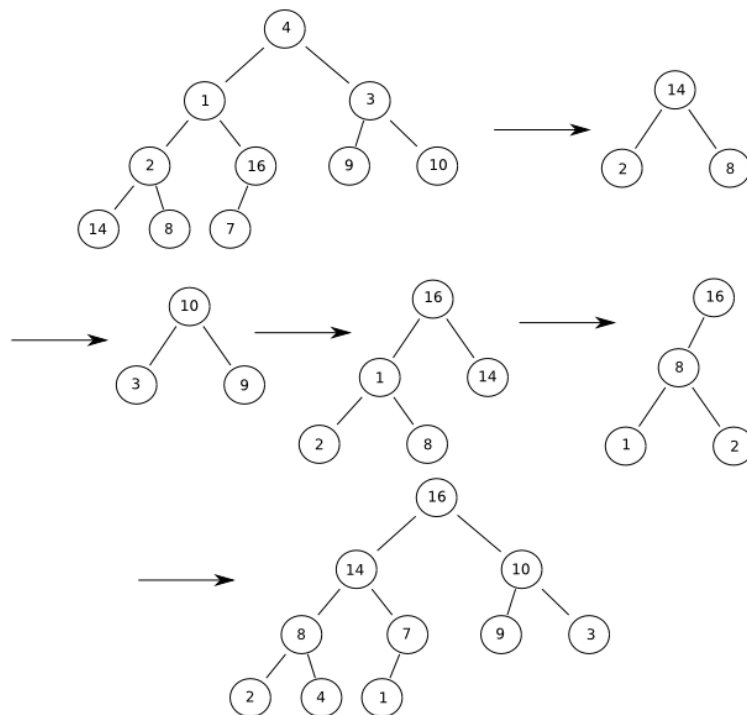
1. Donner l'algorithme correspondant à la construction d'un tas Max à partir d'un tableau donné.
2. Illustrer l'action de cet algorithme sur le tableau  $A = [4, 1, 3, 2, 16, 9, 10, 14, 8, 7]$ .
3. Dans l'algorithme `ConstruireTasMax`, pourquoi fait-on décroître l'indice de boucle, au lieu de le faire croître ?

### Code de l'algorithme

```
Pour i de PartieEntière(taille(A) / 2) à 1 Faire
  EntasserMax(A, i)
FinPour
```

### Application

On utilise le tableau suivant :  $A = [4, 1, 3, 2, 16, 9, 10, 14, 8, 7]$ .



La boucle dans l'algorithme va en décroissant afin de diminuer la complexité.

## Exercice 3

### Files de priorité

Une file de priorité est une structure de données qui permet de gérer un ensemble  $E$  d'éléments, dont chacun a une valeur clé associée. Une file de priorité max reconnaît les trois opérations suivantes :

- Insérer( $E,x$ )
- Maximum( $E$ )
- ExtraireMax( $E$ )

Donnez l'algorithme correspondant à chaque opération. Donnez la complexité associée à chaque algorithme.

### Texte des algorithmes

Maximum( $A$ )

```
retourner A[1]
```

ExtraireMax( $A$ )

```
Si taille(A) <= 1 Alors
```

```
retourner ERREUR
```

```
Sinon
```

```
max <- A[1]
```

```
A[1] <- A[taille(A)]
```

```
taille(A) <- taille(A) - 1
```

```
EntasserMax(A,i)
```

```
retourner max
```

```
Finsi
```

```
Fin
```

InsererTasMax( $A,x$ )

```
taille(A) <- taille(A) + 1
```

```
A[taille(A)] <- x
```

```
i <- taille(A)
```

```
Tant que (i > 1) Et (A[Parent(i)] < A[i]) Faire  
permuter A[i] Et A[Parent(i)]
```

```
    i <- parent(i)
  FinTantQue
Fin
```

### Complexités

- $\text{Max}(A)$  est de complexité  $O(1)$ .
- $\text{ExtraireMax}(A)$  est de complexité  $O(\log(n))$ .
- $\text{Inserer}(A,n)$  est de complexité  $O(\log(n))$ .