Algorithmique avancée: Contrôle Partiel du Lundi 21 Octobre 2019.

Le barème est donné à titre indicatif. La compréhension du sujet faisant partie de l'épreuve, **on ne répondra à aucune question**. Si vous rencontrez des ambiguïtés, vous expliquerez **sur votre copie** comment vous les interprétez.

Durée 2h. Seul document autorisé: 1 feuille A4 recto-verso.

On rappelle que

$$\sum_{i=1}^{n} \lfloor \log i \rfloor \in \Theta(n \log n)$$

I - Tas binaires (10 points)

On considère le tableau $T1$ suivant : $T1 =$

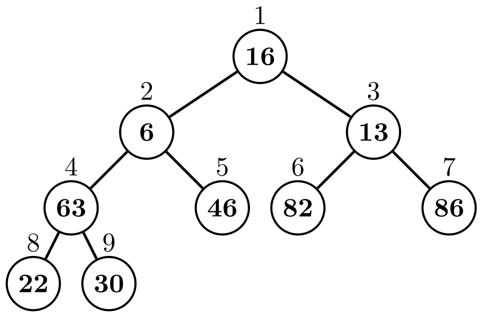
16	6	13	63	46	82	86	22	30

Ce tableau contient 9 éléments ($T1$.LENGTH$= 9$).

1) **(2 pt)** Créez un tas binaire MAXIMAL $H1$ à partir de $T1$ grâce à l'opération BUILDHEAP. Vous donnerez sa représentation sous forme d'arbre et sous forme de tableau.

Corrigé

tableau initial $T1$



après BuildHeap: $H1$

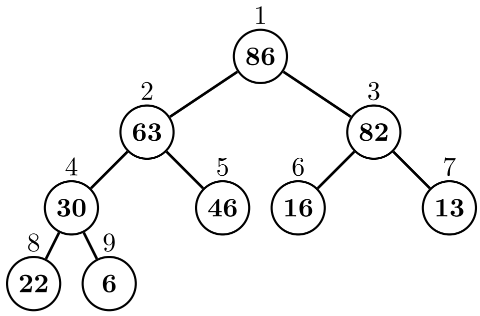


Tableau du tas obtenu $H1$:

86	63	82	30	46	16	13	22	6

2) **(1 pt)** Que donne l'opération REMOVE sur $H1$? Vous décrirez l'arbre obtenu et le tableau.

Corrigé

après Remove($H1$)

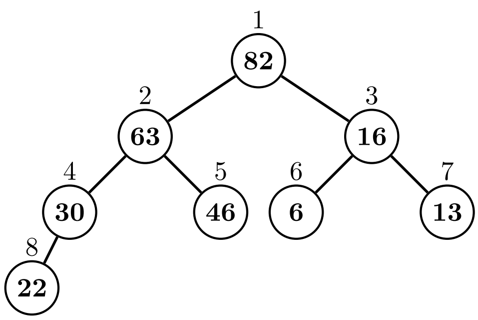


Tableau obtenu :

82	63	16	30	46	6	13	22

On considère l'algorithme suivant :

Procedure ConstruireTas(T)

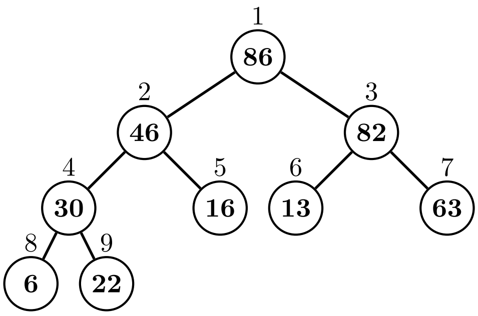
T.Size $\leftarrow 1$
for $i = 2$ *to* T.Length **do**
    Add(T,$T[i]$)

où Add est la procédure d'ajout d'un élément dans un tas binaire MAXIMAL.

3) **(2 pts)** Appliquez ConstruireTas au tableau $T1$. Vous donnerez la représentation du tableau obtenu, nommé $H2$, sous forme d'arbre et sous forme de tableau. $H2$ est-il un tas binaire?

Corrigé

$H2 = ConstruireTas(T1)$



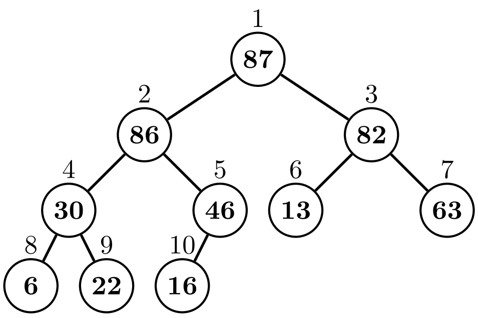
86	46	82	30	16	13	63	6	22

$H2$ est bien un tas binaire puisqu'on le construit avec des Add successifs.

4) **(0.5 pt)** Proposez un élément à ajouter à $H2$ qui demandera le plus d'opérations possibles, et décrivez le tableau $H3$ résultant.

Corrigé

Il suffit de prendre un élément supérieur à 86, soit 87. On obtient: $H3 = Add(H2, 87)$



87	86	82	30	46	13	63	6	22	16

Dans le cas général où le tableau contient n éléments, on désire connaître l'efficacité de ConstruireTas. Pour cela, on s'intéresse aux questions suivantes :

5) **(1 pt)** On considère le $i$ème élément du tableau représentant un tas binaire de n éléments, à quelle profondeur est-il? Justifiez.

Corrigé

Soit p la profondeur de l'élément i, le dernier élément du niveau au dessus de i est à profondeur $p-1$, c'est donc l'élément $2^0 + ... + 2^{p-1}$ c'est donc le $2^p - 1$ème. i est donc tel que $2^p \leq i < 2^{p+1}$. C'est-à-dire que $p \leq \log_2 i \leq p+1$, la profondeur de i est donc $\lfloor \log_2 i \rfloor$

6) **(1 pt)** Étant donné un tableau de n éléments, proposez une configuration qui soit le pire cas possible en complexité pour ConstruireTas, c'est à dire qui impose que ConstruireTas effectue le maximum d'opérations possibles.

Corrigé

Le pire c'est quand on ajoute à chaque fois l'élément MAX au tas, il doit tout remonter jusqu'à la racine. Une pire configuration possible est donc quand le tableau initial est trié par ordre croissant.

7) **(1 pt)** Combien cela demandera-t'il d'opérations en notation Θ en fonction de n ? Vous justifierez votre réponse en sommant les opérations à faire pour chacun des n éléments.

Corrigé

Le $i$ème élément est à profondeur $\lfloor \log i \rfloor$, il fait donc $\lfloor \log i \rfloor$ échanges.
Pour tous les éléments ça donne donc : $nbop = \sum_{i=1}^{n} \lfloor \log_2 i \rfloor \in \Theta(n \log n)$.
Notons que $\log(n!) \in \Omega(n \log n)$ car $\log n! = \log 1 + \log 2 + ... + \log n \geq \log n/2 + ... \log n$ donc $\geq n/2 \times \log n/2$ et $\log(n!) \in O(n \log n)$ car $\log n! = \log 1 + \log 2 + ... + \log n \leq n \log n$.

8) **(0.5 pt)** Proposez une configuration qui soit un meilleur des cas pour ConstruireTas dans le cas général d'un tableau à n éléments (c'est-à-dire une configuration pour laquelle ConstruireTas effectuera le moins d'opérations possibles).

Corrigé

un tableau trié dans l'ordre décroissant : aucun échange n'est nécessaire, ou bien un tableau qui est déjà un tas MAXIMAL.

9) **(0.5 pt)** Comparez les complexités spatiales et temporelles de ConstruireTas en pire cas à celles de BuildHeap.

Corrigé

$T_{\text{ConstruireTas}}(n) \in \Theta(n \log n)$ et $T_{\text{BuildHeap}}(n) \in \Theta(n)$ et $S_{\text{ConstruireTas}}(n) \in \Theta(1)$ et $S_{\text{BuildHeap}}(n) \in \Theta(1)$ et

10) **(0.5 pt)** Existe-t'il des configurations où ConstruireTas et BuildHeap font le même nombre d'échanges?

Corrigé

oui quand les tableaux sont déjà des tas Maximaux

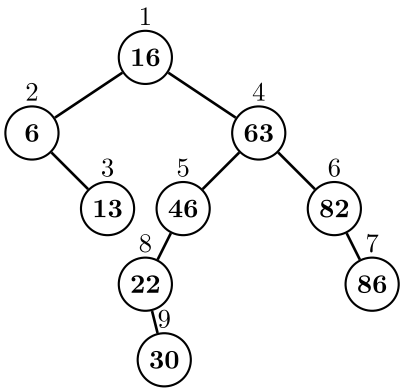
II - AVL (7 pts)

On considère le même tableau $T1$ =

16	6	13	63	46	82	86	22	30

1) **(1 pt)** Créez l'arbre binaire de recherche $A1$ obtenu par ajouts successifs des éléments de $T1$.

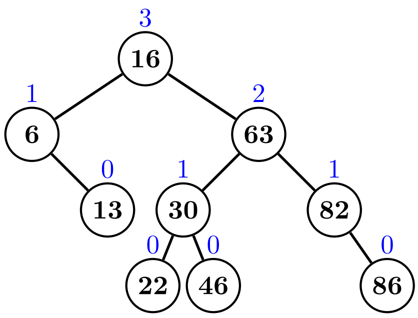
Corrigé



Puis transformez $A1$ en un AVL (s'il ne l'est pas déjà). Vous noterez les hauteurs des sous-arbres gauches et droits de chaque noeud. On rappelle que par convention un arbre vide à une hauteur de -1.

Corrigé

LeftRotate(22) puis RightRotate(46)



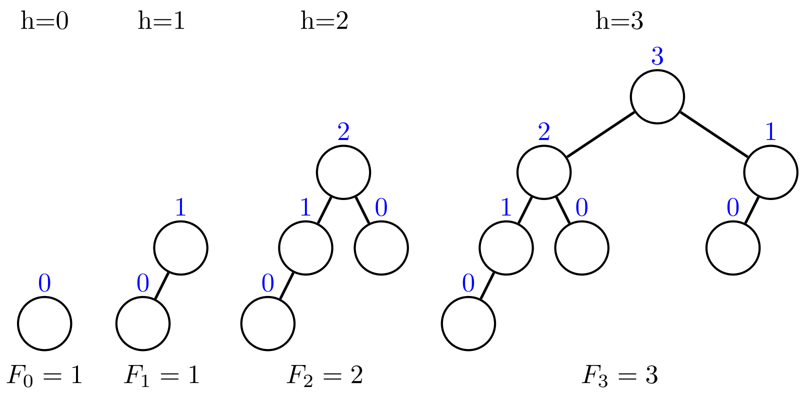
2) **(1 pt)** Quelles sont les insertions qui demanderaient des rotations afin de rétablir la propriété d'AVL? On demande un exemple de valeur à insérer pour chaque emplacement qui pose problème (on ne demande pas le résultat de l'insertion).

Corrigé

à gauche de 13 (7 par exemple), à droite de 13 (14 par exemple), à gauche de 86 (83 par exemple), à droite de 86 (87 par exemple).

3) **(1.5 pts)** Pour chaque hauteur 0, 1, 2 et 3, dessinez un exemple d'AVL ayant cette hauteur et le moins de noeuds possible. Pour chacun de ces AVL, comptez le nombre de feuilles obtenues (notés respectivement F_0, F_1, F_2, F_3).

Corrigé



4) **(1.5 pts)** Proposez une expression du nombre de feuilles F_h, de l'AVL A_h de hauteur h ayant le moins de noeuds possibles, en fonction des nombres de feuilles des AVL A_{h-1} de hauteur $h-1$ et A_{h-2} de hauteur $h-2$ ayant le moins de noeuds possibles (leur nombre de feuilles étant notés respectivement F_{h-1} et F_{h-2}). Justifiez.

En utilisant le fait que le ième terme de la suite de Fibonacci u_i est tel que $u_i > \frac{\varphi^i - 1}{\sqrt{5}}$ où $\varphi = \frac{1+\sqrt{5}}{2}$ (φ est appelé le nombre d'or[1]), en déduire une borne asymptotique supérieure de la hauteur d'un AVL fonction du nombre de noeuds (donc préciser le f dans l'expression "la hauteur h d'un AVL de taille n est en $O(f(n))$").

Corrigé

Pour avoir le plus petit AVL de hauteur h, A_h, on doit avoir au moins un sous-arbre de hauteur $h-1$, si on veut que A_h soit minimal alors son deuxième sous-arbre doit être le moins haut possible, mais comme c'est un AVL il doit être de hauteur $\geq h-2$ sinon ça déséquilibre trop. Donc on prend comme sous-arbres les arbres minimaux de hauteur $h-1$ et $h-2$, le nombre de feuilles est donc $F_h = F_{h-1} + F_{h-2}$.
F_h est donc un terme de la suite de Fibonacci. Donc $F_h > \frac{\varphi^h - 1}{\sqrt{5}}$. Ce nombre F_h est plus petit que le nombre total de noeuds. Donc la taille n de tout AVL est t.q. $n > \frac{\varphi^h - 1}{\sqrt{5}}$ i.e., $\sqrt{5}n + 1 > \varphi^h$, or quand $n > 1$, $\sqrt{5}n + 1 < 3n$, donc $\varphi^h < 3n$ i.e., $h \log(\varphi) < \log n + \log 3$, i.e., $h < \frac{\log n + \log 3}{\log \varphi}$ i.e. $h \in O(\log n)$.

5) **(1 pt)** Reprendre la question précédente en donnant les AVLs avec le plus de noeuds possibles pour une hauteur de 0, 1, 2 et 3. Donnez une expression générale du nombre de noeuds dans de tels AVL en fonction de la hauteur h. En déduire une borne asymptotique inférieure de la hauteur d'un AVL en fonction du nombre de noeuds (donc préciser le g dans l'expression "la hauteur h est en $\Omega(g(n))$"). Conclure sur une expression en Θ de la hauteur d'un AVL.

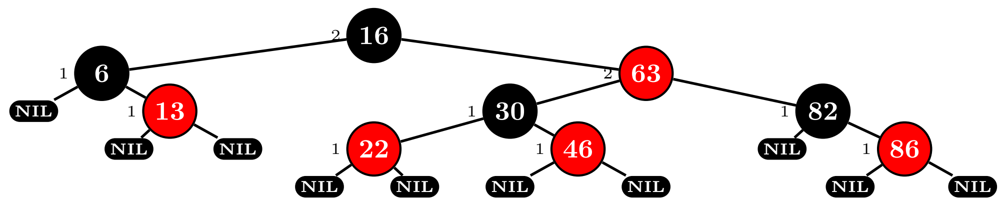
Corrigé

Un AVL de hauteur h ayant le plus de noeuds possibles est un arbre binaire plein c'est-à-dire avec $2^{h+1} - 1$ noeuds. A_0 possède 1 seul noeud, A_1 possède 3 noeuds, A_2 possède 7 noeuds et A_3 possède 15 noeuds. Tout AVL vérifie donc $n \leq 2^{h+1} - 1$ c'est-à-dire $n + 1 \leq 2^{h+1}$, $n \leq 2^{h+1}$ donc $h + 1 \geq \log_2 n$ et $h \geq \log_2 n - 1$. Or $\log_2 n - 1 \geq 1/2 \log_2 n$ quand $1/2 \log_2 n \geq 1$ (i.e. $n \geq 2^2$) donc $h \in \Omega(\log n)$. Et donc avec la question 4, $h \in \Theta(\log n)$.

[1]Plus précisément $u_i = \frac{\varphi^i - \varphi'^i}{\sqrt{5}}$ où $\varphi' = \frac{1-\sqrt{5}}{2}$ mais l'inégalité suffit pour l'exercice.

6) **(1 pt)** Peut-on créer un arbre rouge-noir à partir de l'AVL obtenu à la question 1? si oui dessinez-le sinon expliquez.

Corrigé



III - B-Arbres (3 pts)

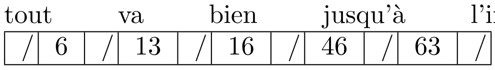
On considère encore le même tableau $T1 =$

16	6	13	63	46	82	86	22	30

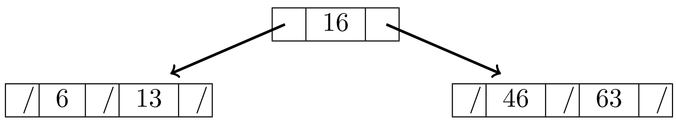
1) **(2.5 pts)** Créez le B-arbre $B1$ de degré $t = 3$ résultant de l'insertion successive des éléments du tableau $T1$. Vous détaillerez les passages d'éclatement des noeuds.

Corrigé

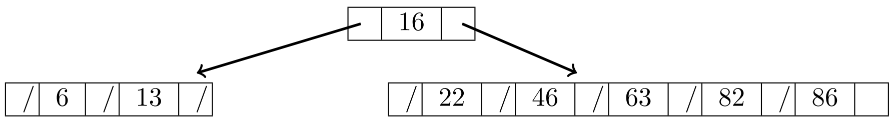
(a) tout va bien jusqu'à l'insertion de 16,6,13,63, 46:



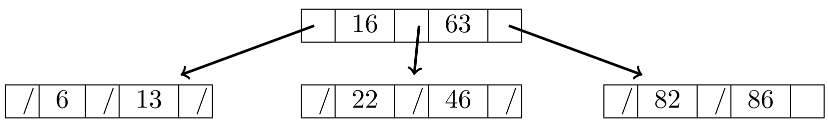
(b) insertion de 82 on éclate le noeud plein



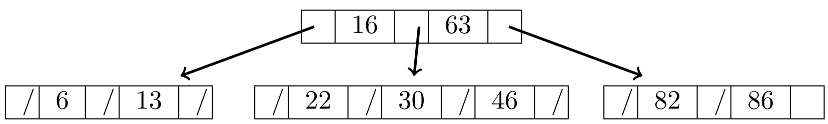
(c) on continue sans pb d'insertion 82, 86 et 22



(d) avant d'insérer 30 on éclate



(e) puis on insère 30



2) **(0.5 pt)** Décrivez le B-arbre obtenu après insertion de 64 et 45 dans $B1$.

Corrigé

insertion de 64 et de 45 sans besoin d'éclatement.