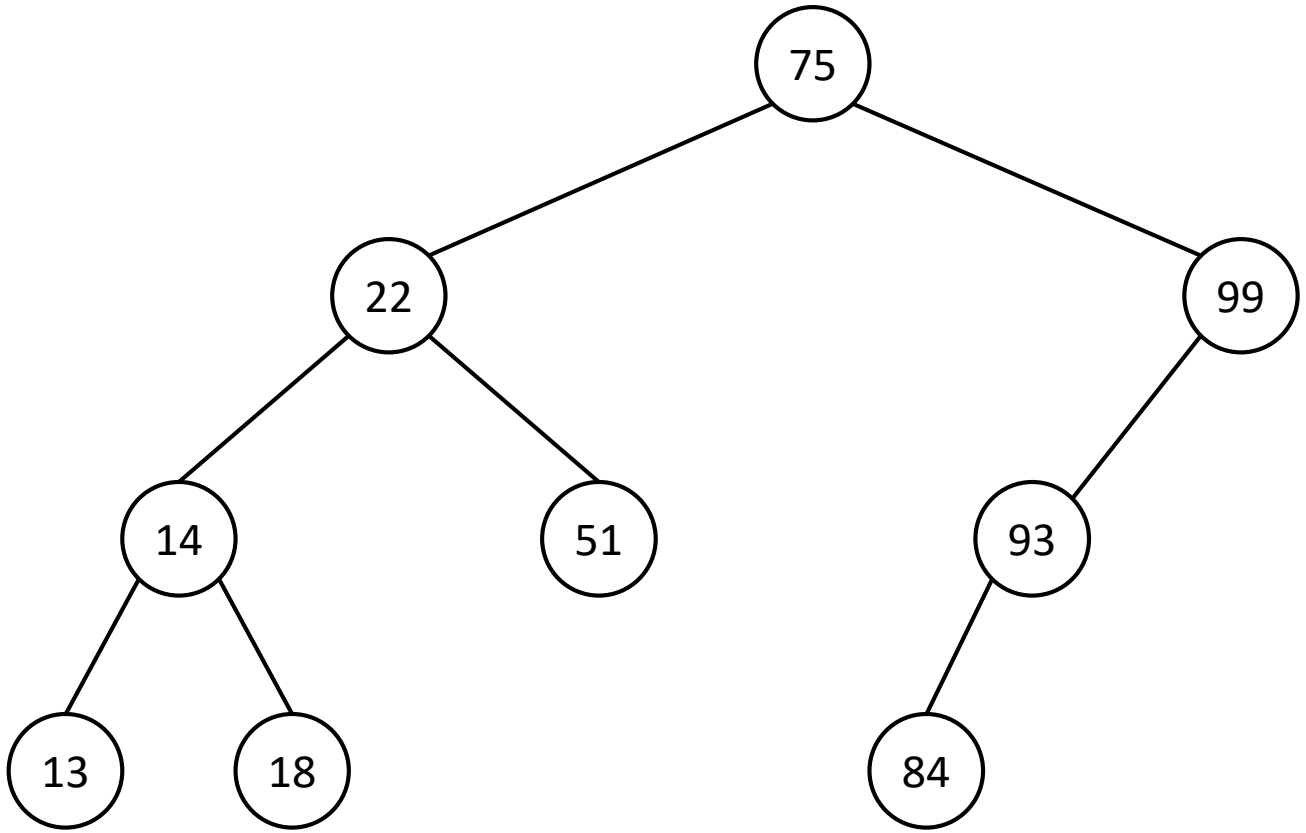
Algorithmique et structures de données - NF16

Arbres binaires - Tas

Mohamed Ali KANDI

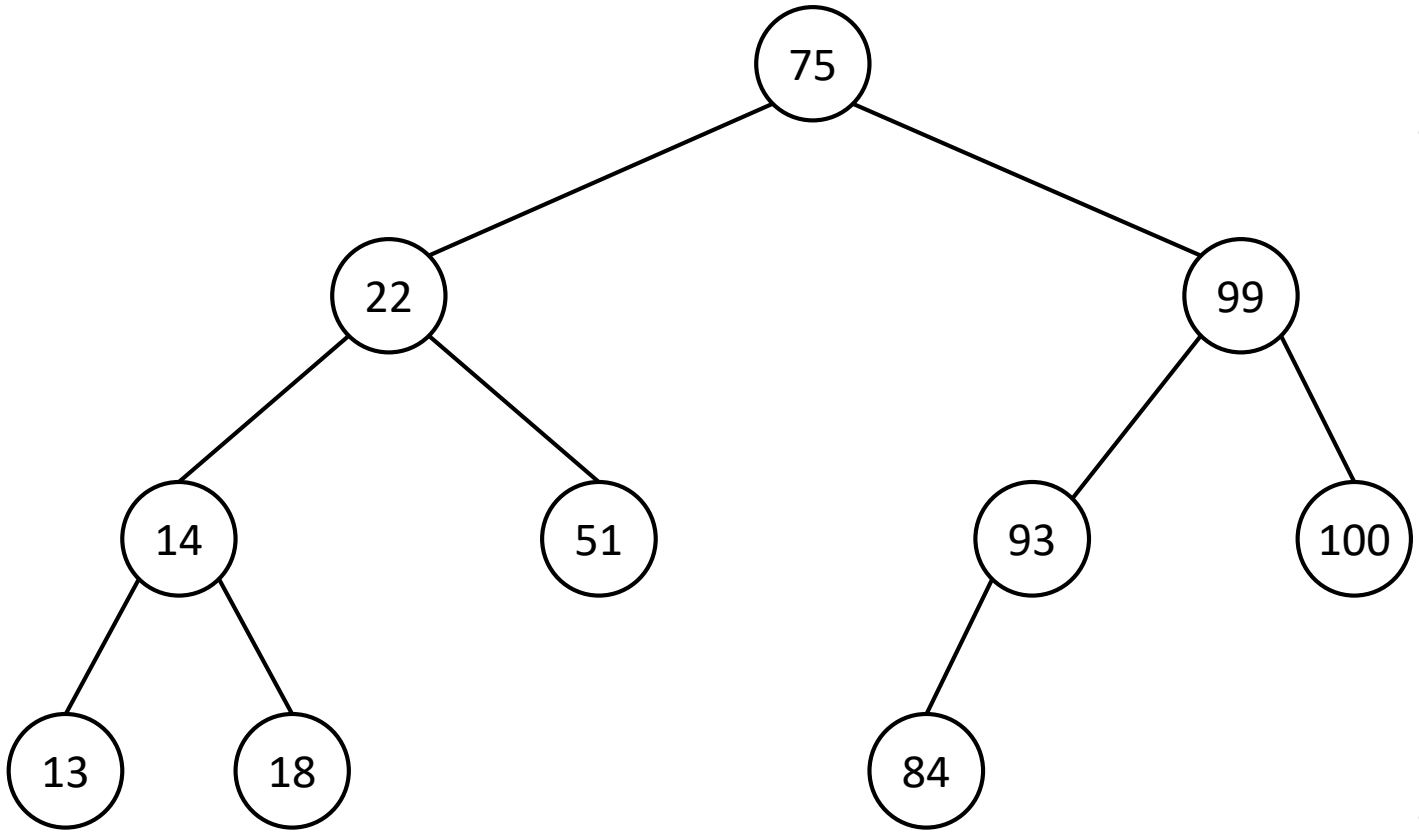
mohamed-ali.kandi@hds.utc.fr

Les arbres de recherche binaires



- Un nœud ne peut avoir que deux fils au maximum.
- Les clés sont ordonnées de la façon suivante:
  - Chaque noeud du sous-arbre gauche a une clé inférieure (ou égale) à celle du noeud considéré.
  - Chaque noeud du sous-arbre droit a une clé supérieure (ou égale) à celle du noeud considéré.

Les AVLs

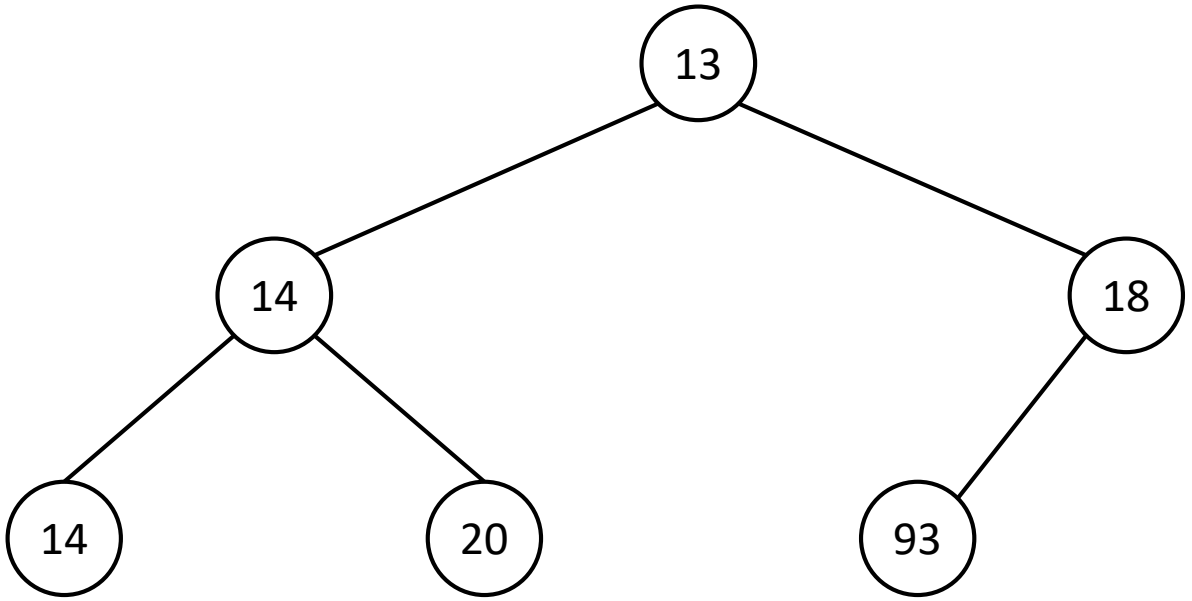


Recherche ➔ $O(log_2(n))$

- Un nœud ne peut avoir que deux fils au maximum.
- Les clés sont ordonnées de la façon suivante:
  - Chaque noeud du sous-arbre gauche a une clé inférieure (ou égale) à celle du noeud considéré.
  - Chaque noeud du sous-arbre droit a une clé supérieure (ou égale) à celle du noeud considéré.
- Pour tout nœud N :

$$|h\big(gauche(N)\big) - h\big(droit(N)\big)| \leq 1$$

Les tas

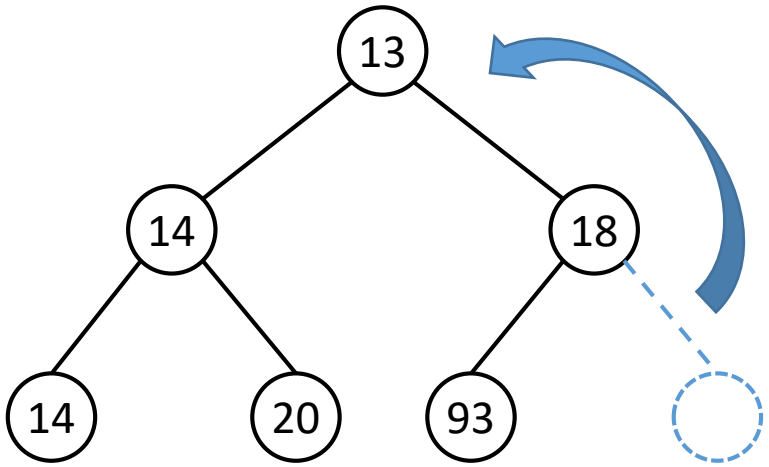


Tas Min

- Un nœud ne peut avoir que deux fils au maximum.
- Tous les niveaux sont totalement remplis, sauf le dernier ou les nœuds sont regroupés le plus à gauche possible (arbre parfait).
- $\begin{cases} \text{Tas Min} : \text{Val(Pere)} \leq \text{Val(Fils)} \\ \text{Tas Max} : \text{Val(Pere)} \geq \text{Val(Fils)} \end{cases}$

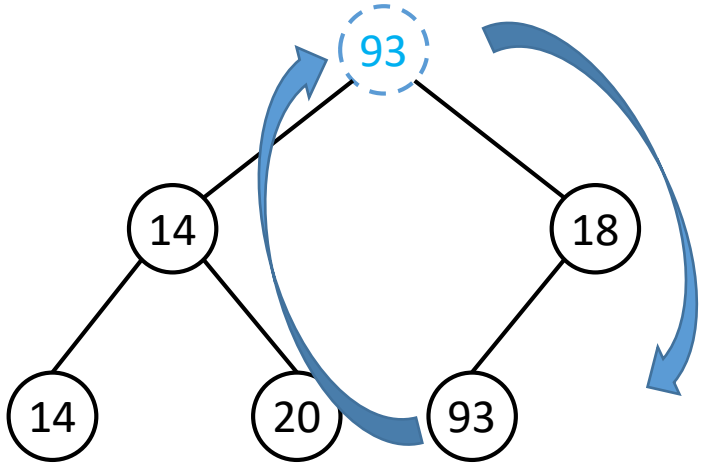
Les tas

Insertion



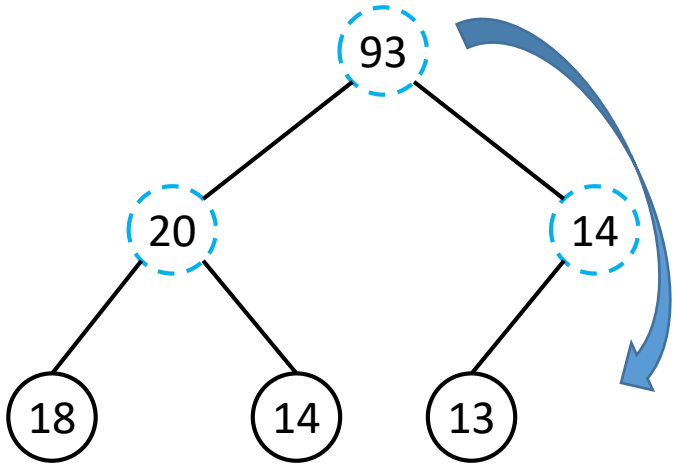
- Insérer à la fin
- Remonter

Suppression (Racine)



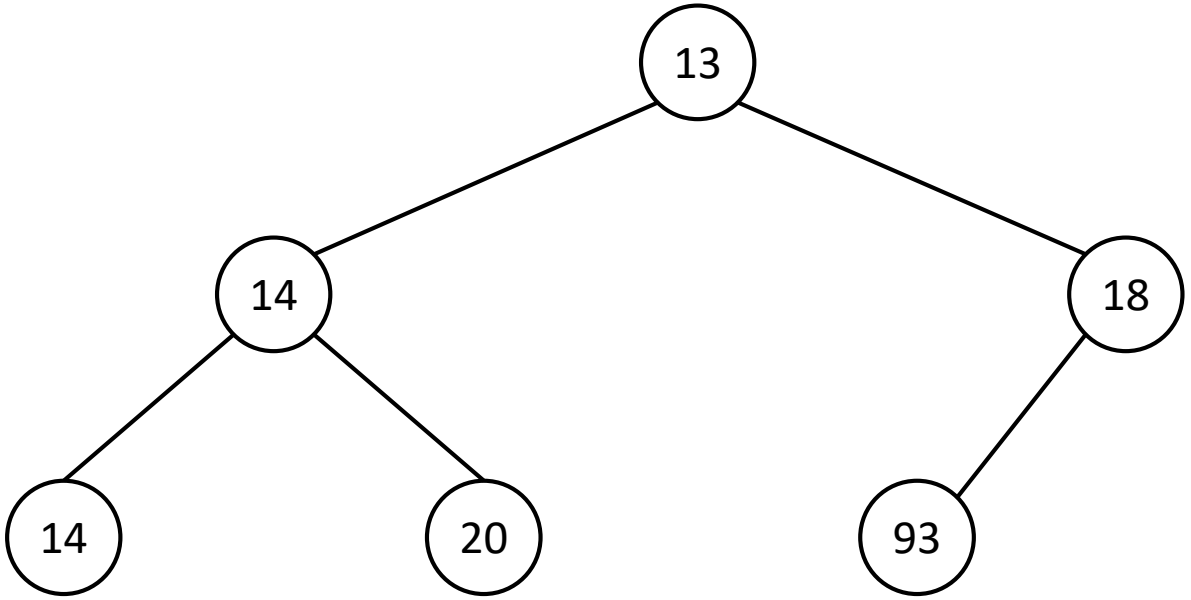
- Remplacer la racine par le dernier élément
- Entasser

Construction



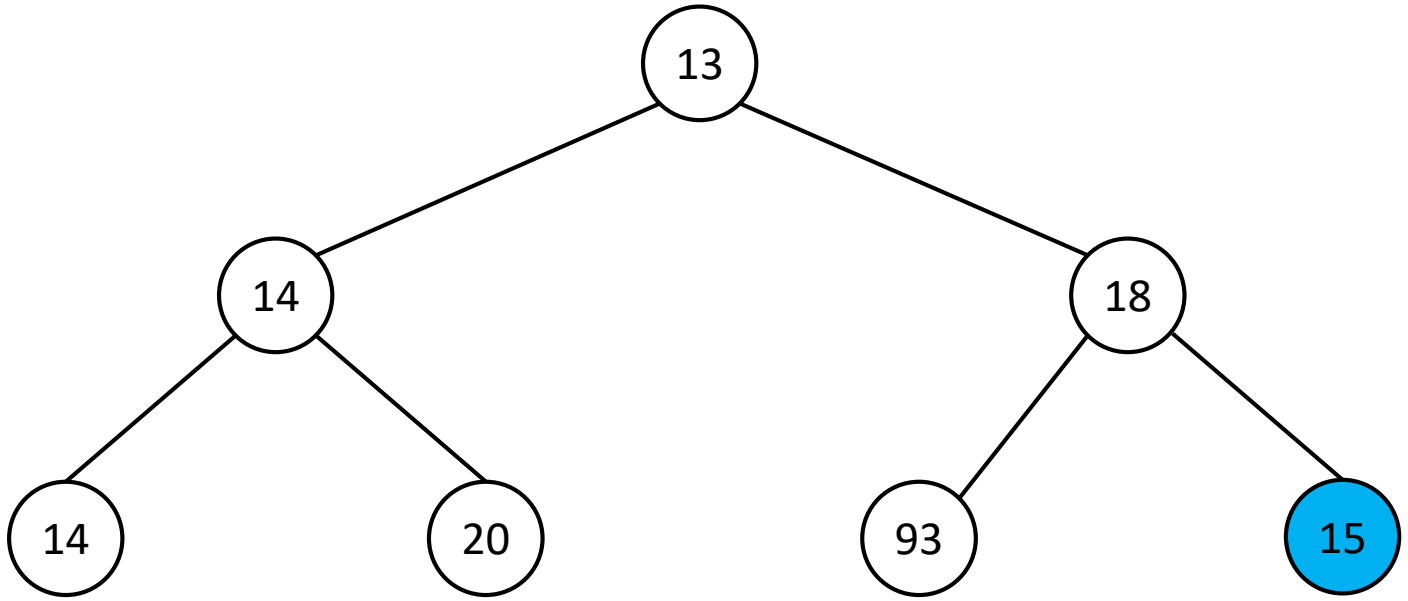
- Entasser tous les éléments sauf les feuilles

Les tas (Insertion)



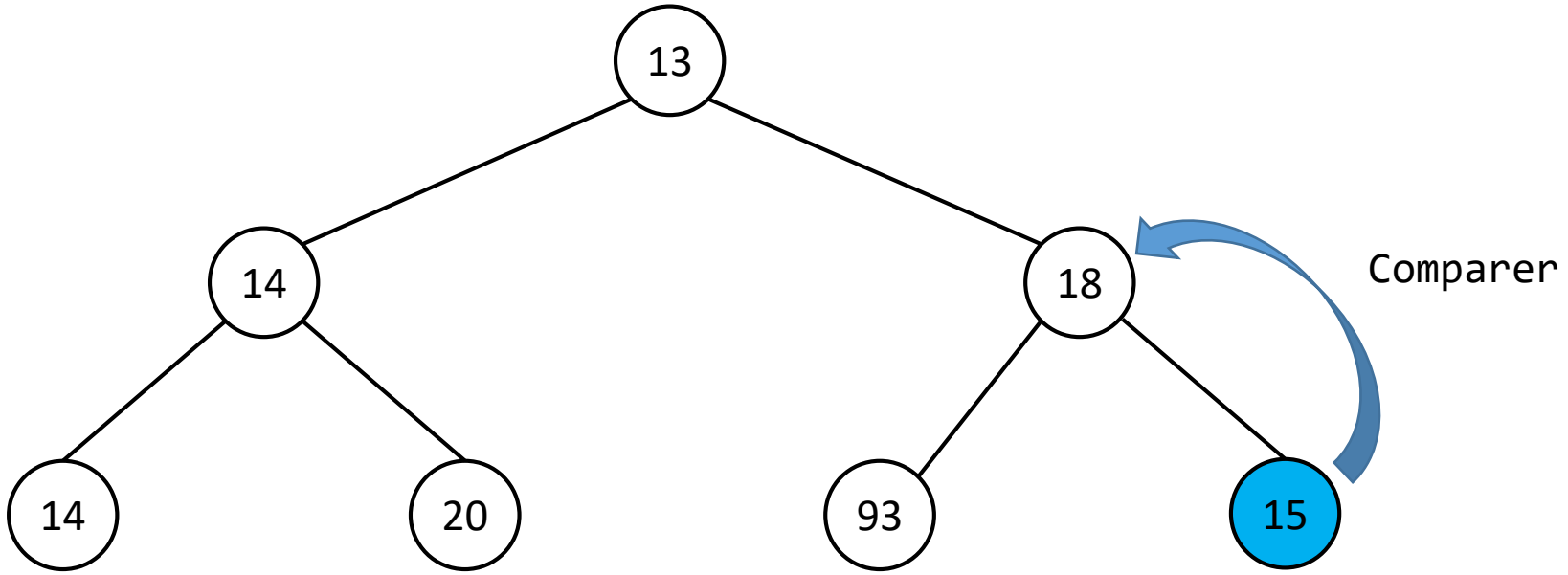
Insérer 15

Les tas (Insertion)



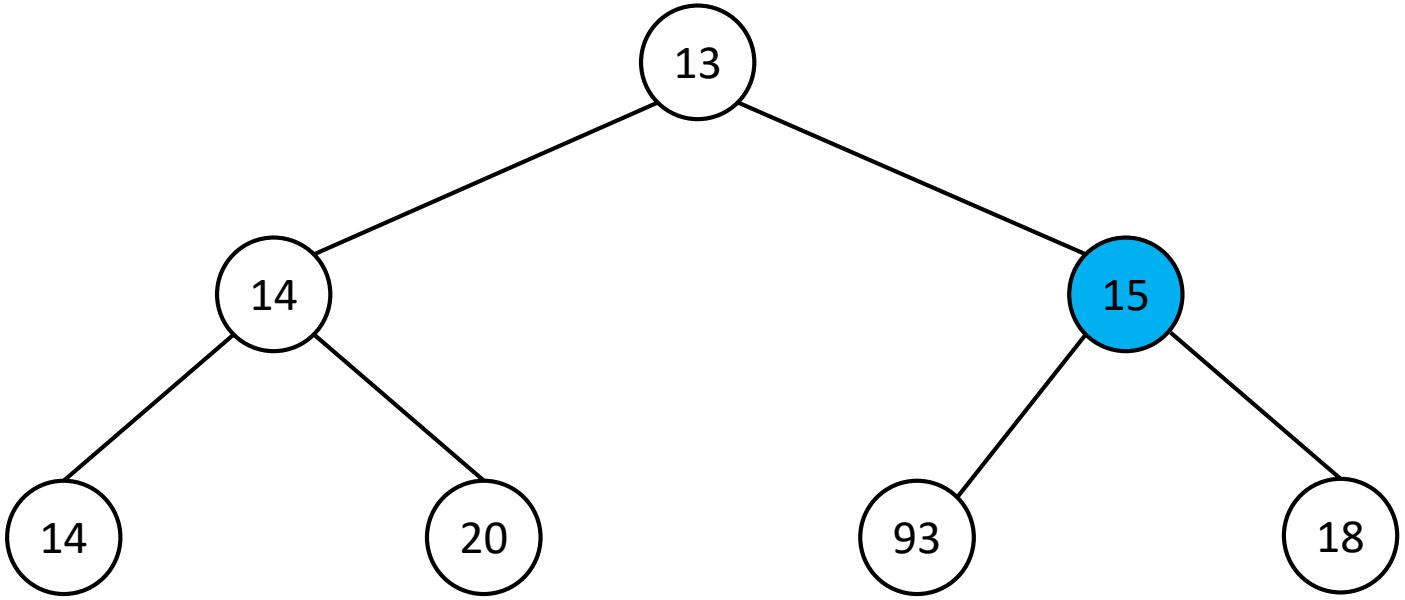
Insérer 15

Les tas (Insertion)



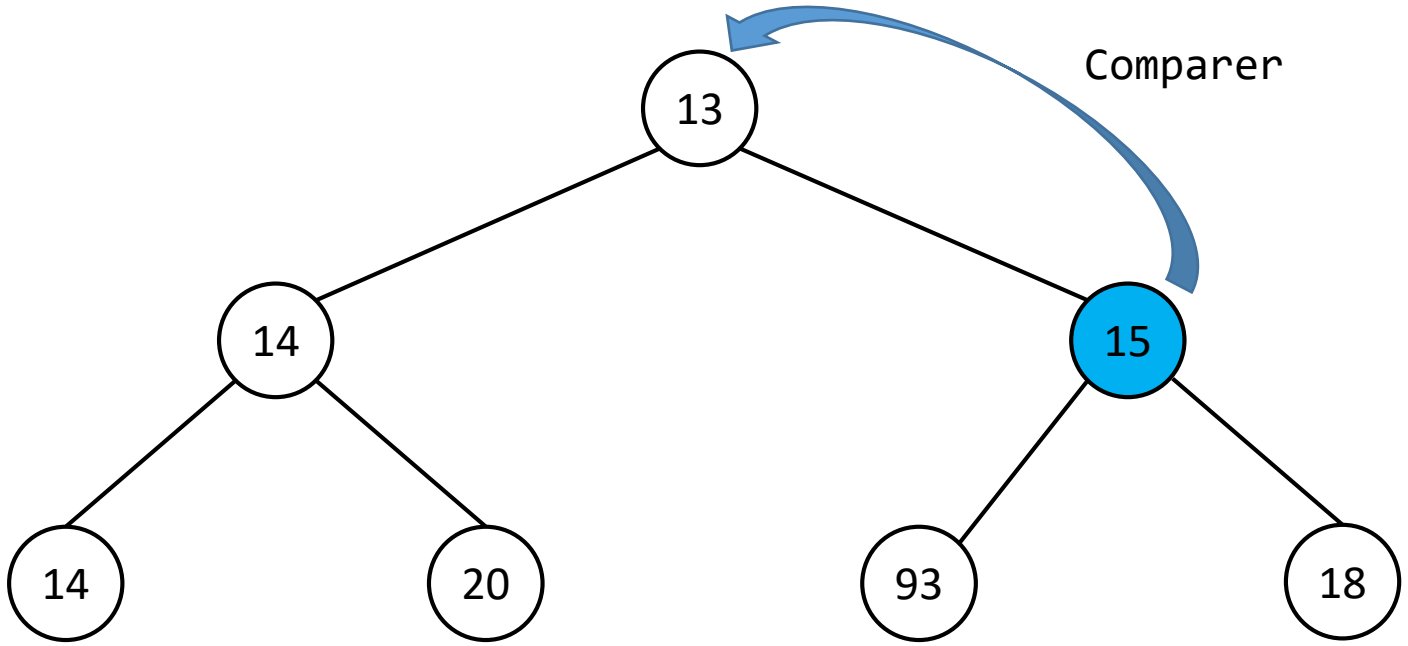
Insérer 15

Les tas (Insertion)



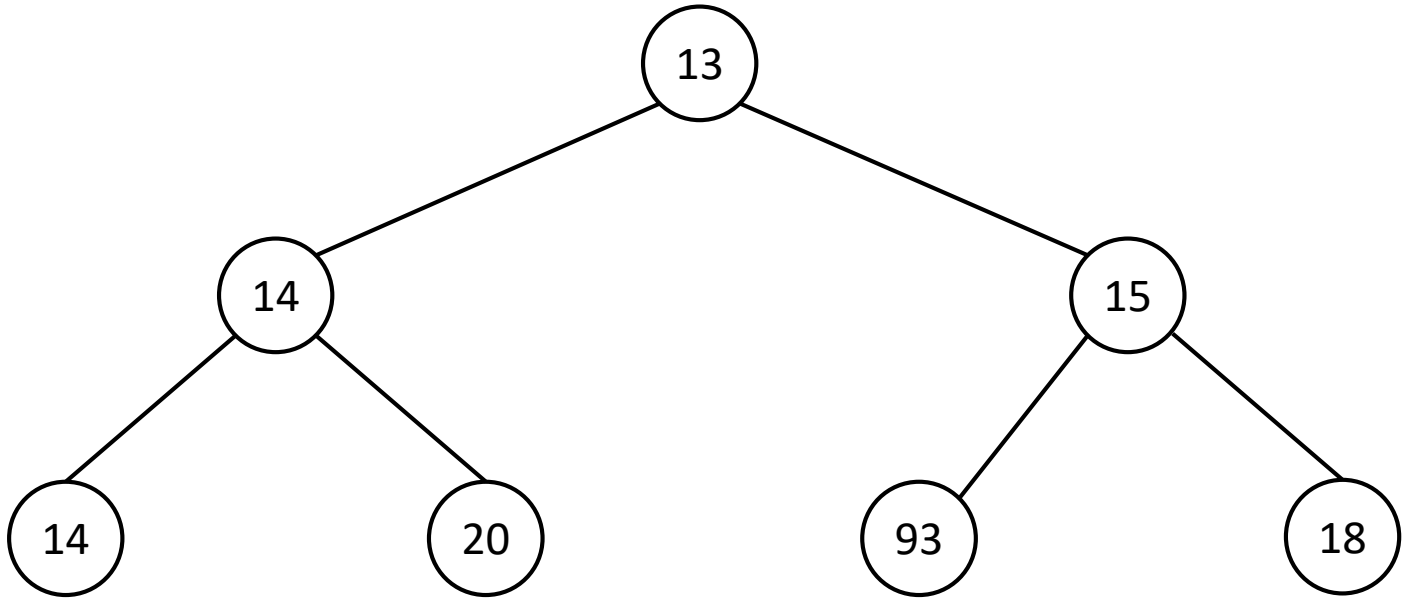
Insérer 15

Les tas (Insertion)



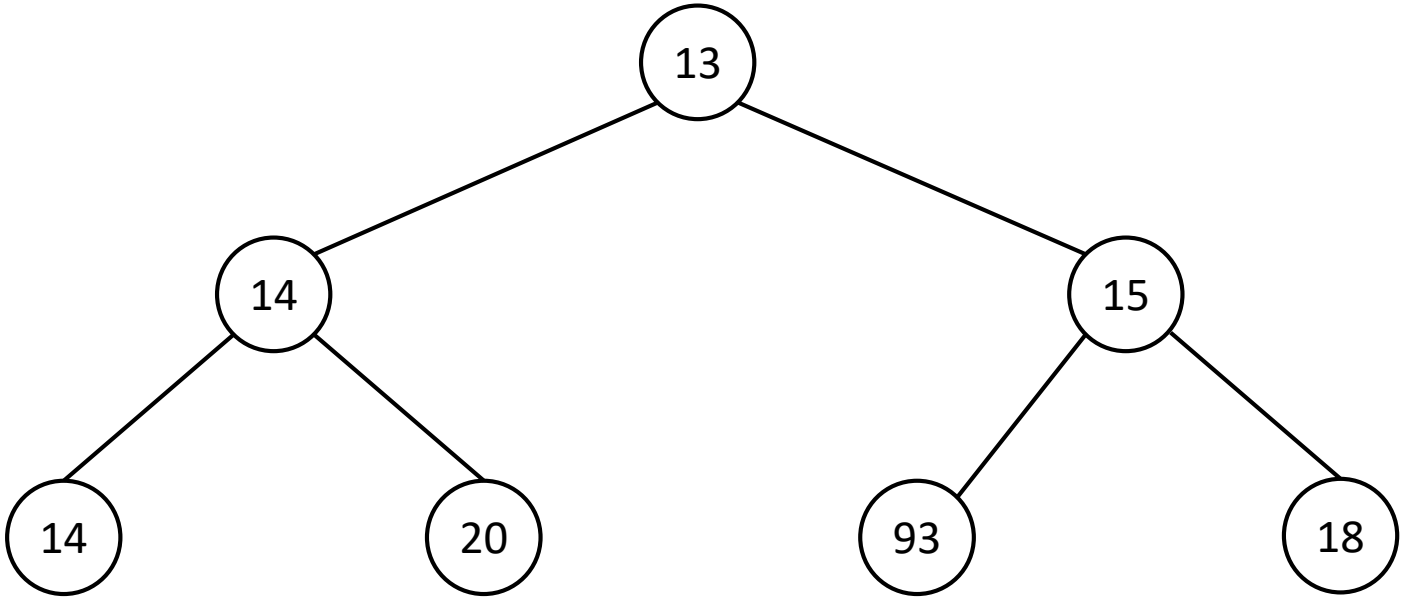
Insérer 15

Les tas (Insertion)

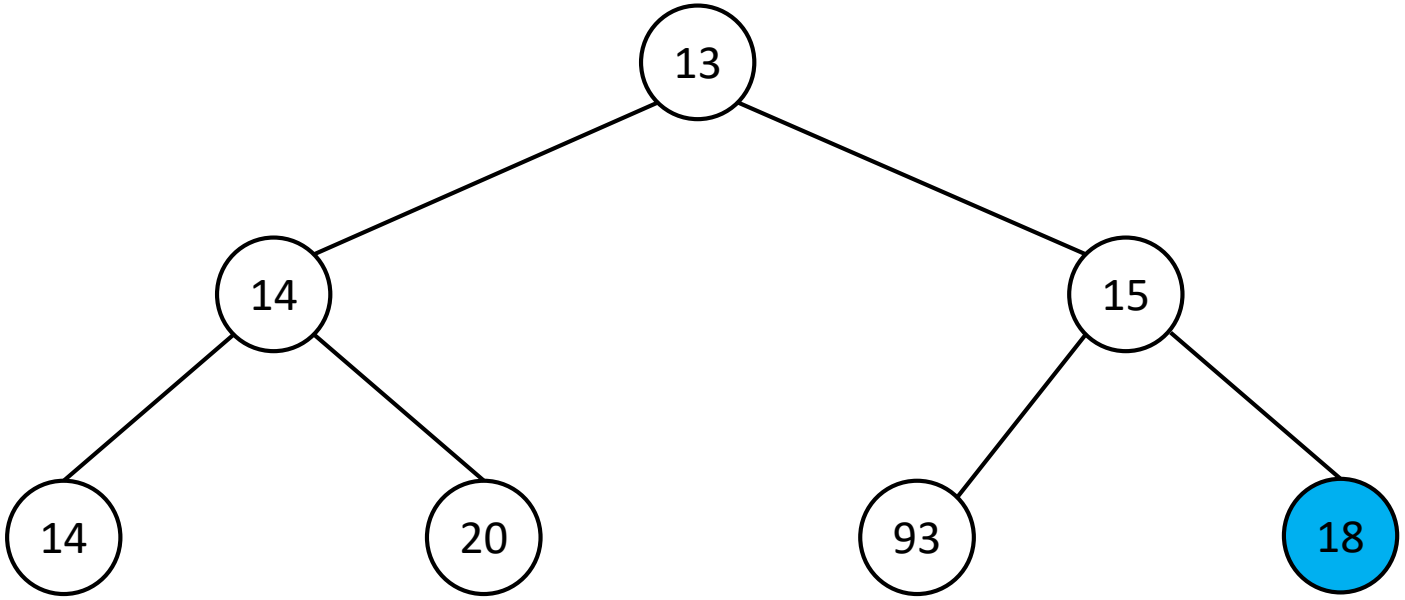


Insérer 15

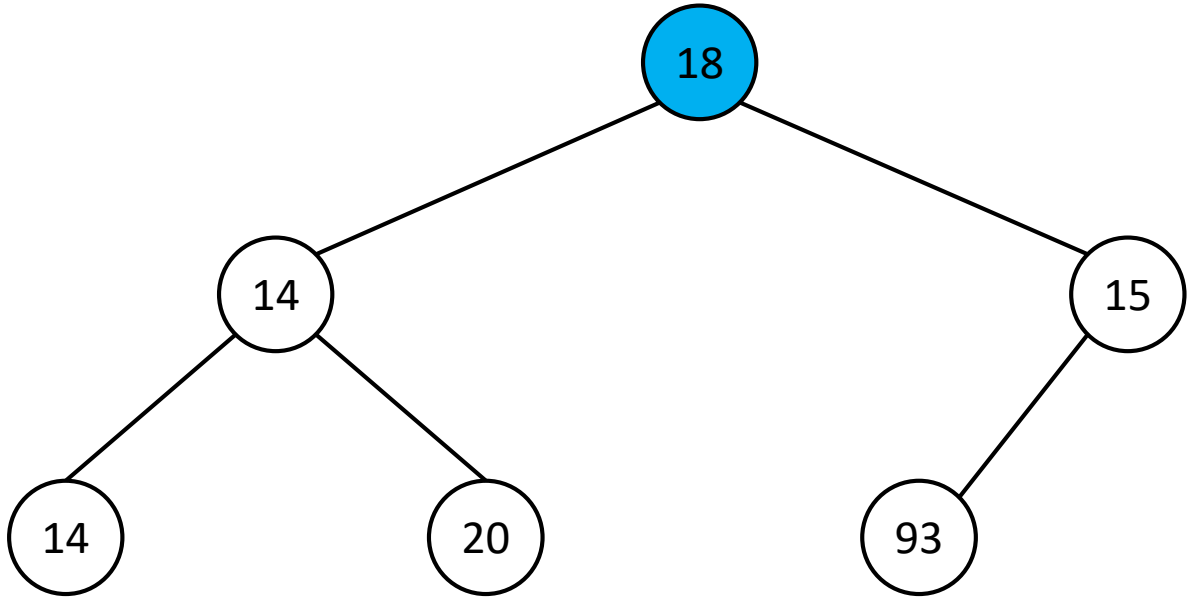
Les tas (Suppression racine)



Les tas (Suppression racine)

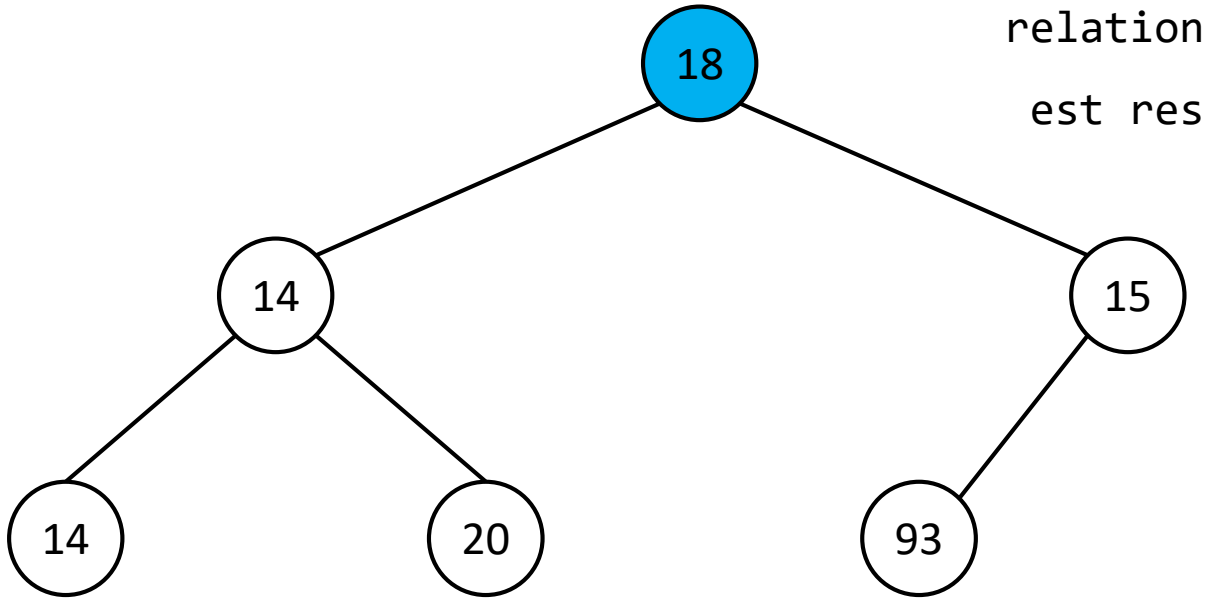


Les tas (Suppression racine)

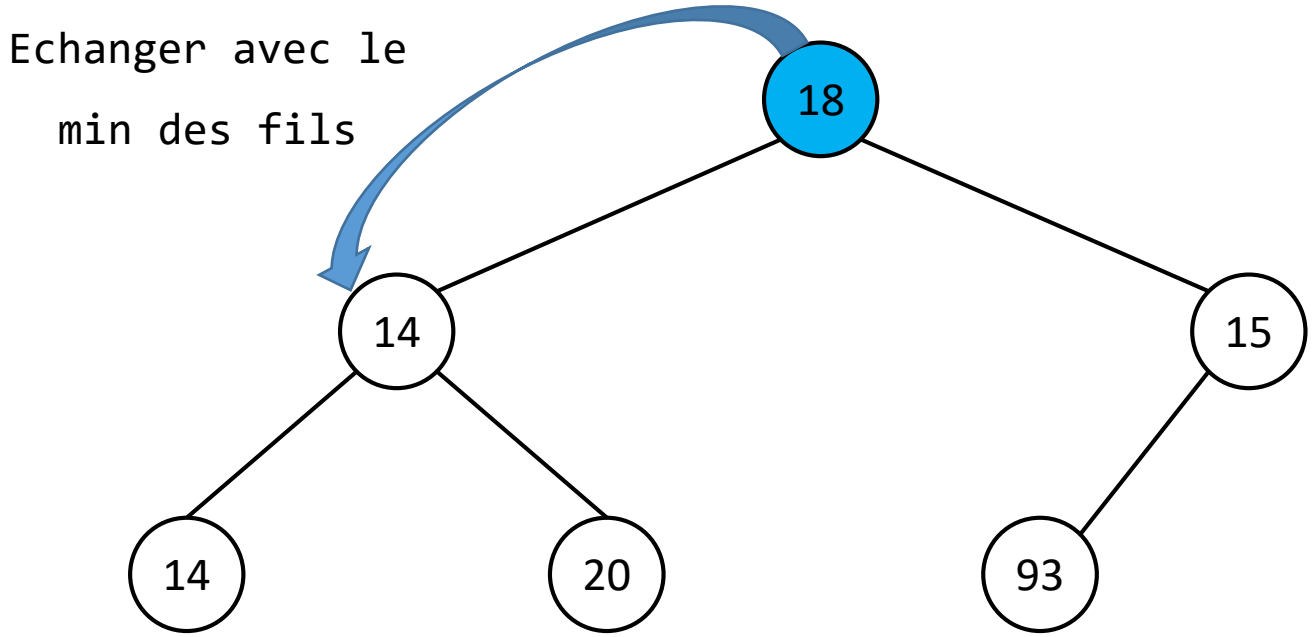


Les tas (Suppression racine)

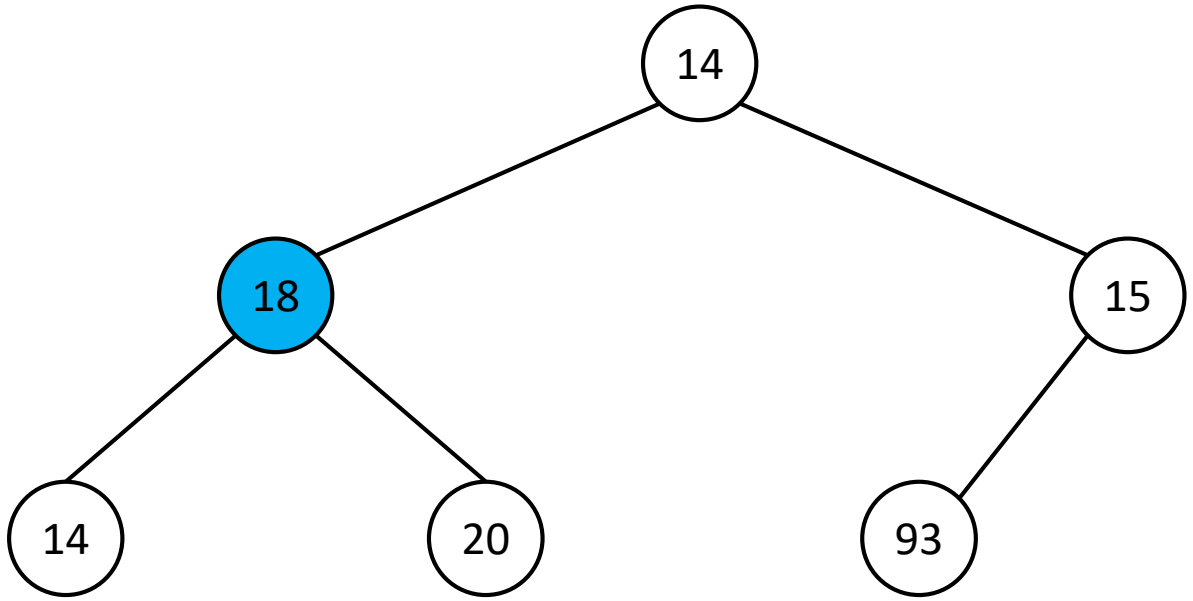
Est-ce que la relation d'ordre est respectée?



Les tas (Suppression racine)

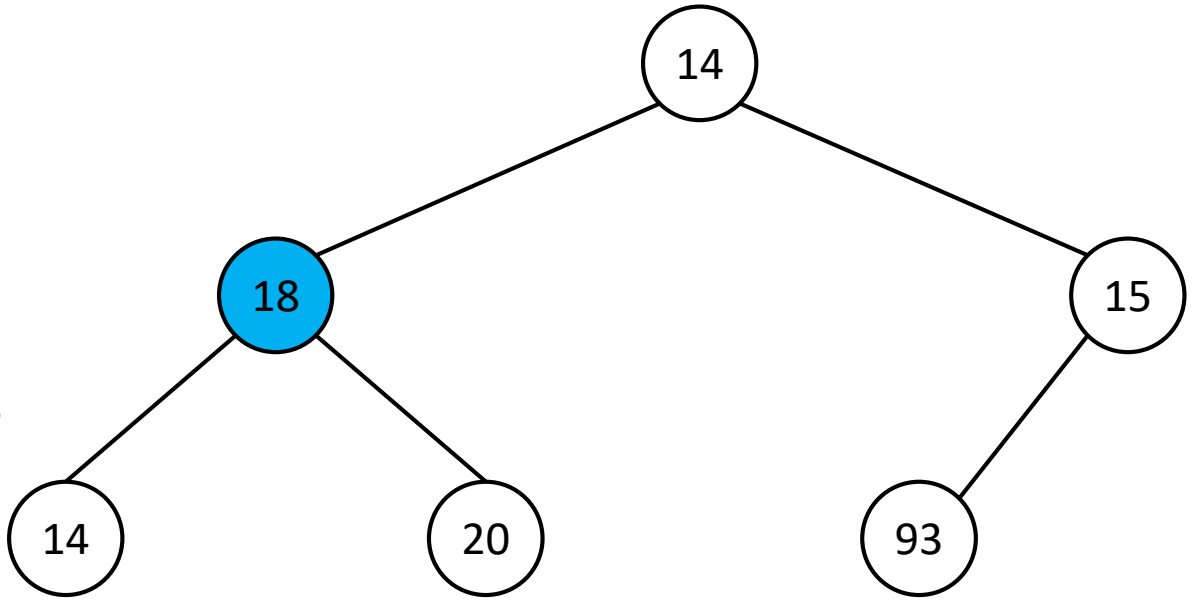


Les tas (Suppression racine)



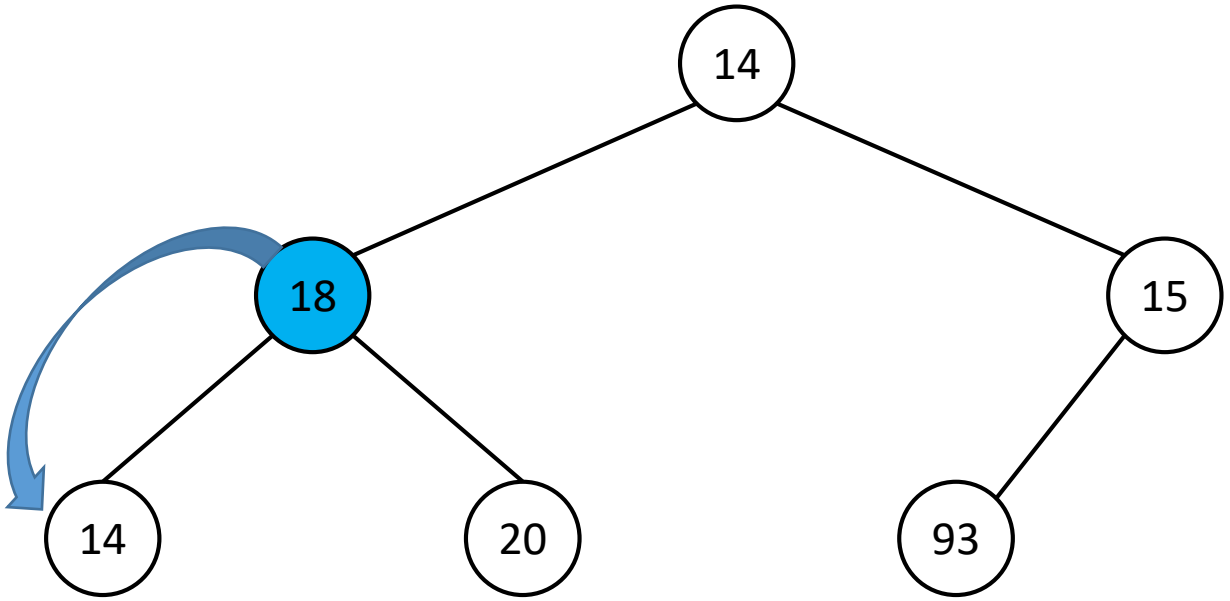
Les tas (Suppression racine)

Est-ce que la relation d'ordre est respectée?

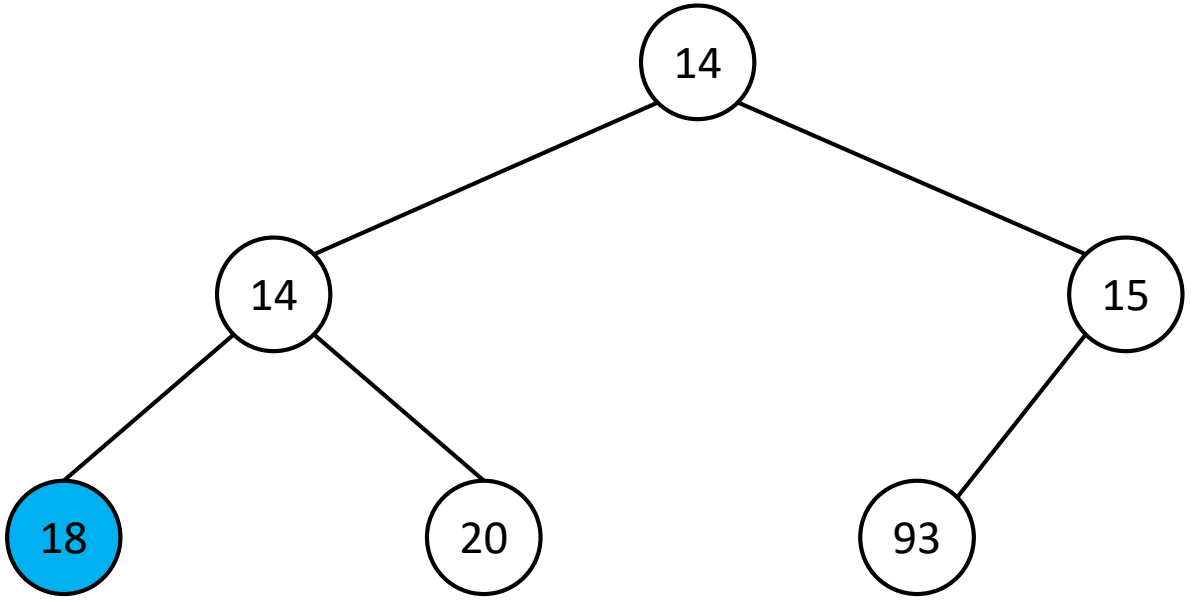


Les tas (Suppression racine)

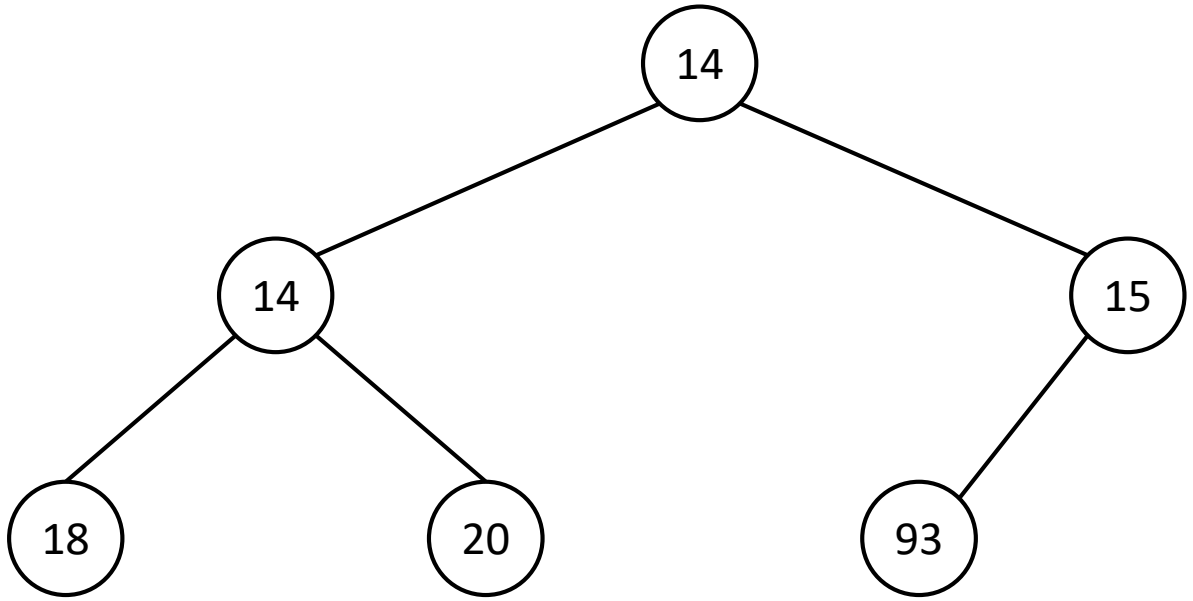
Echanger avec le min des fils



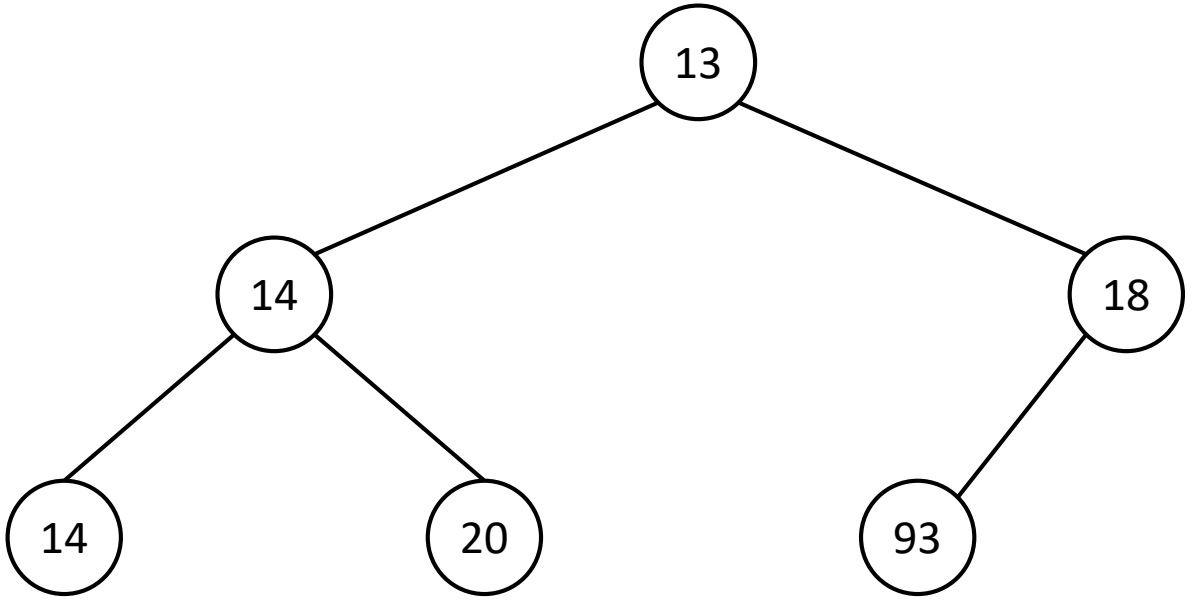
Les tas (Suppression racine)



Les tas (Suppression racine)



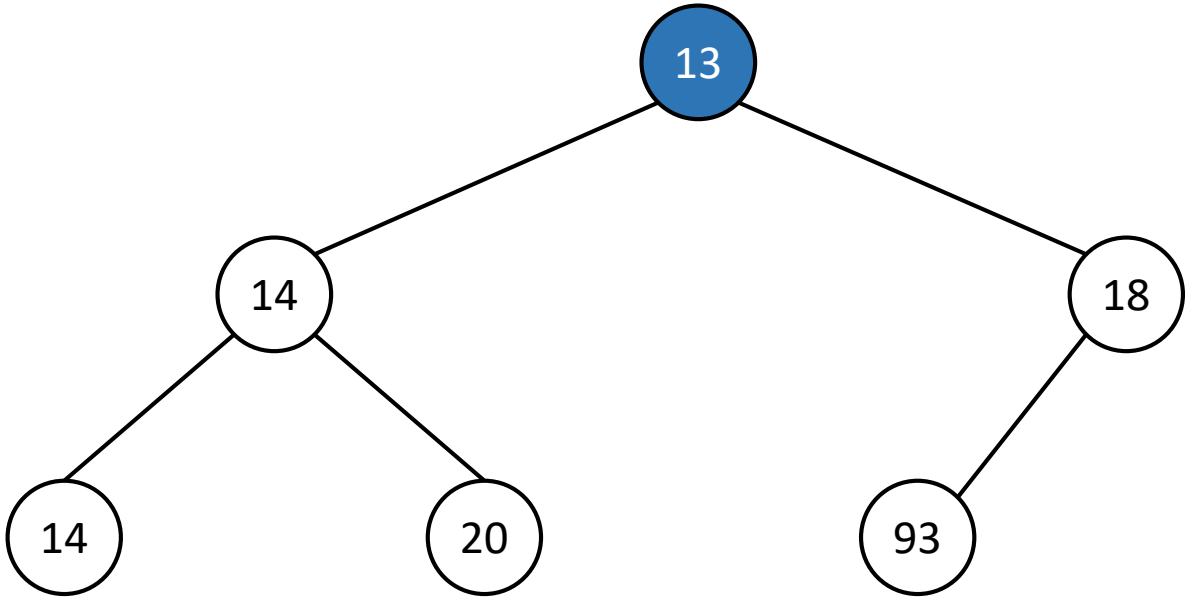
Les tas (Représentation par un tableau)

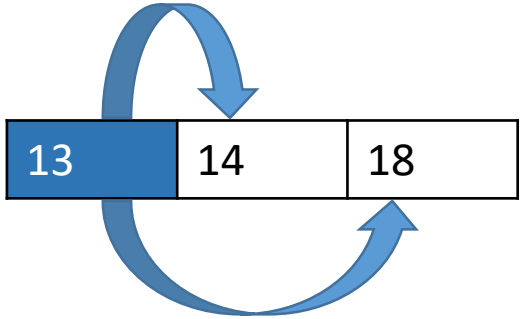


13

Tas Min

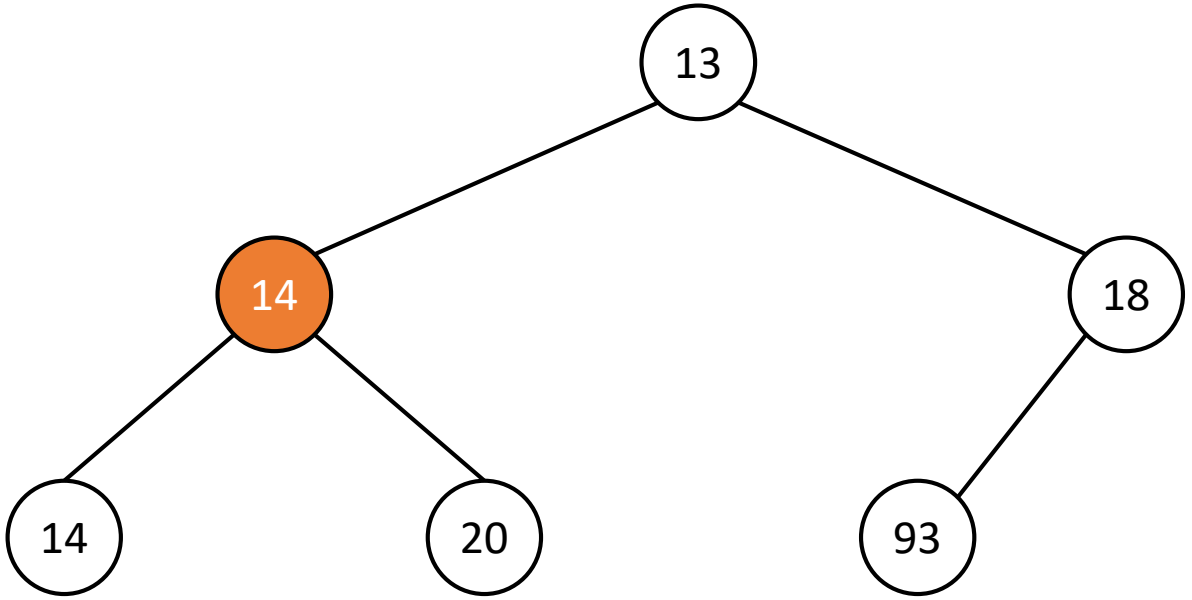
Les tas (Représentation par un tableau)



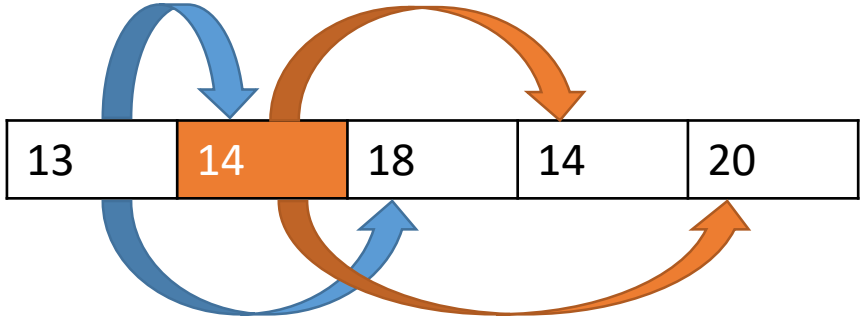


Tas Min

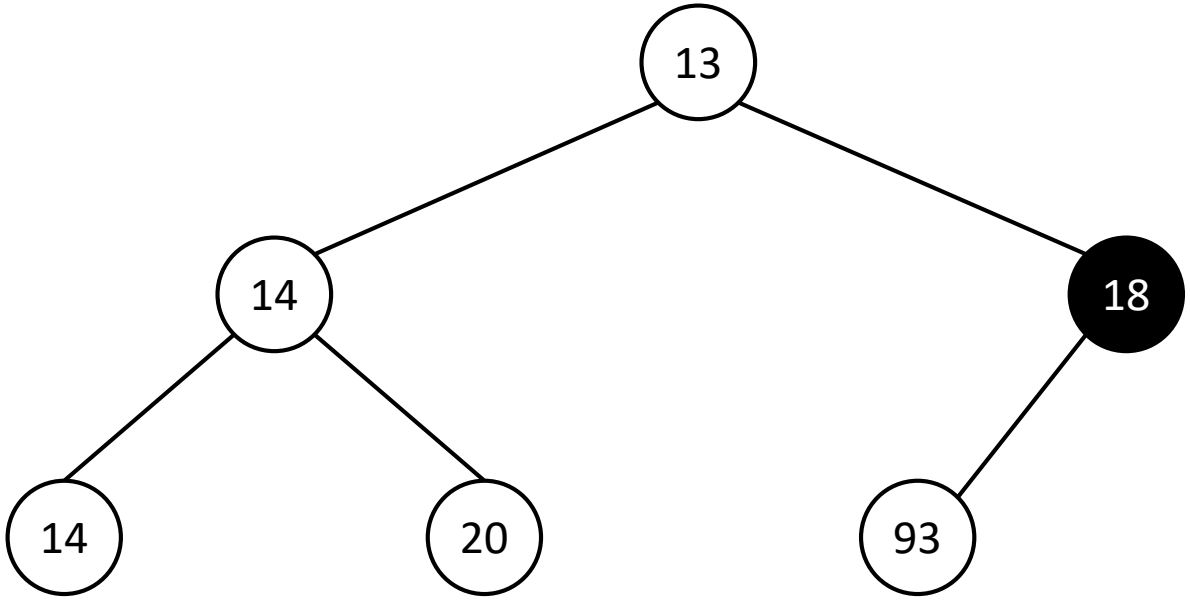
Les tas (Représentation par un tableau)



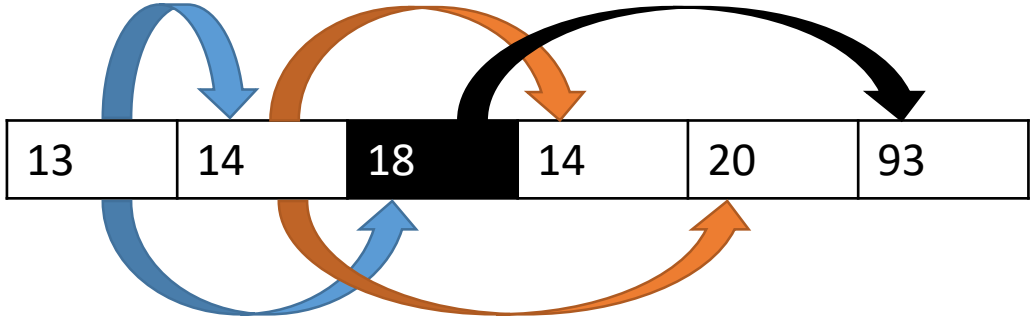
Tas Min



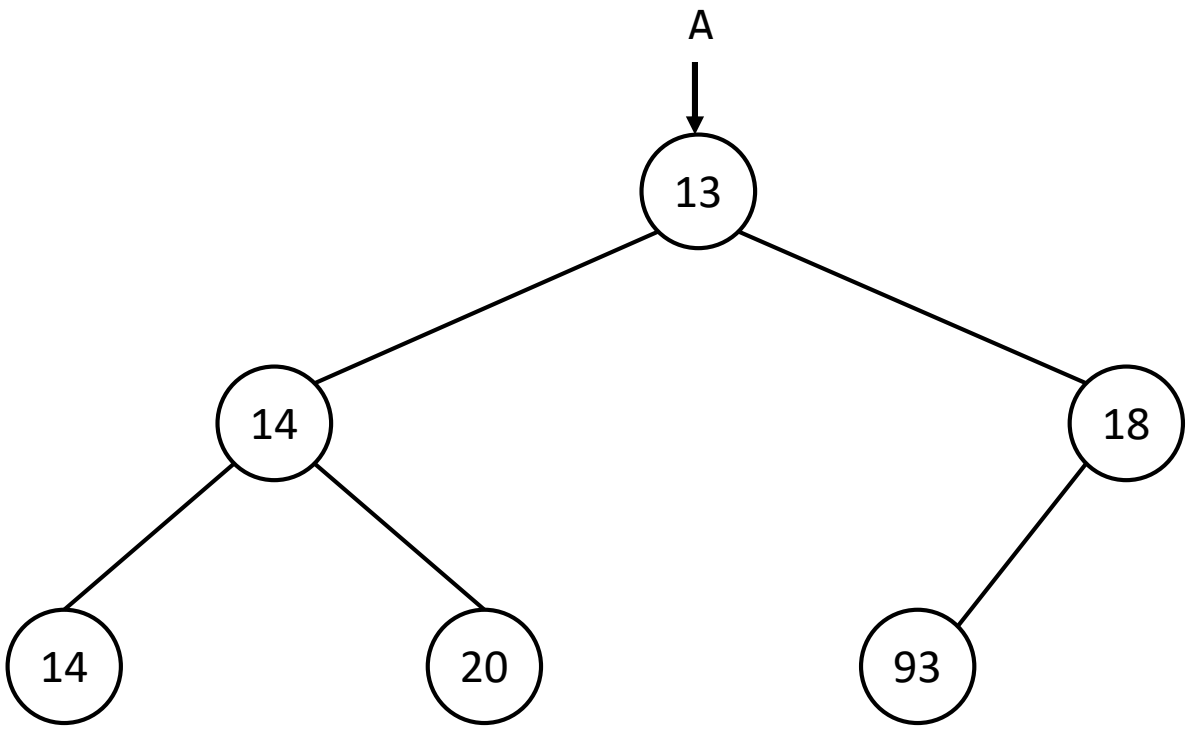
Les tas (Représentation par un tableau)



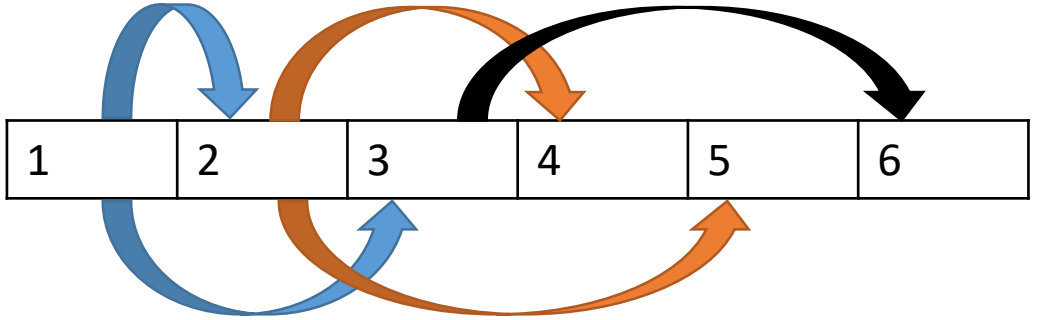
Tas Min



Les tas (Représentation par un tableau)



Tas Min



Racine(A): Case 1
Pere(i): Case ⌊i/2⌋
Gauche(i): Case 2i
Droit(i): Case 2i + 1

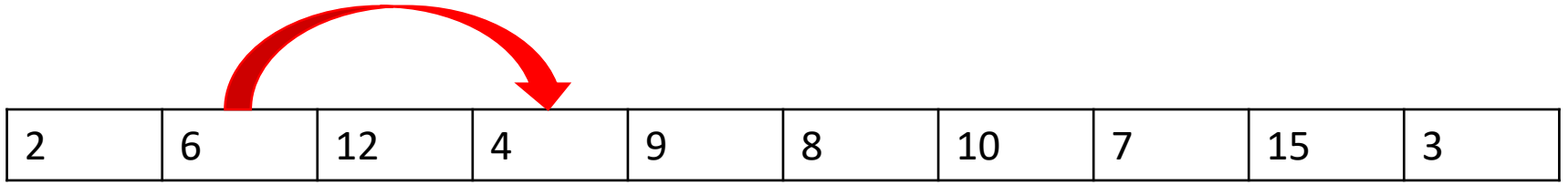
Question 1

- **Liste L1:** L1[i] ≥ L1[⌊i/2⌋] ∀i>1 donc cette liste respecte les propriétés d'un tas.

1	3	1	6	9	7	12	8	7	9	10	10

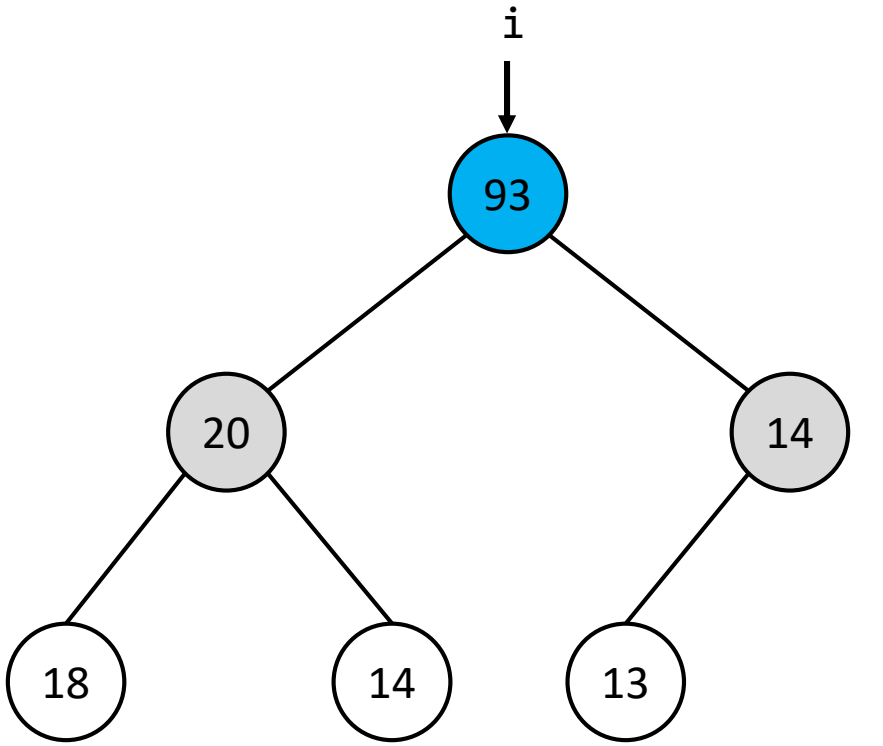
Question 1

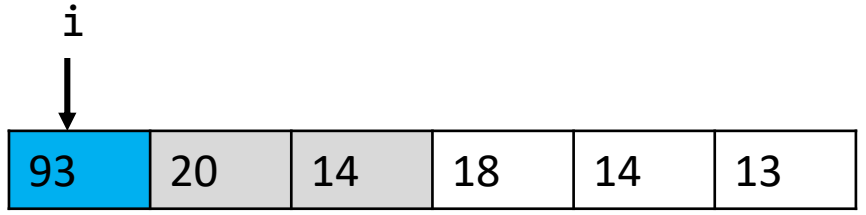
- **Liste L2:** L2[4] < L2[2] donc cette liste ne respecte pas les propriétés d'un tas.



Question 2

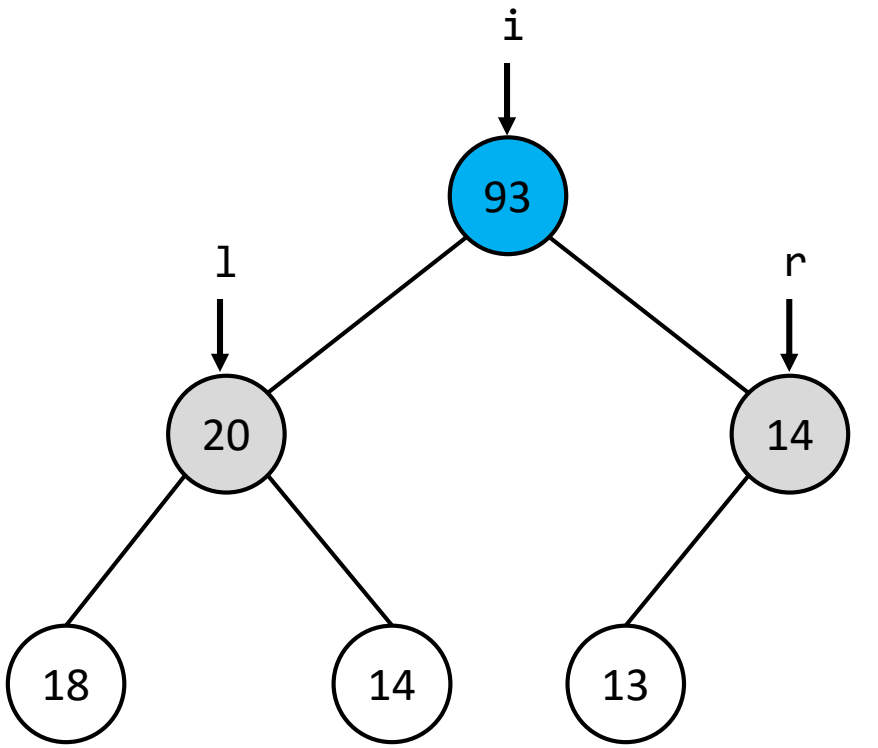
```
Entasser (A: Tableau, i: noeud)
```

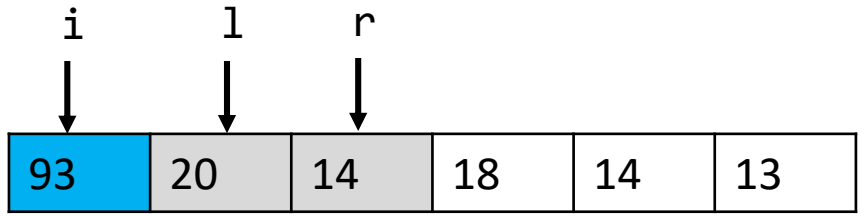




Question 2

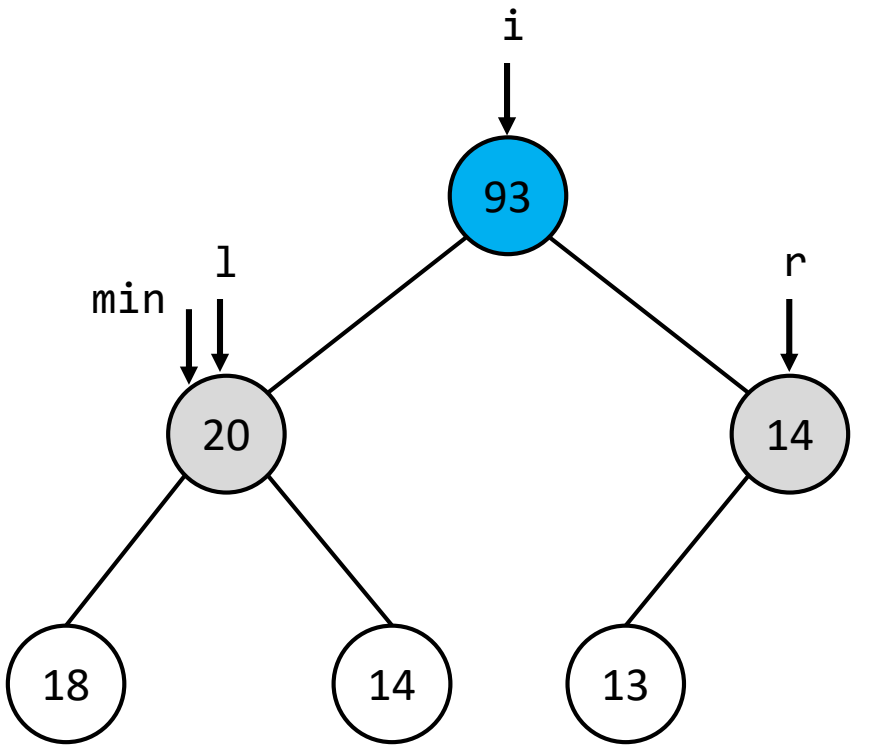
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]
```

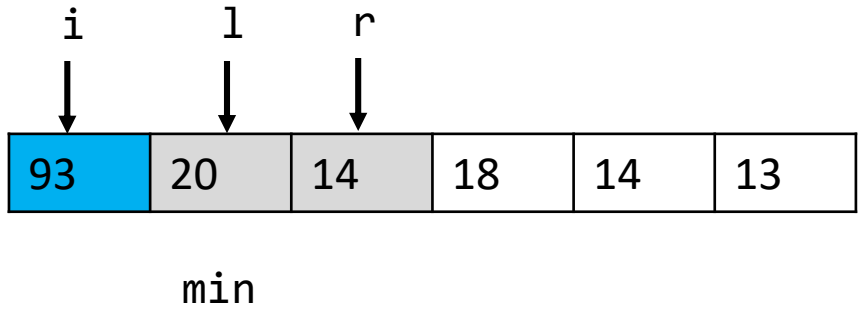




Question 2

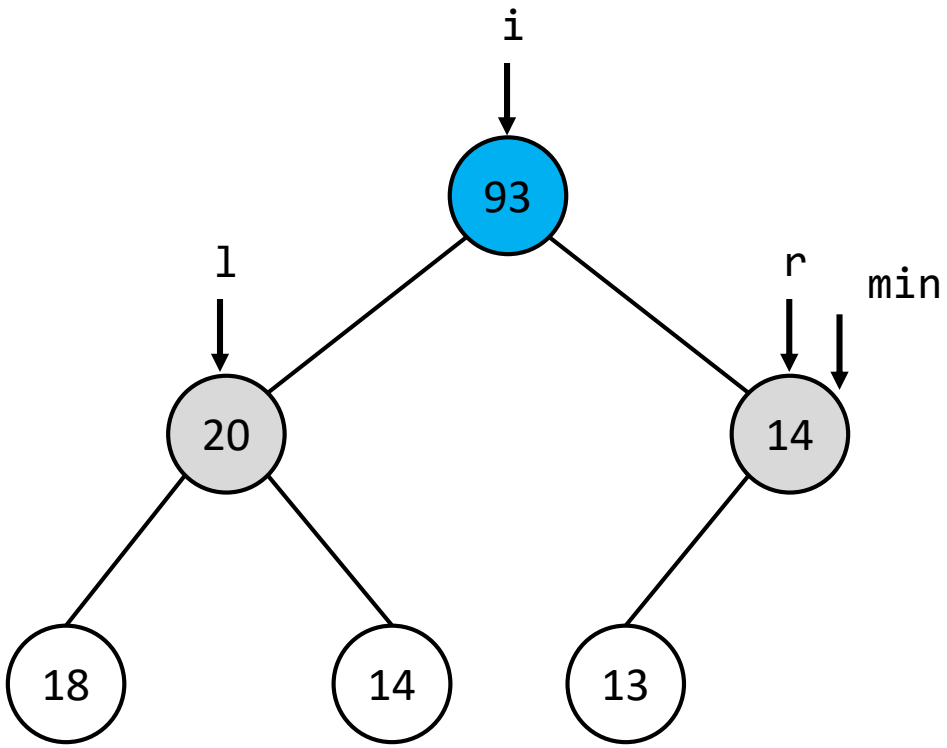
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i
```

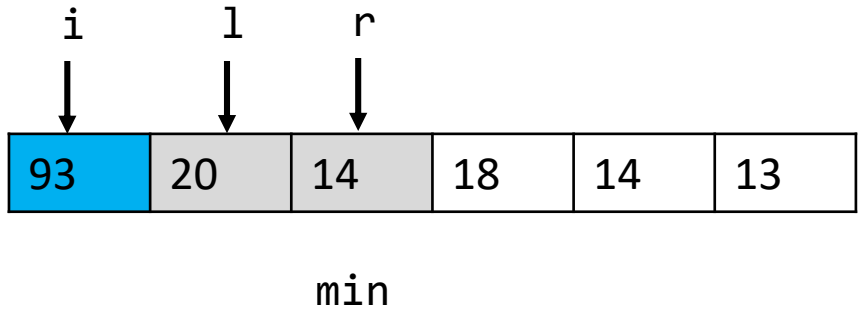




Question 2

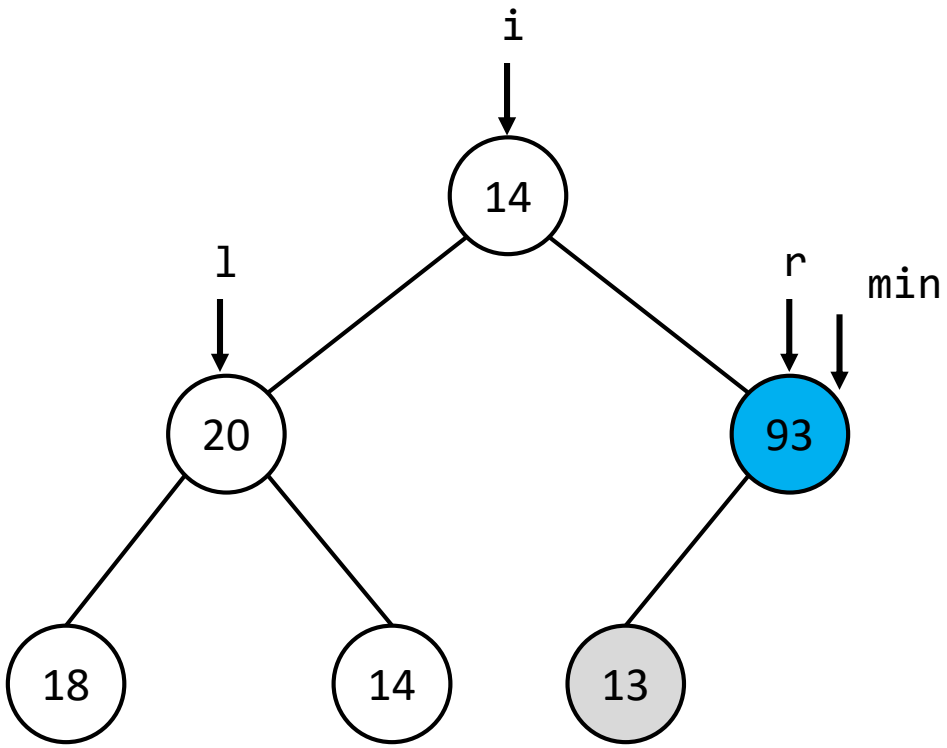
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r
```

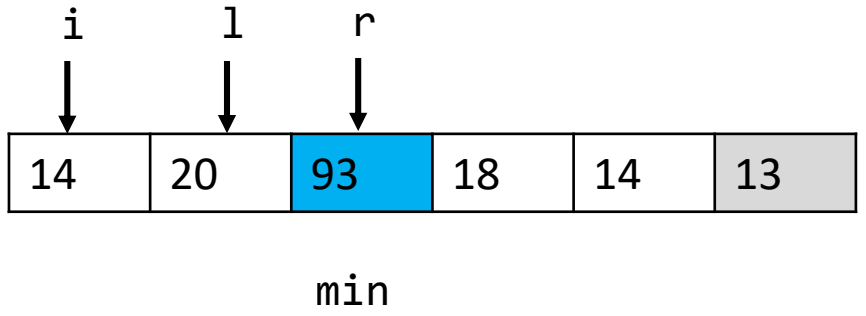




Question 2

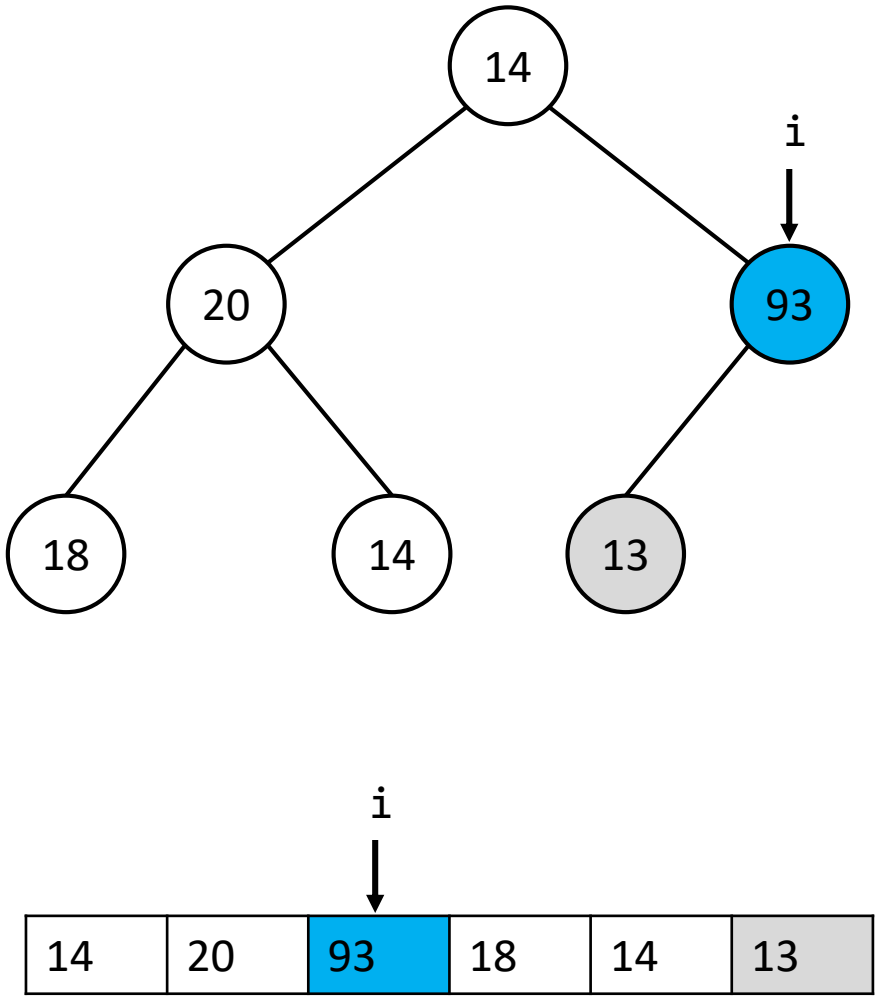
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
```





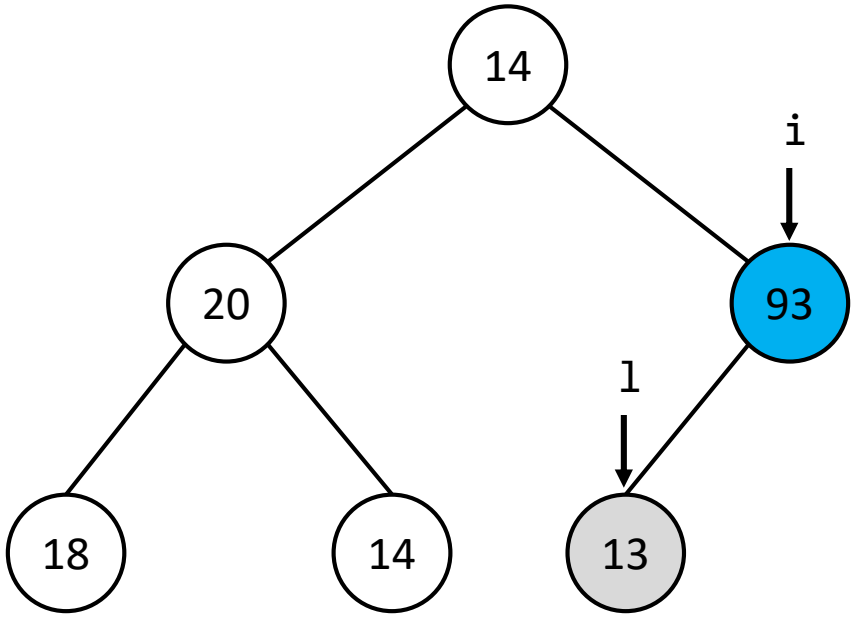
Question 2

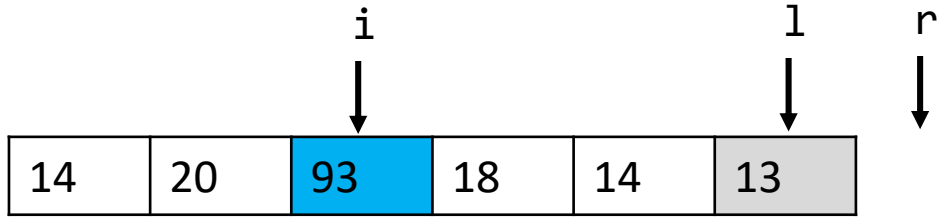
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
        Entasser (A, min)
```



Question 2

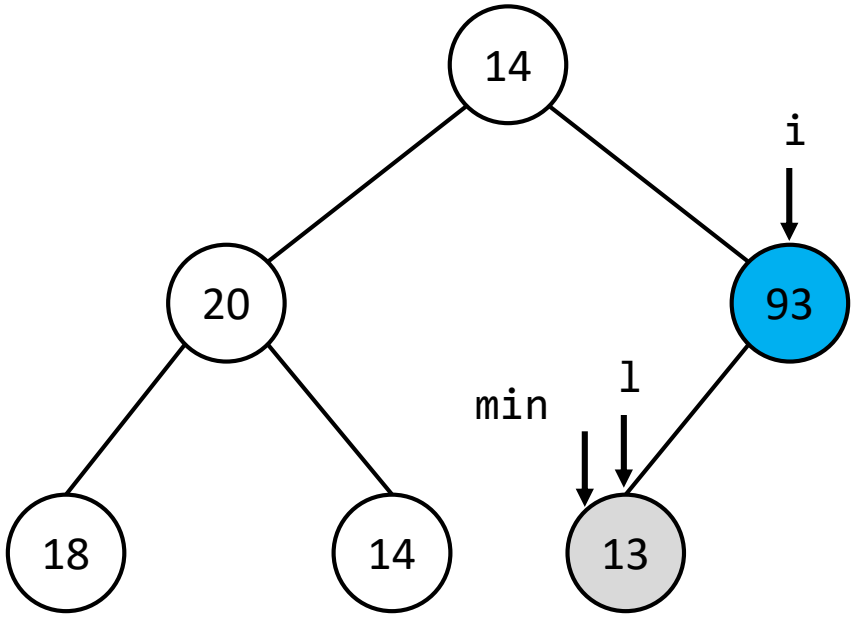
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
        Entasser (A, min)
```

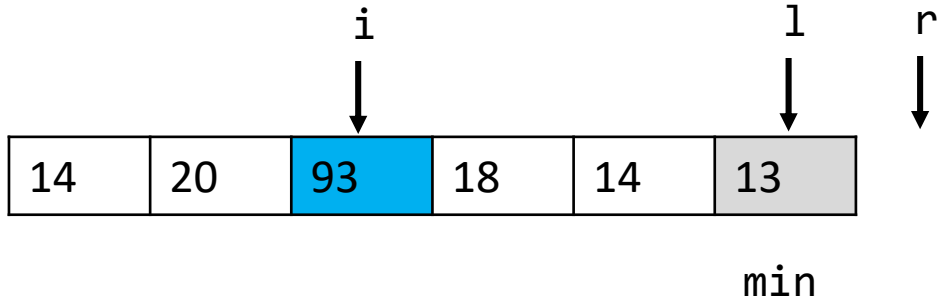




Question 2

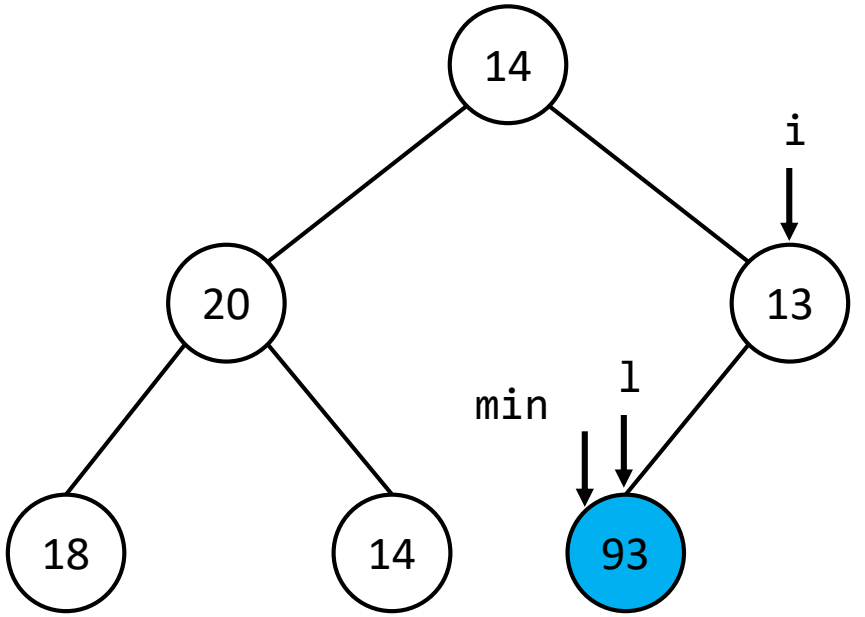
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
        Entasser (A, min)
```

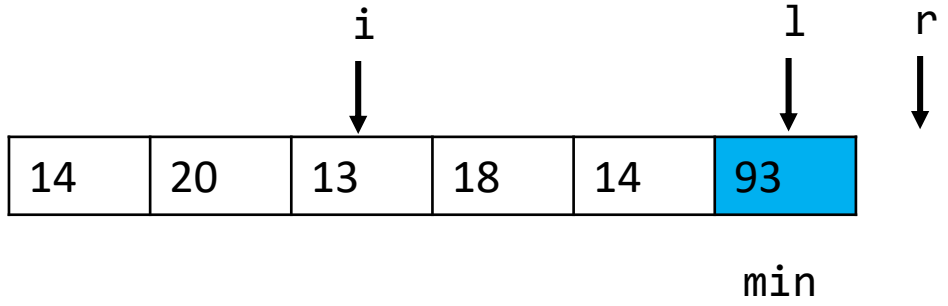




Question 2

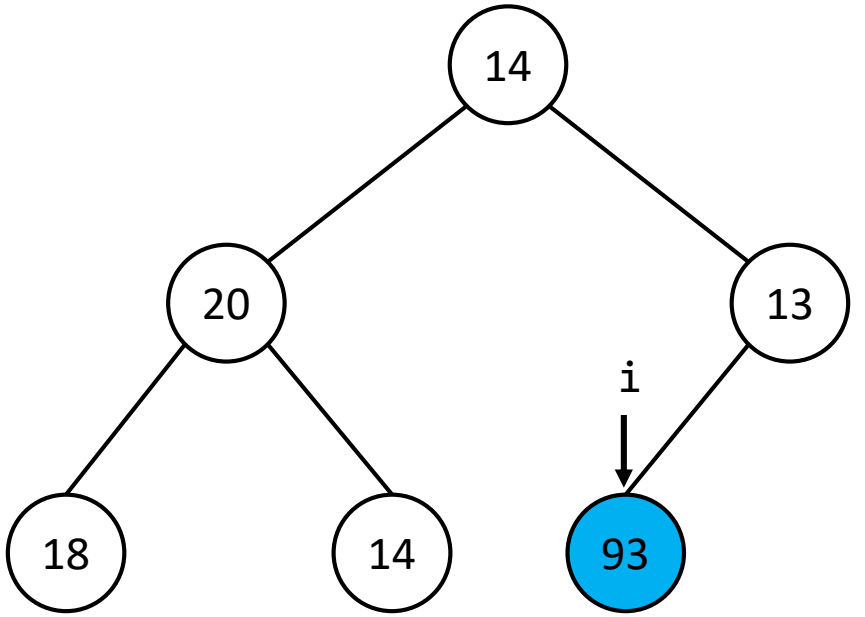
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
        Entasser (A, min)
```

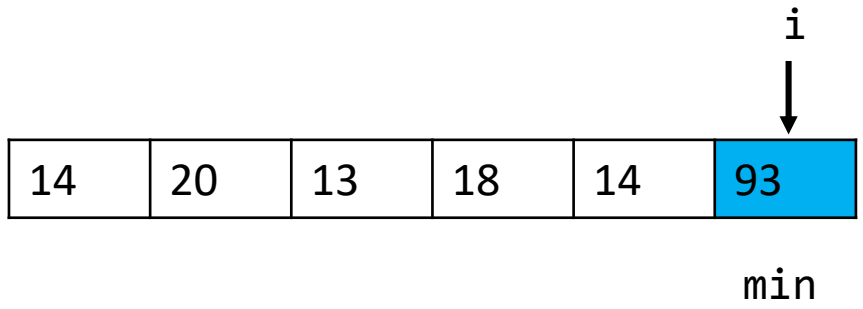




Question 2

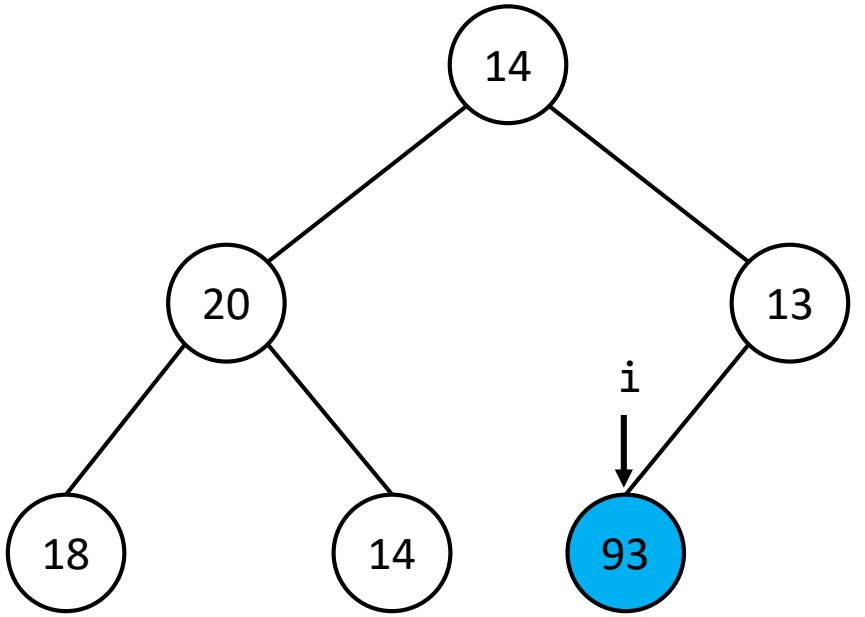
```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
        Entasser (A, min)
```

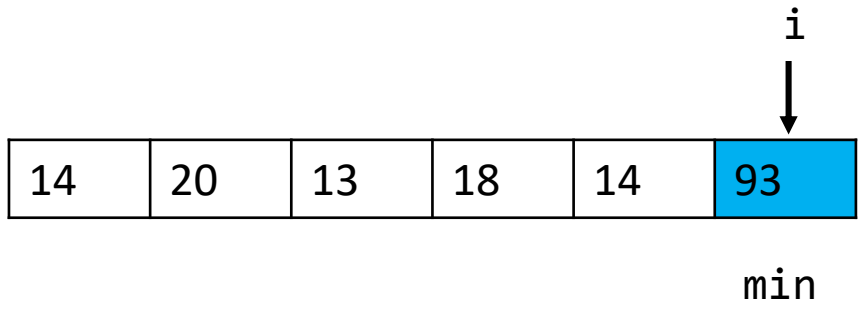




Question 2

```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
        Entasser (A, min)
```





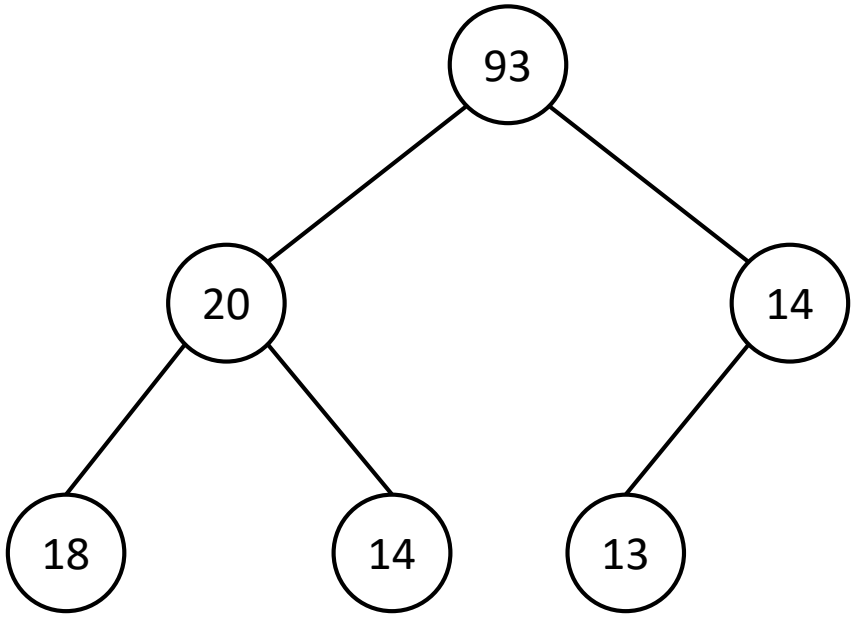
Question 2

```
Entasser (A: Tableau, i: noeud)
    l ← Gauche[i]
    r ← Droit[i]

    // Recuperer le min entre i son fils gauche
    Si l <= Taille[A] et A[l] < A[i]
        min ← l
    Sinon
        min ← i

    // Recuperer le min entre i et ses deux fils
    Si r <= Taille[A] et A[r] < A[min]
        min ← r

    // Si i a un fils inférieur à lui
    Si min != i
        Echanger (A, i, min)
        Entasser (A, min)
```

Complexité: O(h)

Question 2

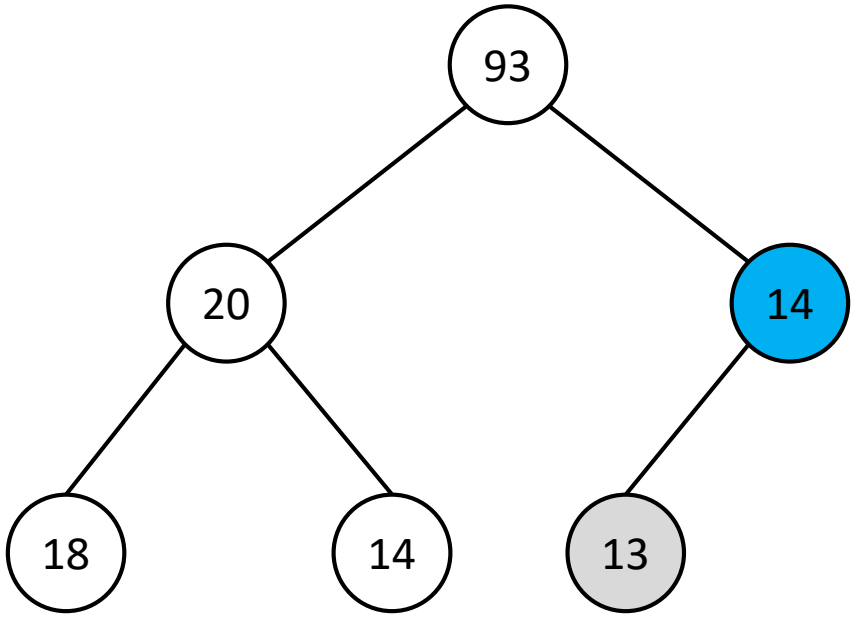
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

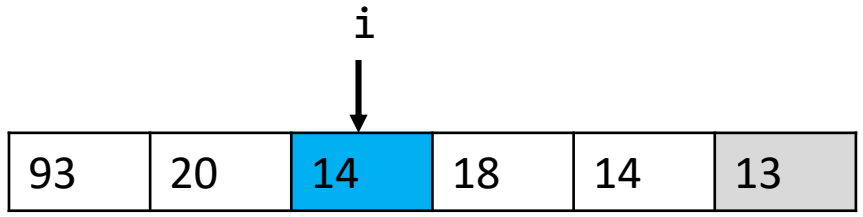


93	20	14	18	14	13

Question 2

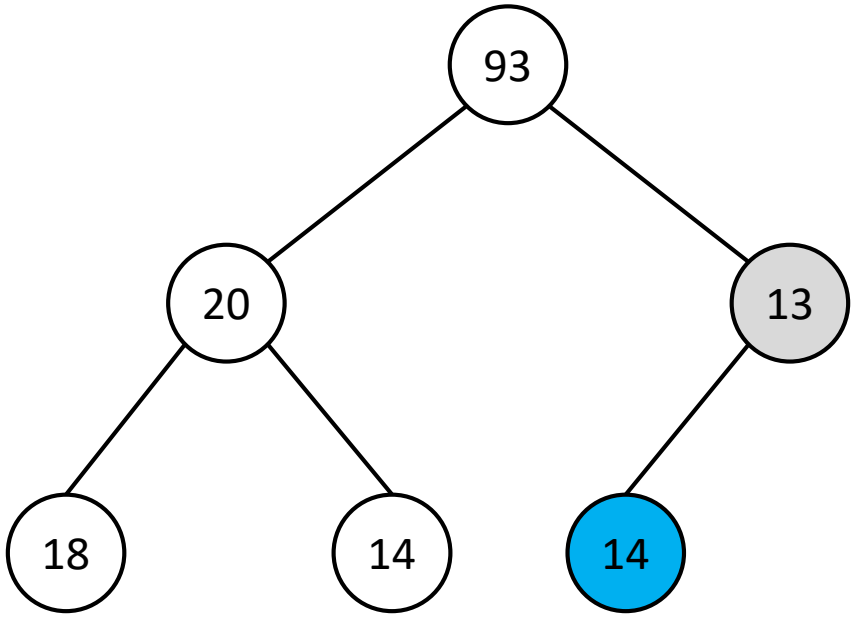
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

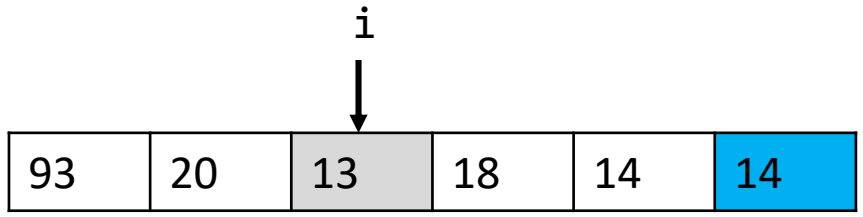




Question 2

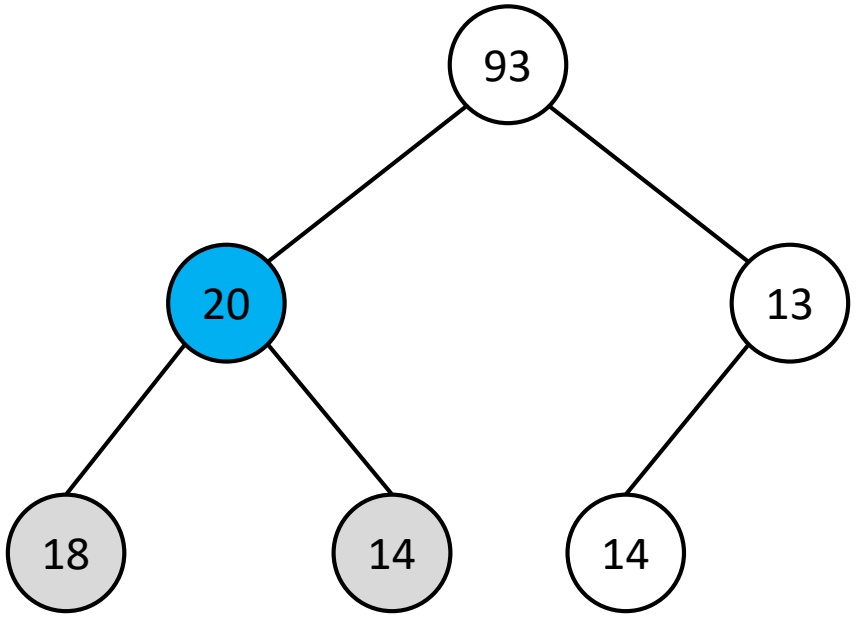
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

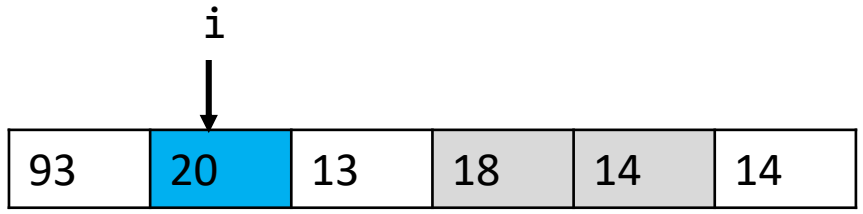




Question 2

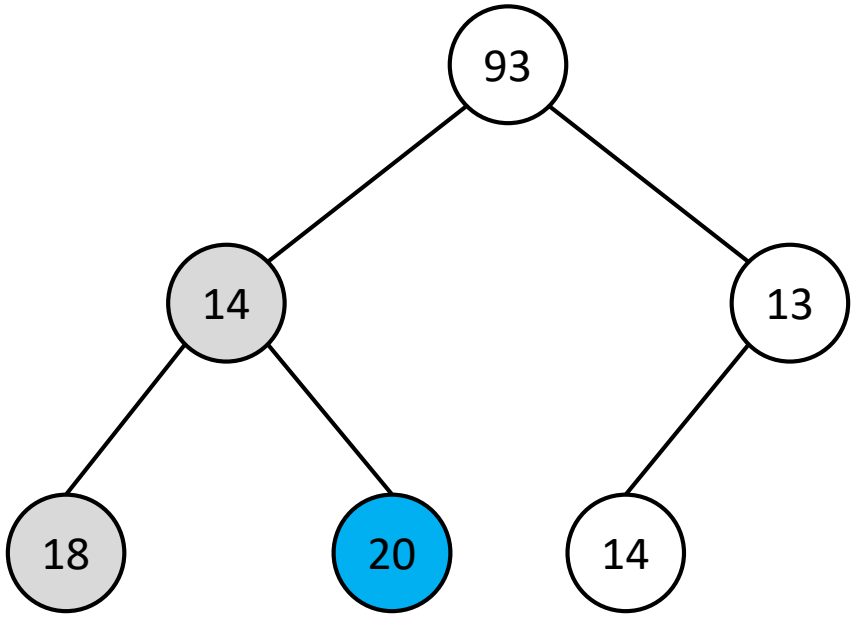
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

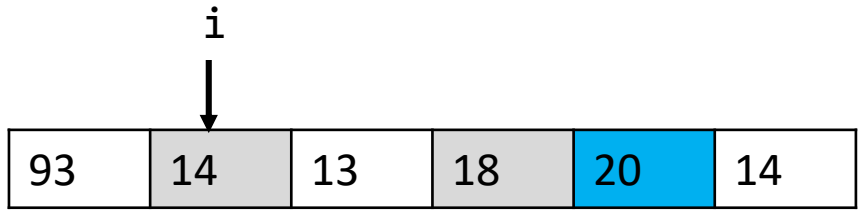




Question 2

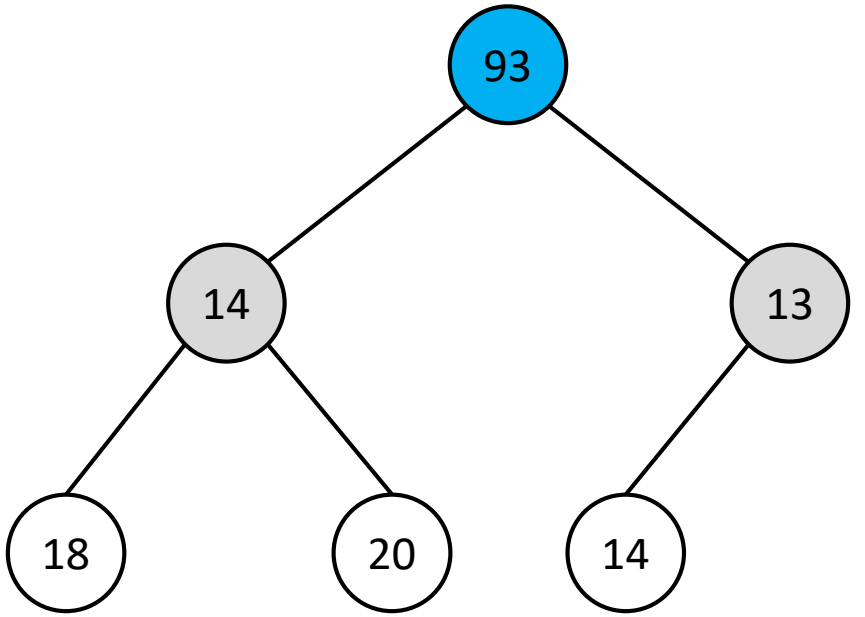
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

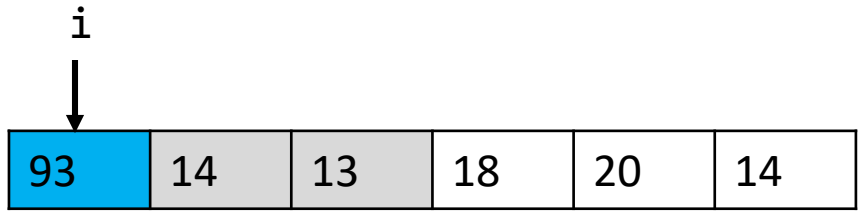




Question 2

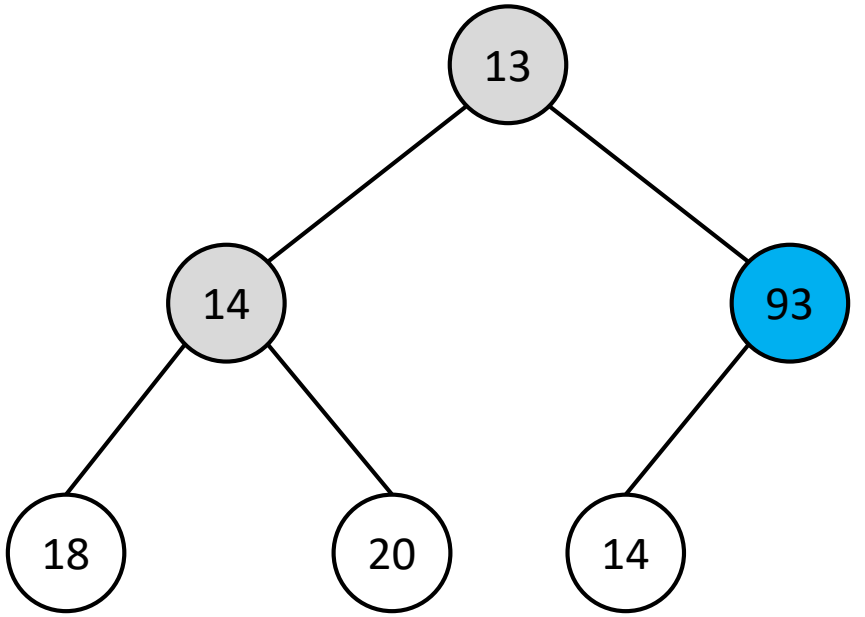
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

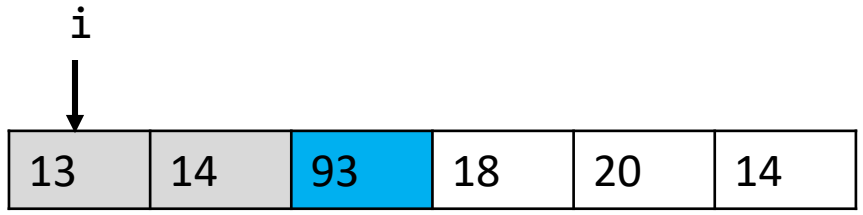




Question 2

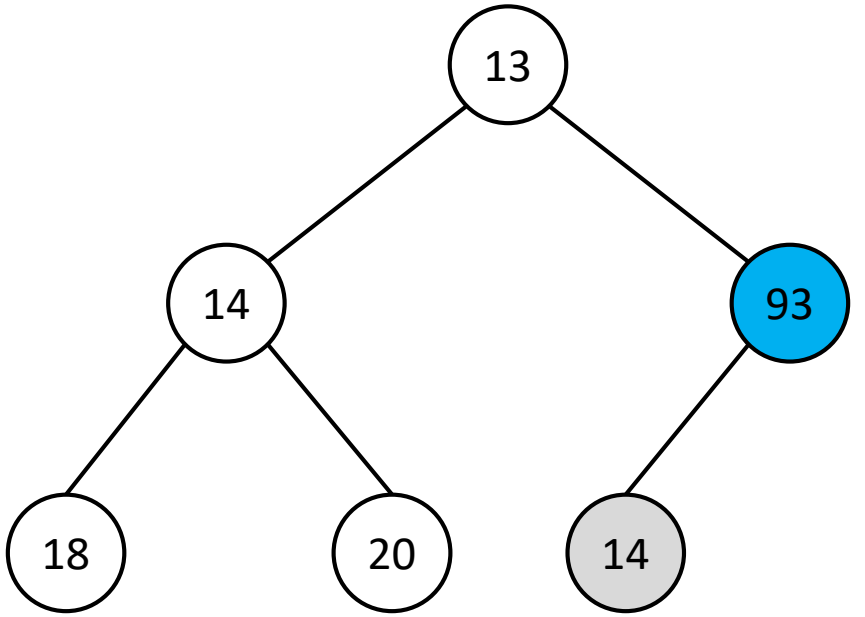
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

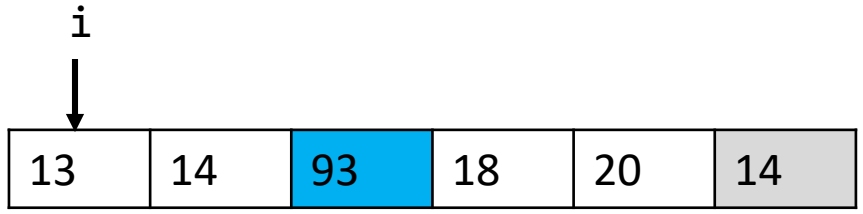




Question 2

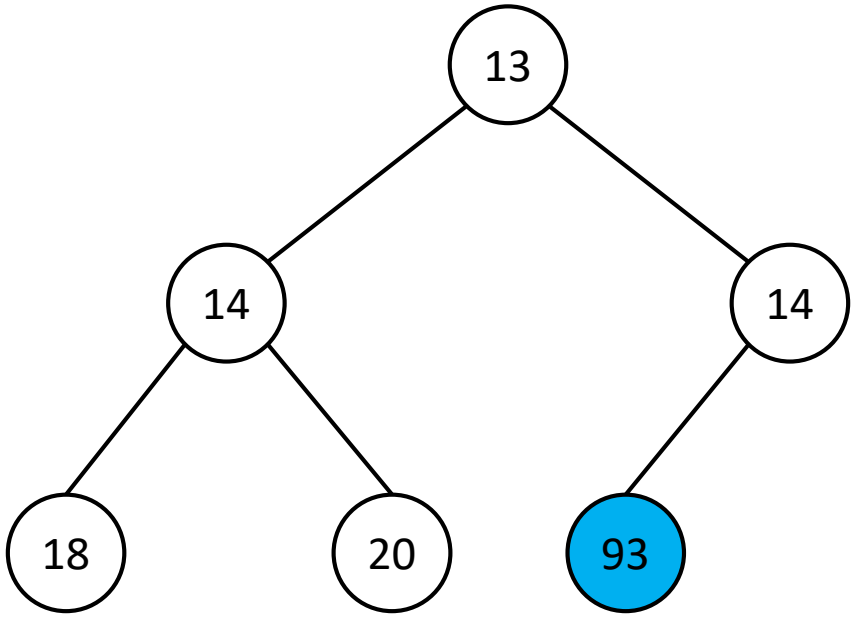
```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

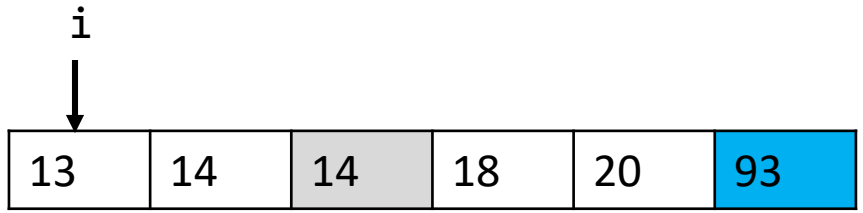




Question 2

```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```





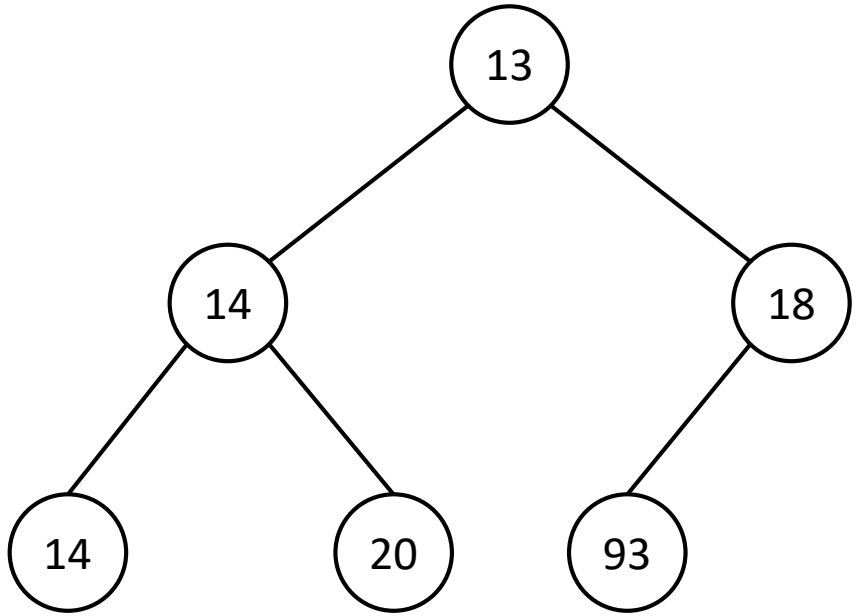
Question 2

```
construire_tas(A: Tableau)
    Pour i allant de Taille[A]/2 à 1 par pas de -1
        Entasser (A, i)
```

Complexité: O(n)

Question 3

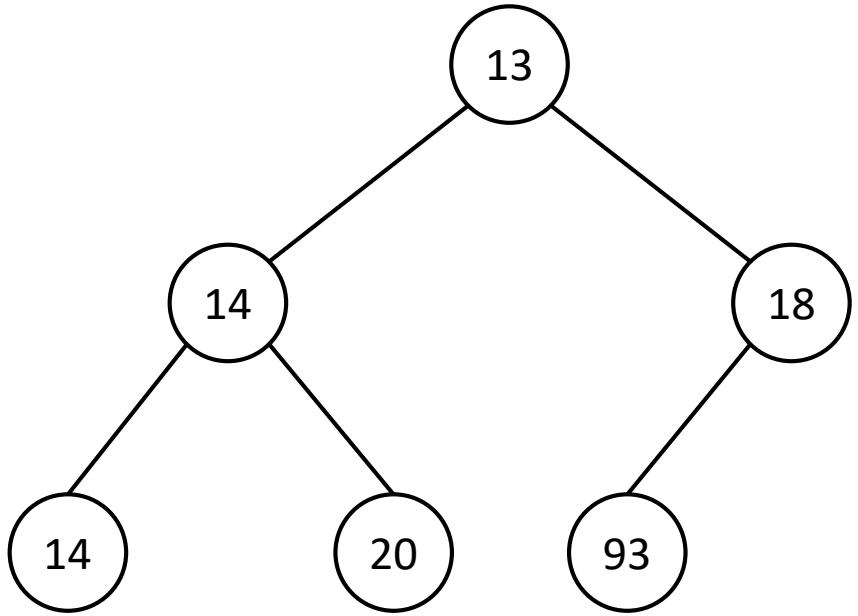
```
inserer(A: Tas , cle: valeur)
```



13	14	18	14	20	93

Question 3

```
inserer(A: Tas , cle: valeur)
```

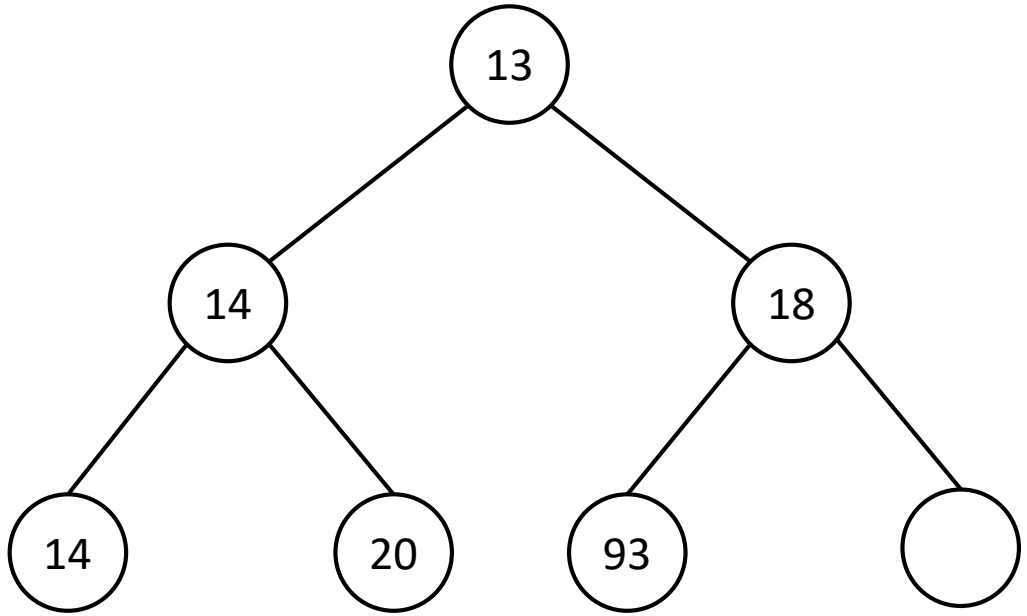


Insérer 15

13	14	18	14	20	93

Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
```

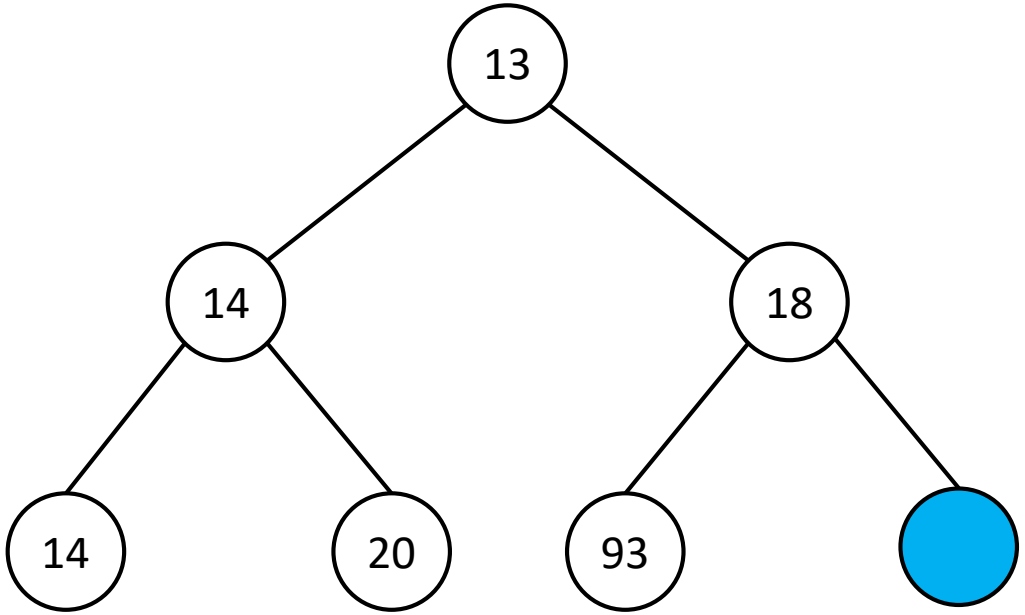


Insérer 15

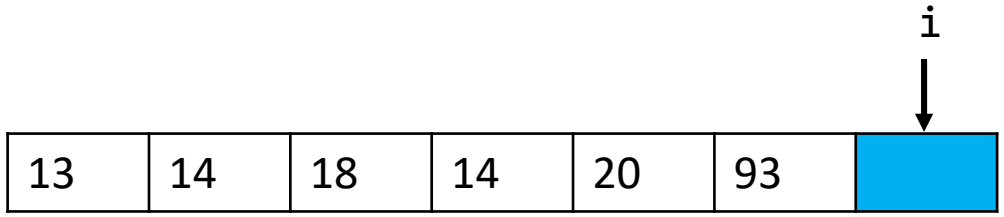
13	14	18	14	20	93	

Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
    i ← Taille[A]
```

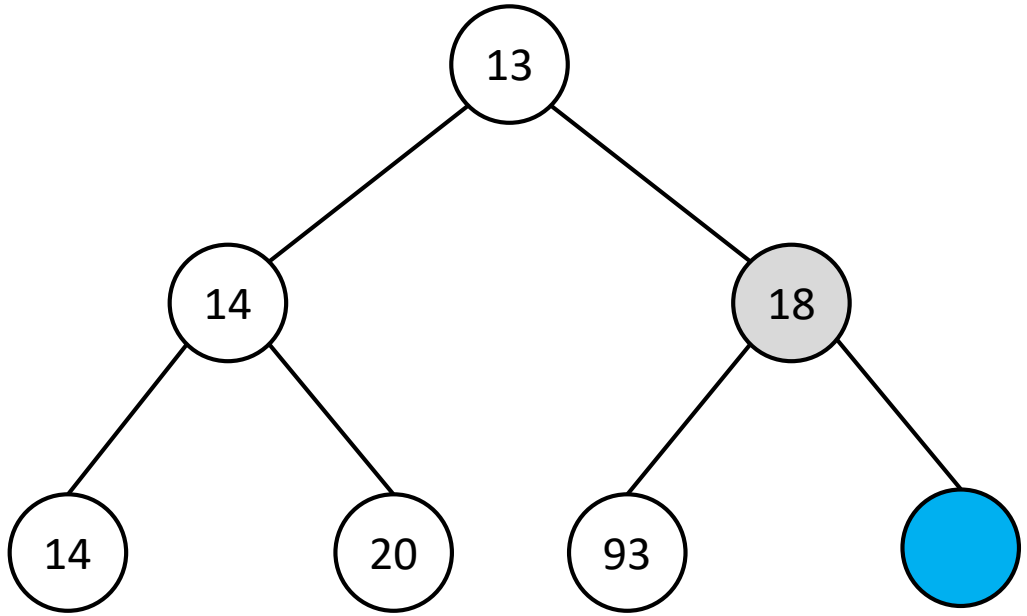


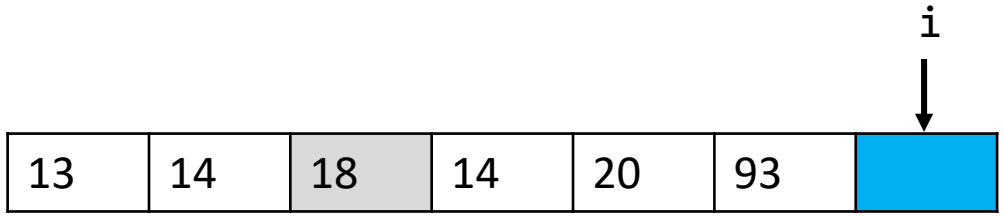
Insérer 15



Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
    i ← Taille[A]
    TantQue i > 1 et A[Pere[i]]> cle
```

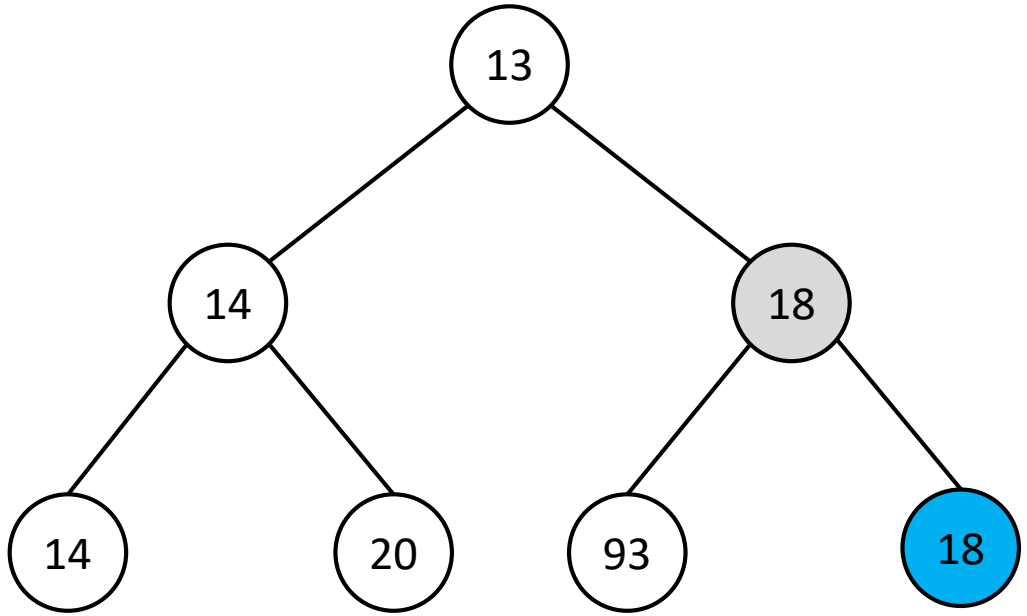




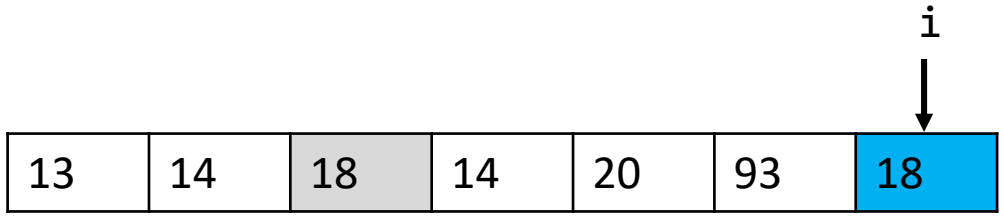
Insérer 15

Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
    i ← Taille[A]
    TantQue i > 1 et A[Pere[i]]> cle
        A[i] ← A[Pere[i]]
```

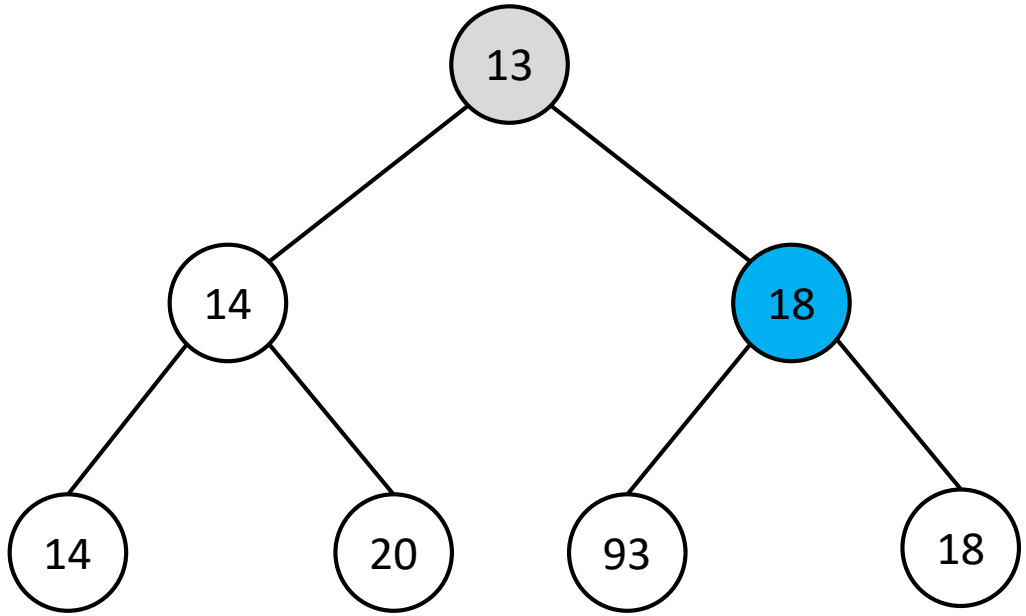


Insérer 15

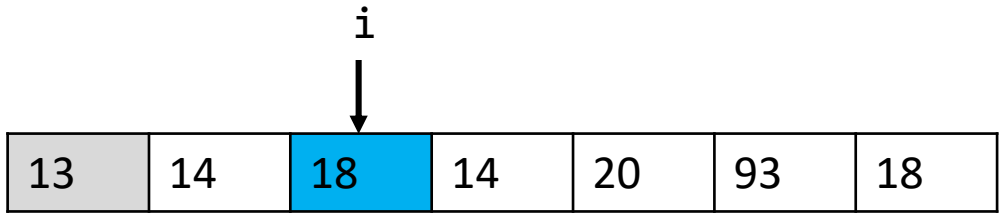


Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
    i ← Taille[A]
    TantQue i > 1 et A[Pere[i]]> cle
        A[i] ← A[Pere[i]]
        i ← Pere[i]
```

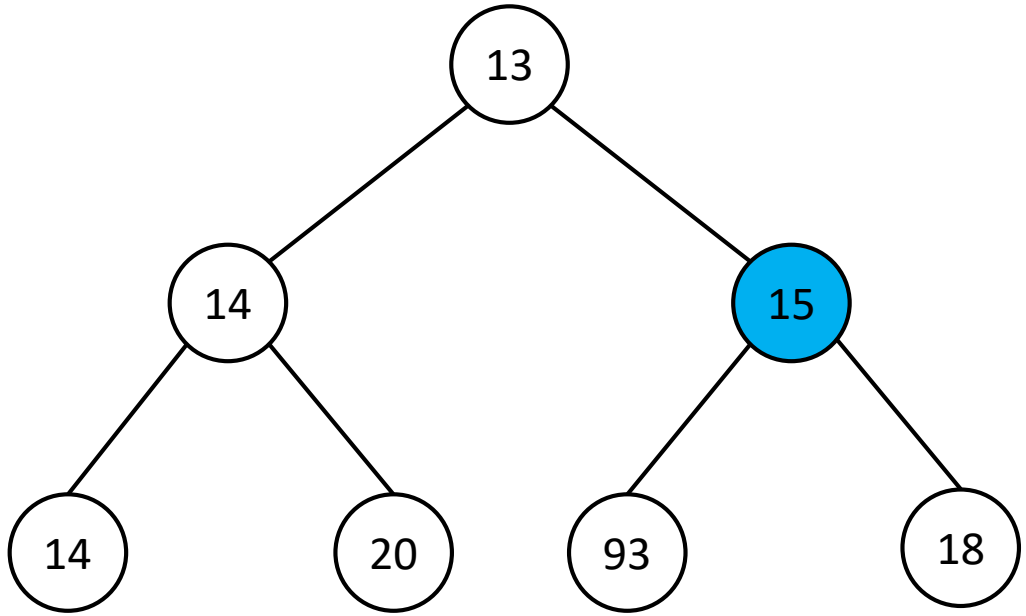


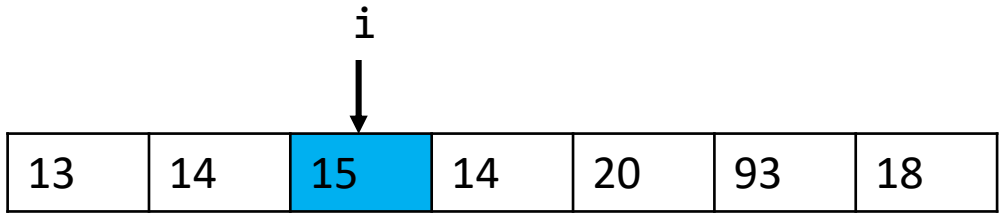
Insérer 15



Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
    i ← Taille[A]
    TantQue i > 1 et A[Pere[i]]> cle
        A[i] ← A[Pere[i]]
        i ← Pere[i]
    A[i] ← cle
```

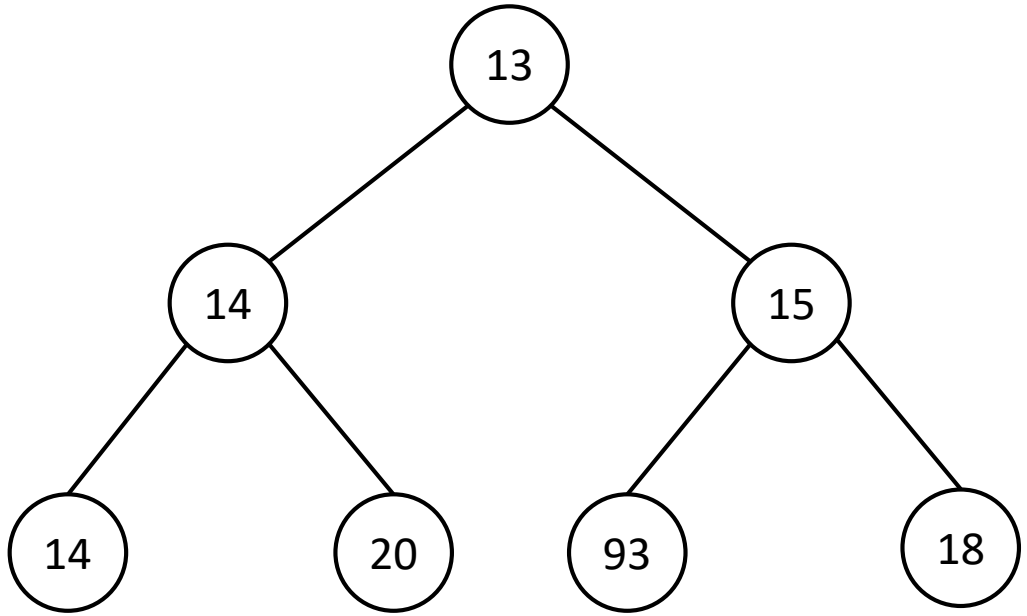




Insérer 15

Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
    i ← Taille[A]
    TantQue i > 1 et A[Pere[i]]> cle
        A[i] ← A[Pere[i]]
        i ← Pere[i]
    A[i] ← cle
```



Insérer 15

13	14	15	14	20	93	18

Question 3

```
inserer(A: Tas , cle: valeur)
    Taille[A] ← Taille[A]+1
    i ← Taille[A]
    TantQue i > 1 et A[Pere[i]]> cle
        A[i] ← A[Pere[i]]
        i ← Pere[i]
    A[i] ← cle
```

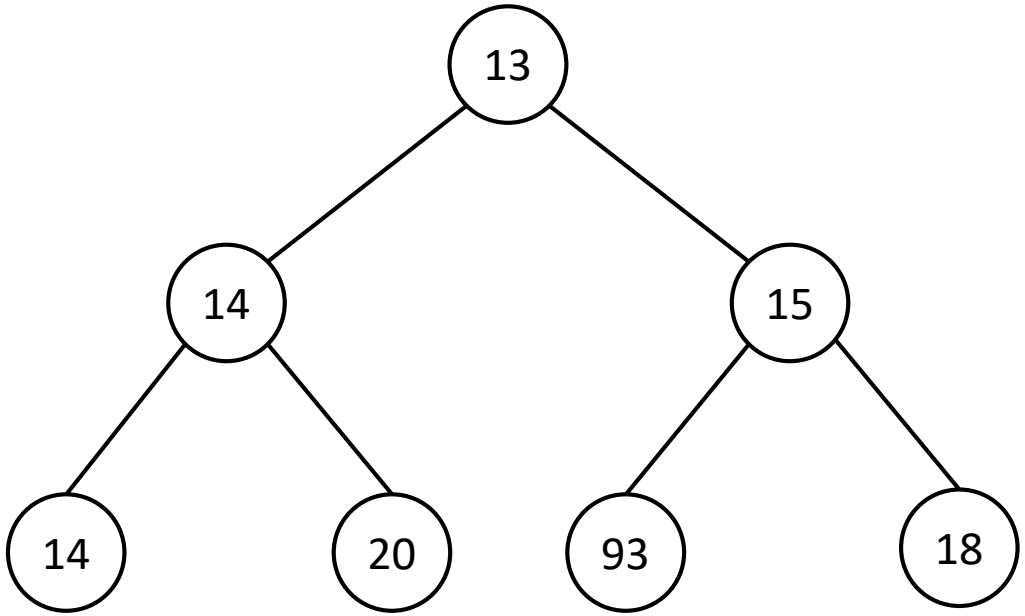
Complexité: O(h)

Question 4

- **Plus petit élément :** à la racine du tas.
- **Plus grand élément :** au niveau d’une feuille.

Question 5

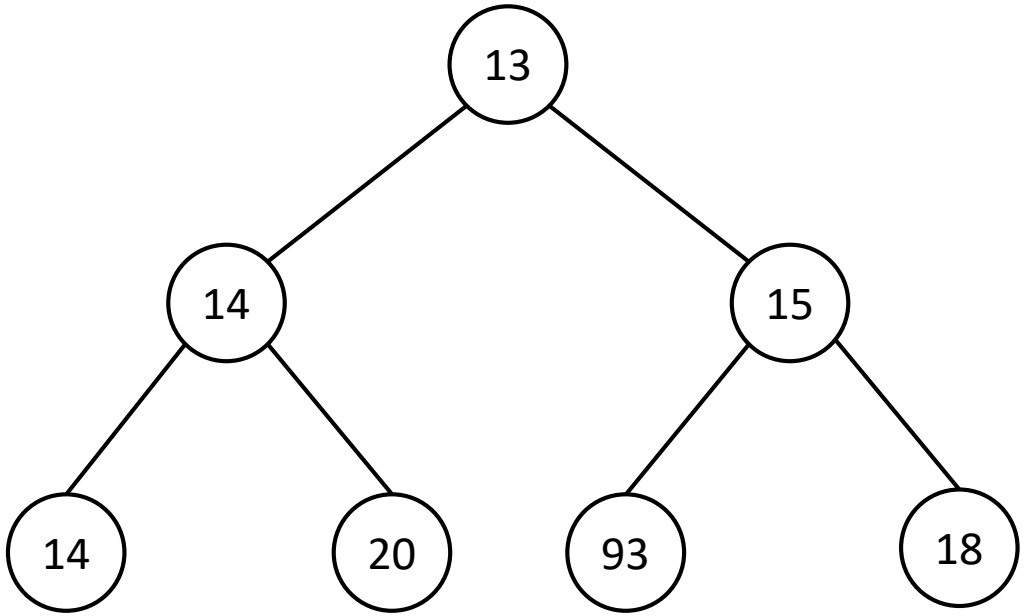
```
supprimer_min(A:Tas)
```



13	14	15	14	20	93	18

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
```

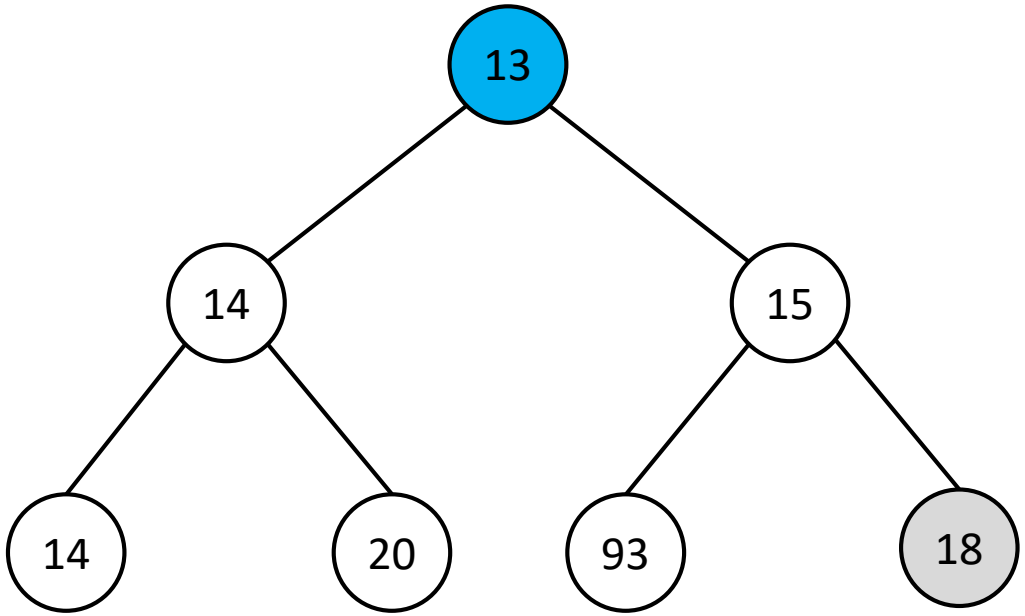


min ← 13

13	14	15	14	20	93	18

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
    A[1] ← A[Taille[A]]
```

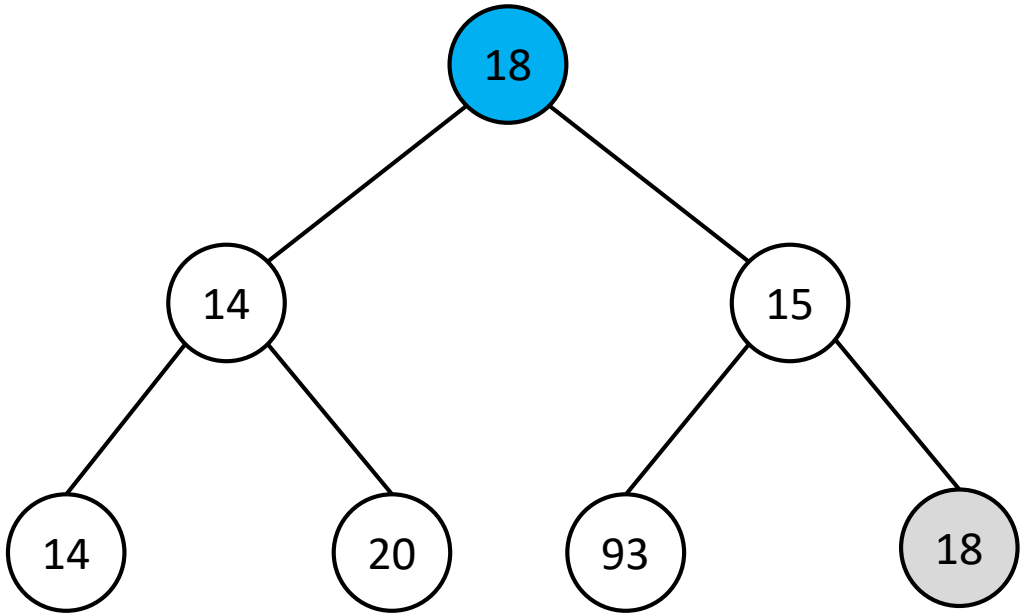


min ← 13

13	14	15	14	20	93	18

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
    A[1] ← A[Taille[A]]
```

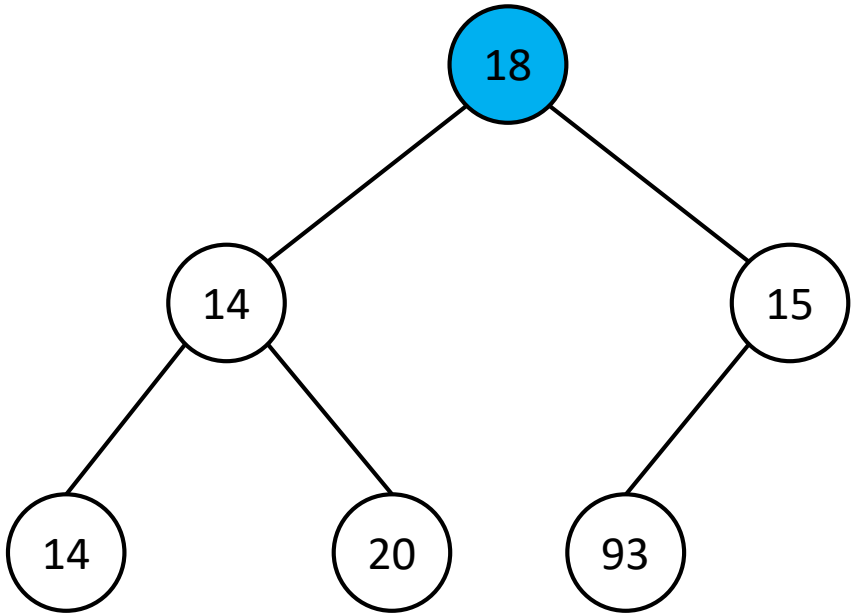


min ← 13

18	14	15	14	20	93	18

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
    A[1] ← A[Taille[A]]
    Taille[A] ← Taille[A]-1
```

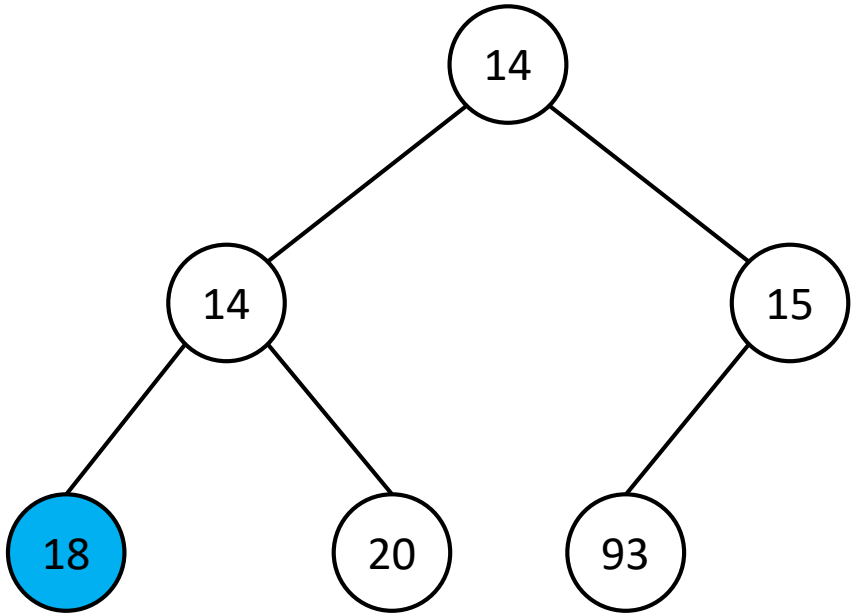


min ← 13

18	14	15	14	20	93

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
    A[1] ← A[Taille[A]]
    Taille[A] ← Taille[A]-1
    Entasser (A, 1)
```

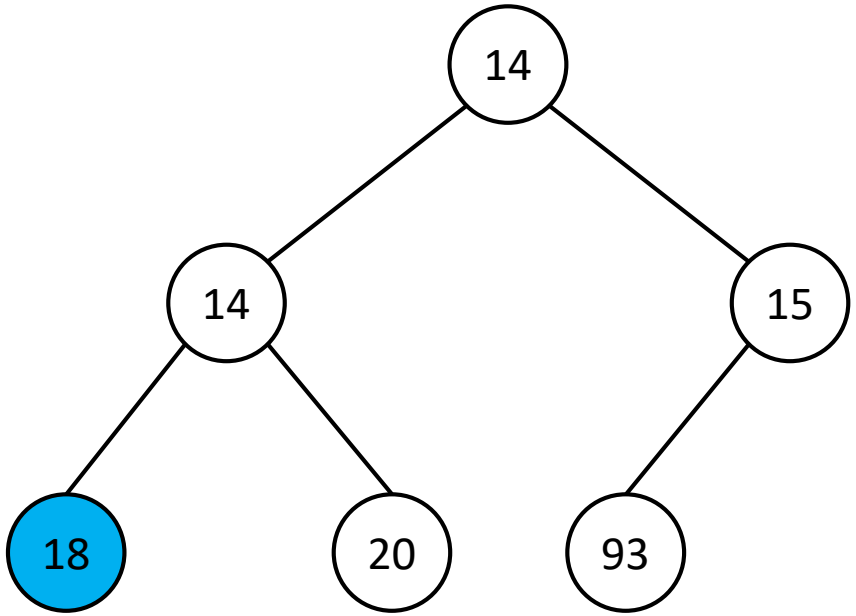


min ← 13

14	14	15	18	20	93

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
    A[1] ← A[Taille[A]]
    Taille[A] ← Taille[A]-1
    Entasser (A, 1)
    retourner min
```

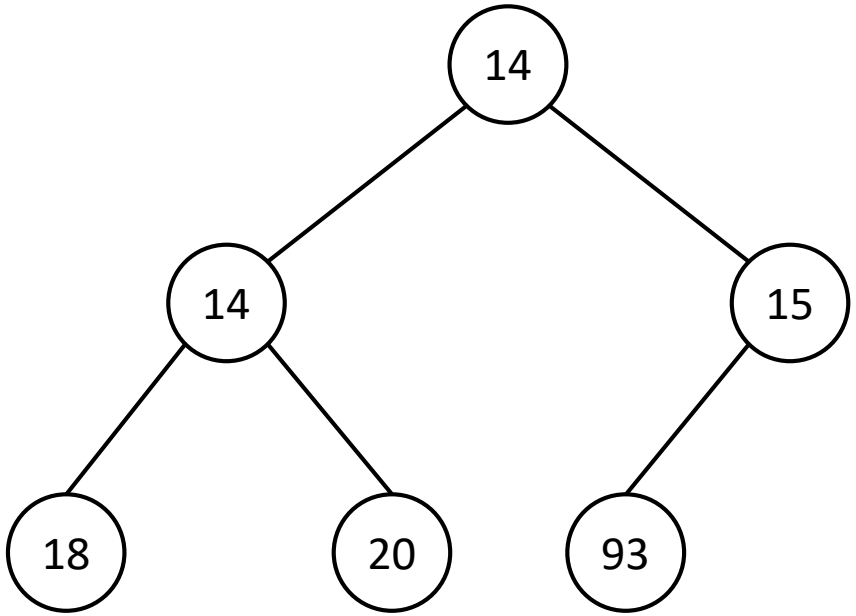


min ← 13

14	14	15	18	20	93

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
    A[1] ← A[Taille[A]]
    Taille[A] ← Taille[A]-1
    Entasser (A, 1)
    retourner min
```



min ← 13

14	14	15	18	20	93

Question 5

```
supprimer_min(A:Tas)
    min ← A[1]
    A[1] ← A[Taille[A]]
    Taille[A] ← Taille[A]-1
    Entasser (A, 1)
    retourner min
```

Complexité: O(h)

Question 6

Insérer 2 dans L1 : (1, 3, 1, 6, 9, 7, 12, 8, 7, 9, 10, 10)

Question 6

Insérer 2 dans L1 : (1, 3, 1, 6, 9, 7, 12, 8, 7, 9, 10, 10, 2)

Question 6

Insérer 2 dans L1 : (1, 3, 1, 6, 9, 7, 12, 8, 7, 9, 10, 10, 2)

Question 6

Insérer 2 dans L1 : (1, 3, 1, 6, 9, 2, 12, 8, 7, 9, 10, 10, 7)

Question 6

Insérer 2 dans L1 : (1, 3, 1, 6, 9, 2, 12, 8, 7, 9, 10, 10, 7)

Question 6

Insérer 2 dans L1 : (1, 3, 1, 6, 9, 2, 12, 8, 7, 9, 10, 10, 7)

Question 6

Supprimer le min dans L1 : (1, 3, 1, 6, 9, 2, 12, 8, 7, 9, 10, 10, 7)

Question 6

Supprimer le min dans L1 : (1, 3, 1, 6, 9, 2, 12, 8, 7, 9, 10, 10, 7)

Question 6

Supprimer le min dans L1 : (7, 3, 1, 6, 9, 2, 12, 8, 7, 9, 10, 10)

Question 6

Supprimer le min dans L1 : (7, 3, 1, 6, 9, 2, 12, 8, 7, 9, 10, 10)

Question 6

Supprimer le min dans L1 : (1, 3, 7, 6, 9, 2, 12, 8, 7, 9, 10, 10)

Question 6

Supprimer le min dans L1 : (1, 3, 7, 6, 9, 2, 12, 8, 7, 9, 10, 10)

Question 6

Supprimer le min dans L1 : (1, 3, 2, 6, 9, 7, 12, 8, 7, 9, 10, 10)

Question 6

Supprimer le min dans L1 : (1, 3, 2, 6, 9, 7, 12, 8, 7, 9, 10, 10)

Question 6

Supprimer le min dans L1 : (1, 3, 2, 6, 9, 7, 12, 8, 7, 9, 10, 10)

Question 7

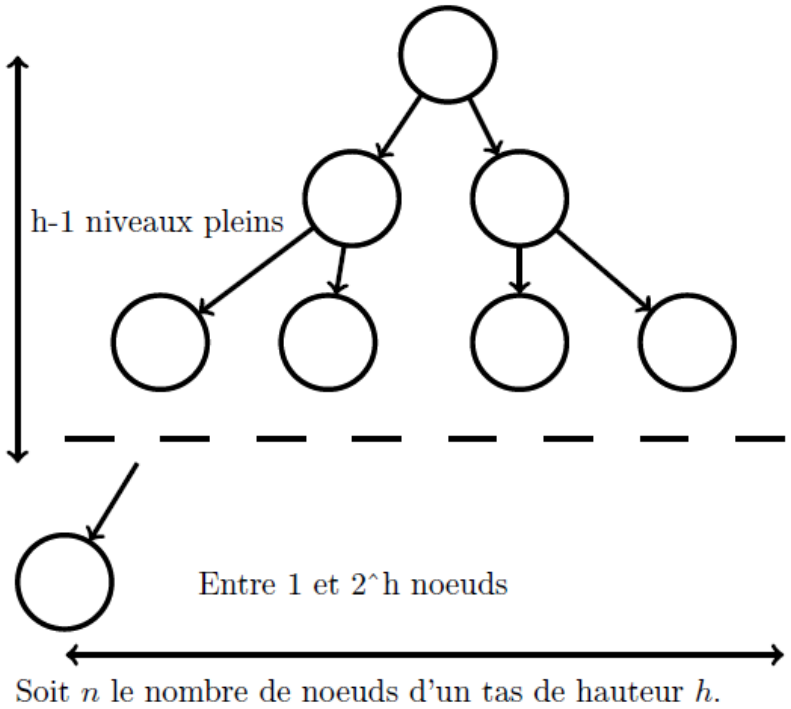
- Une liste triée est toujours un tas.

  ➔ Dans une liste triée : $A[i] \geq A[\lfloor i/2 \rfloor] \; \forall \; i > 1$

- Un tas n'est pas toujours une liste triée.

  ➔ Contre-exemple : L1.

Question 8



1

2

2^2

…

2^{h-1}

entre 1 et 2^h

Nombre de nœuds :

$$\sum_{i=0}^{h-1} 2^i + 1 \leq n \leq \sum_{i=0}^{h} 2^i$$

Question 8

$$\sum_{i=0}^{h-1} 2^i + 1 \leq \mathrm{n} \leq \sum_{i=0}^{h} 2^i$$

$$(2^h - 1) + 1 \leq \mathrm{n} \leq 2^{h+1} - 1$$

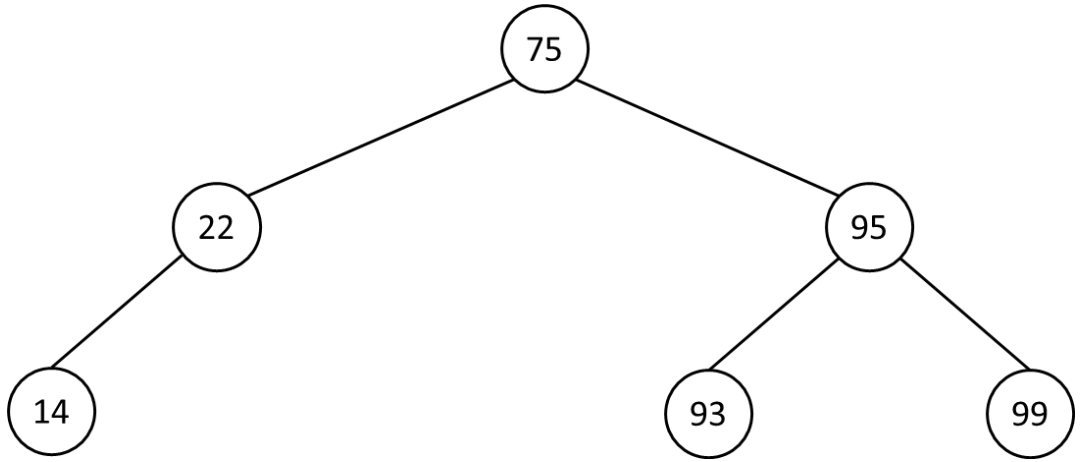
$$2^h \leq \mathrm{n} < 2^{h+1}$$

$$\mathrm{h} \leq log_2(\mathrm{n}) < \mathrm{h} + 1$$

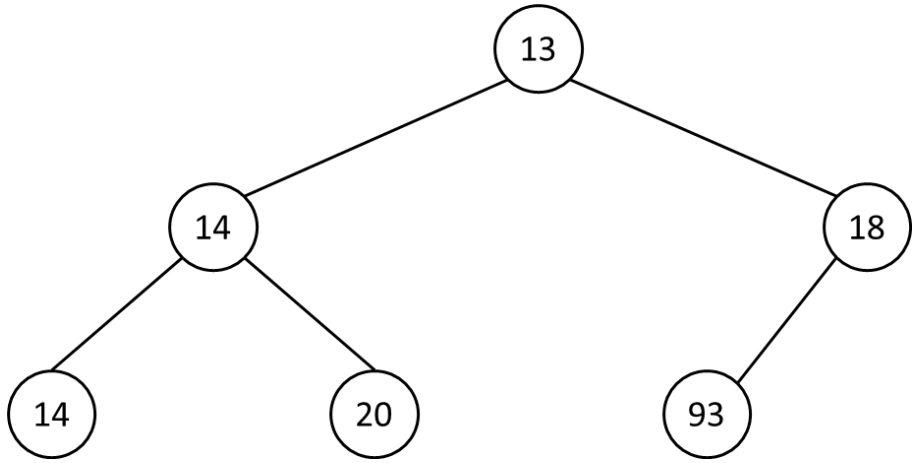
$$h = \lfloor log_2(\mathrm{n}) \rfloor$$

AVL vs TAS

AVL



TAS



Operation	AVL	TAS
Recherche	$\boldsymbol{O(log_2(n))}$	$O(n)$
Insertion	$O(log_2(n))$	$O(log_2(n))$
Suppression	$O(log_2(n))$	$O(log_2(n))$
Recherche MIN/MAX	$O(log_2(n))$	$\boldsymbol{O(1)}$
Insertion (Moyenne)	$O(log_2(n))$	$\boldsymbol{O(1)}$
Création	$O(n.log_2(n))$	$\boldsymbol{O(n)}$