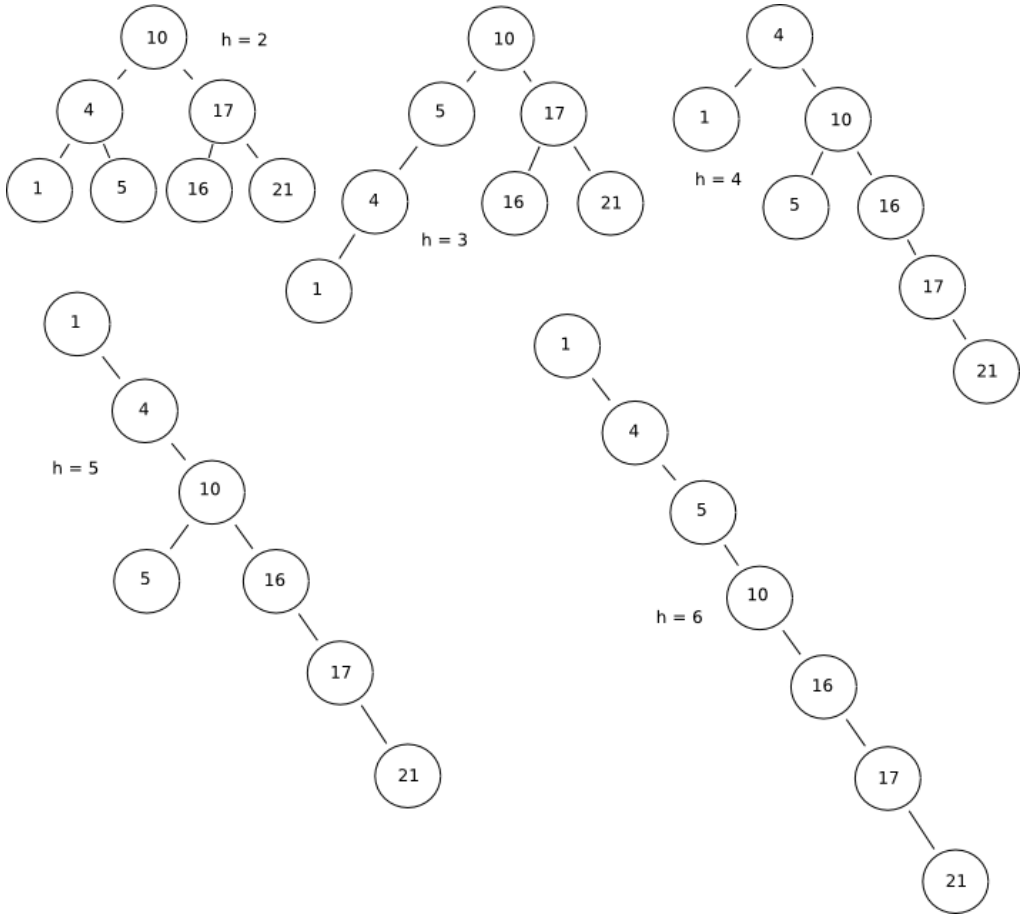
Séance de TD 05

1 Exercice 1

1. Dessinez les arbres binaires de recherche de hauteur 2,3,4,5 et 6 pour le même ensemble de clés S = 1,4,5,10,16,17,21 .
2. Donnez l'algorithme de l'opération ArbreRechercher(x,k) qui consiste à rechercher un noeud ayant la clé k dans un arbre binaire de recherche. Quelle est la complexité de cet algorithme ?
3. Donnez l'algorithme de l'opération ArbreInserer(T,z) qui consiste à insérer un noeud z ayant la clé clé(z) dans un arbre binaire de recherche T.
4. Dans les B-arbres, donnez l'algorithme de l'opération BArbreRechercher(x,k).

1.1 Question 1 : dessins des ABR avec hauteurs différentes

Note : on numérote les niveaux à partir de 0.



1.2 Question 2 : algorithme de ArbreRechercher(x,k)

```
ArbreRechercher(x,k)
  si x = Nil Alors
    retourner Nil
  Sinon Si k = clé(x) Alors
    retourner x
  Sinon Si k < clé(x) Alors
    ArbreRechercher(filsGauche(x), k)
  Sinon si k > clé(x) Alors
    ArbreRechercher(filsDroit(x), k)
  Finsi
Fin
```

L'algorithme est $O(n)$ dans le pire des cas, $O(log(n))$ dans le meilleur.

1.3 Question 3 : algorithme de ArbreInserer(T,z)

```
ArbreInserer(T,z)
  Si T = Nil Alors
    racine(T) <- z
    Fin
  Sinon si (racine(T) > clé(z))
    ArbreInserer(filsgauche(T), z)
  Sinon
    ArbreInserer(filsdroit(T), z)
  Finsi
Fin
```

1.4 Question 4 : algorithme de BArbreRechercher(x,k)

```
BArbreRechercher(x, k)
  i <- 1
  Tant que i <= n[x] et k > clé_i[x] Faire
    i <- i + 1
    Si i <= n[x] et k = clé_i[x] Alors
      retourner (x,i)
    Sinon si feuille(x) Faire
      retourner Nil
    Sinon si lire(c(x))
      BarbreRechercher(c(x), k)
```

```
    Finsi
  Fintantque
Fin
```

2 Exercice 2

Voici une liste de 14 éléments :

$$25, 60, 35, 10, 5, 20, 65, 45, 70, 40, 50, 55, 30, 15$$

On s'intéresse aux arbres binaires de recherche.

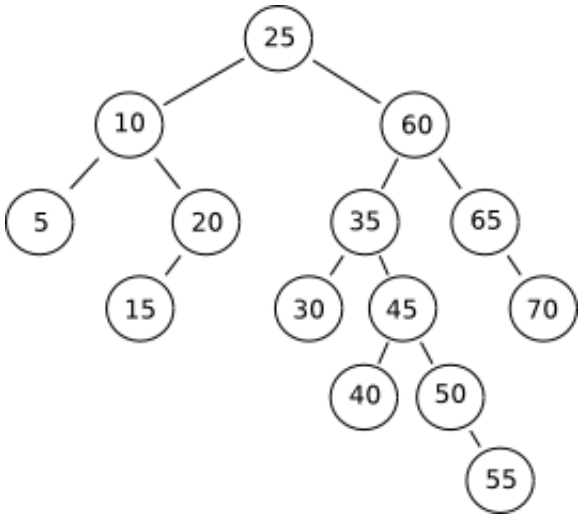
1. Rappelez ce qu'est l'opération d'adjonction aux feuilles.
2. Construire l'arbre binaire de recherche par adjonction des valeurs aux feuilles, dans l'ordre de la liste. Expliquez votre raisonnement.
3. Donner la liste infixée de l'arbre obtenu, en justifiant votre raisonnement. Quelle propriété a-t'elle ? Est-ce toujours le cas ? Donner l'algorithme correspondant à cette opération. Quelle est sa complexité ? Sachant que le temps d'exécution de ParcoursInfixe(x) pour un arbre à n noeuds est $T(n) = T(k) + T(n-k-1) + d$, avec $T(0) = c$, x est un noeud dont le sous-arbre de gauche a k noeuds et dont le sous-arbre de droite a n-k-1 noeuds ; c et d sont deux constantes positives.
4. Rappelez ce qu'est l'opération d'adjonction à la racine.
5. Construire l'ABR par adjonction des valeurs à la racine, dans l'ordre de la liste initiale.
6. Donnez la liste préfixée de l'arbre obtenu.
7. Donnez l'arbre obtenu par adjonction des valeurs aux feuilles, dans l'ordre de la liste prefixée que vous venez d'obtenir. Quelle propriété a-t'il vis à vis de l'arbre obtenu à la question 5 ?
8. Rappelez ce qu'est l'opération de suppression et donner sa complexité.
9. Donnez l'arbre obtenu par la suppression de 30 dans l'arbre de la question 7, en justifiant votre raisonnement.

2.1 Question 1 : l'adjonction aux feuilles

L'adjonction aux feuilles consiste à rechercher où devrait être une feuille dans l'ABR et à la placer à cet endroit.

2.2 Question 2 : l'adjonction aux feuilles

L'adjonction aux feuilles consiste à rechercher où devrait être une feuille dans l'ABR et à la placer à cet endroit.



2.3 Question 3 : liste infixée obtenue

On obtient la liste suivante : 5-10-15-20-25-30-35-40-45-50-55-60-65-70.

On remarque que la liste est triée.

```
ParcoursInfixe(x)
  Si x = Nil Alors
    retourner Nil
  Finsi
  ParcoursInfixe(filsGauche(x))
  Afficher(Val(x))
  ParcoursInfixe(filsDroit(x))
Fin
```

Le coût de cet algorithme est en $O(n)$.

Soit $n = 0$. T(0) = c $\Rightarrow O(1)$

Supposons que $T(n-1) = (c+d)(n-1) + c$

$T(n) = T(k) + T(n-k-1)$

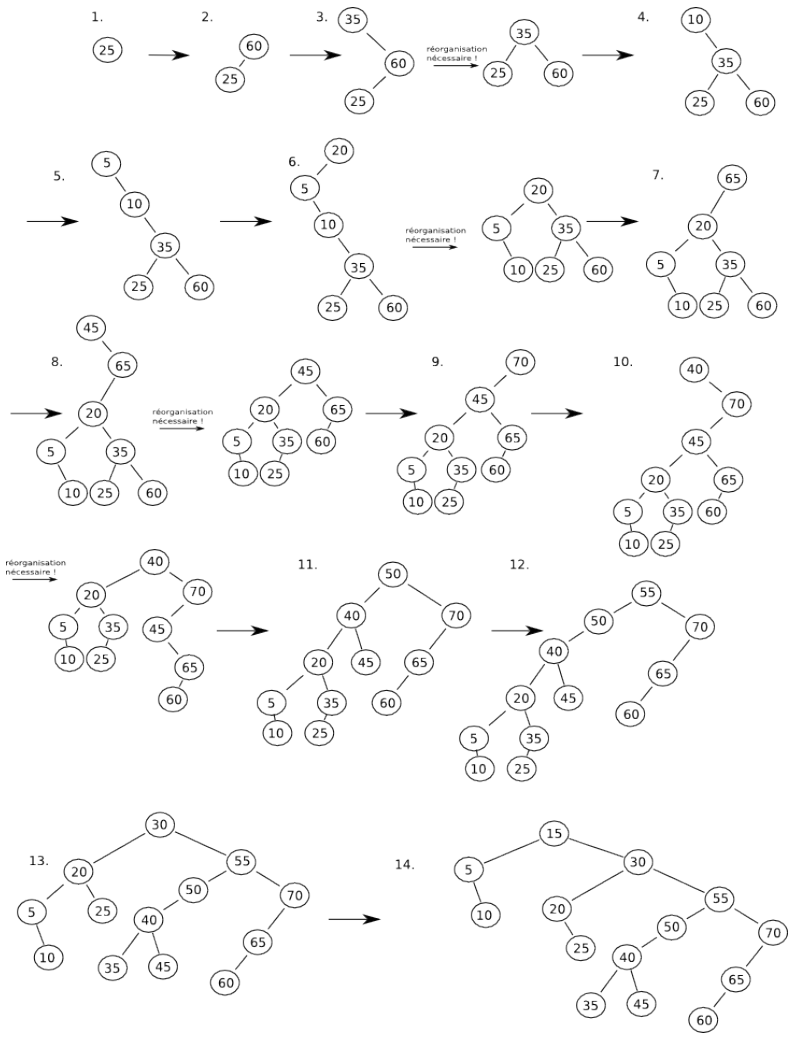
$T(n) = (c+d) \cdot k + (c+d)(n-k-1) + c$

$T(n) = (c+d) + c'$, avec d et $c' \geq 0$

2.4 Question 4 : adjonction à la racine

L'adjonction à la racine consiste à construire l'ABR en plaçant la nouvelle valeur à la racine.

2.5 Question 5 : construction par adjonction à la racine



2.6 Question 6 : liste préfixée

La liste préfixée s'obtient en sortant la racine avant son sous-arbre de gauche, et ensuite son sous-arbre droit.

On obtient donc la liste suivante :

15-5-10-30-20-25-55-50-40-35-45-70-65-60

2.7 Question 7 : adjonction aux feuilles

On obtient le même arbre que celui de la question 5.

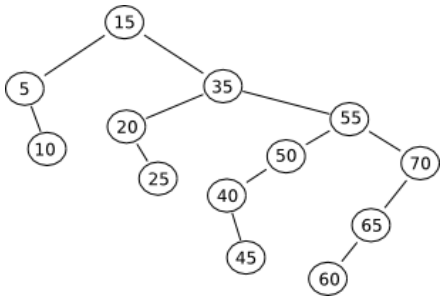
2.8 Question 8 : suppression

Cet algorithme se déroule suivant plusieurs étapes :

- Recherche de l'élément à supprimer
- Si le noeud à supprimer est une feuille, l'enlever de l'arbre
- Si le noeud à supprimer possède un fils, on l'enlève et on relie son parent à son fils.

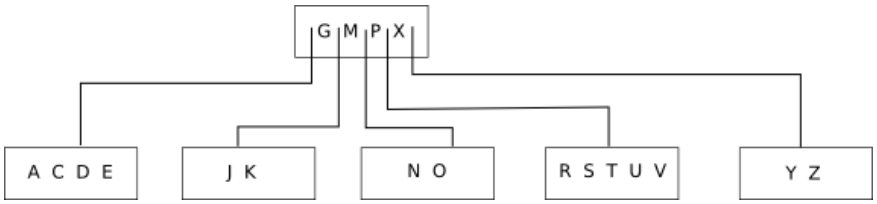
La complexité de cet algorithme est en $O(log(n))$.

Exemple avec la suppression de 30 : On interchange avec le plus petit des fils droit (ici, 35) et on supprime 30.



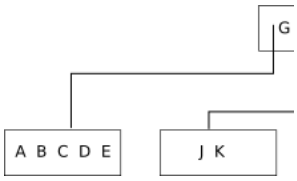
3 Exercice 3

Soit le B-arbre suivant :



3.1 Question 1

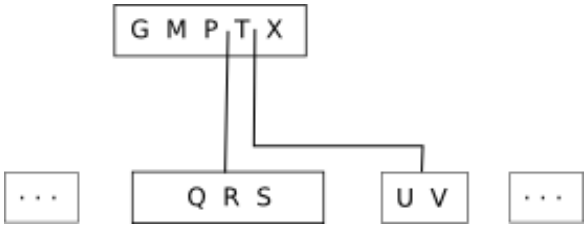
Insérer B dans l'arbre



Le cardinal de la feuille contenant B est de 5, ce qui est le maximum.

3.2 Question 2

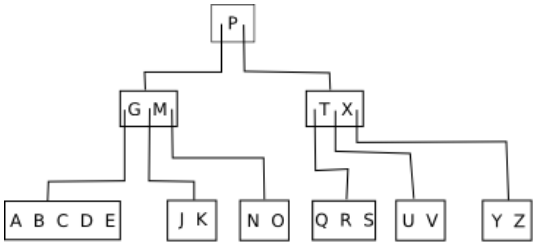
Insérer Q dans l'arbre



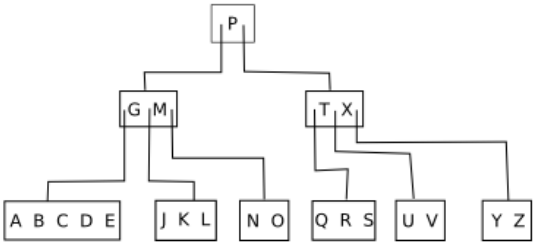
3.3 Question 3

Insérer L dans l'arbre

On est obligé de réorganiser la racine au préalable :



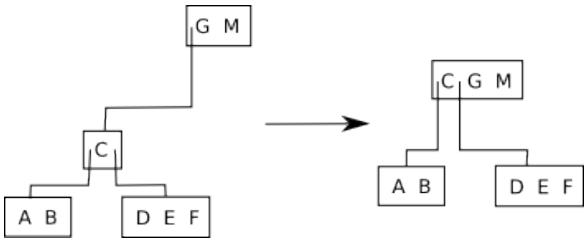
Il est alors possible de rajouter le L :



3.4 Question 4

Insérer F dans l'arbre

Seule la partie de l'arbre concernée par le changement est représentée ici.



3.5 Question 5

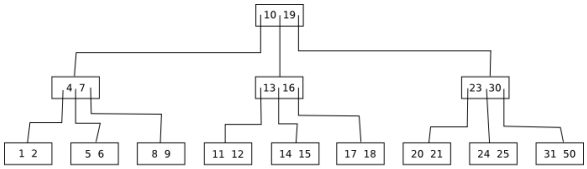
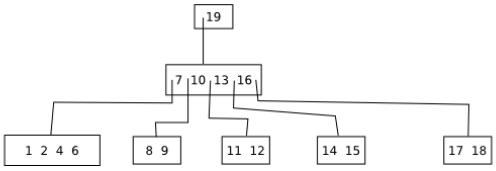
Donnez l'algorithme de l'ajout d'une clé dans un B-Arbre

?

3.6 Question 6

Suppression d'un noeud au sein d'un 2B-Arbre

Soit le B-arbre suivant :



Le but de l'exercice est de supprimer le 5. Il est alors possible, après suppression de réaliser des regroupements :