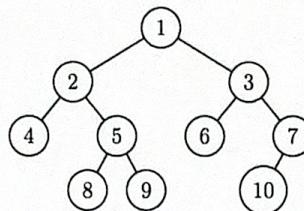


EA4 – Éléments d’algorithmique

TD n° 8 : arbres binaires de recherche

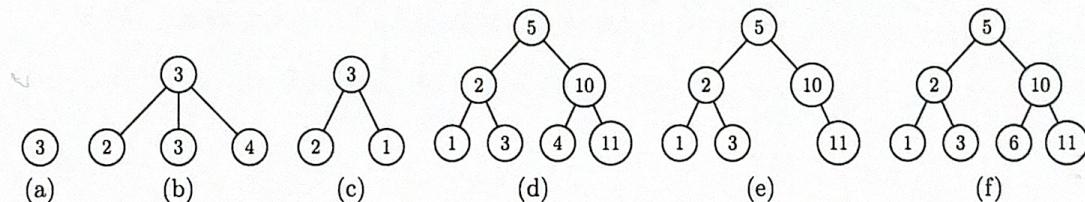
Exercice 1 : parcours d’arbres binaires

1. Vérifier que les sommets de l’arbre ci-dessous sont étiquetés dans l’ordre d’un parcours en largeur.
2. Lister les sommets de l’arbre binaire ci-dessous selon les ordres préfixe, infixé et suffixe.



Exercice 2 : arbres binaires de recherche

1. Parmi les arbres ci-dessous, lesquels sont des ABR ? Justifier.



2. Dessiner des ABR de toutes les hauteurs possibles pour l’ensemble de clés $\{1, 2, 3, 4, 5, 6, 7\}$.
3. Combien y a-t-il d’ABR d’une forme donnée pour un ensemble de n valeurs fixées ?
4. Donner un algorithme de complexité linéaire en la taille de l’arbre qui teste si un arbre binaire est un arbre binaire de recherche.
5. À partir de l’arbre vide, insérer successivement les nœuds d’étiquette 5, 9, 4, 2, 7, 1, 6, 3, et 8 en appliquant l’algorithme d’insertion dans un ABR vu en cours.

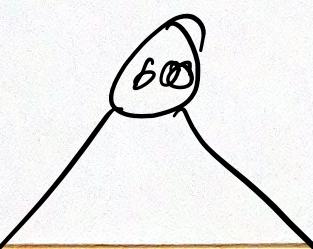
Exercice 3 : manipulation d’insertions

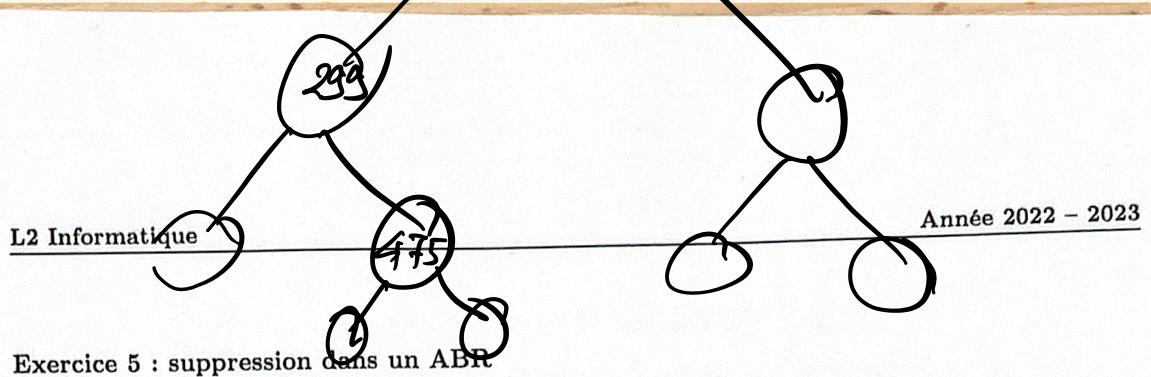
1. Dessiner les trois ABR obtenus par insertion successive pour les différents ordres suivants :
 - 1, 9, 8, 2, 3, 7, 6, 4, 5
 - 4, 2, 1, 3, 6, 5, 8, 7, 9
 - 4, 1, 2, 3, 8, 6, 5, 7, 9
2. Proposer, si possible, d’autres ordres menant aux mêmes ABR.
3. Pour chacun des ABR, dénombrer les ordres possibles.
4. On appelle arbre binaire *parfait* un arbre binaire dont toutes les feuilles sont à la profondeur maximale – autrement dit, tous ses niveaux sont entièrement remplis. Donner une équation de récurrence pour $N(h)$, le nombre d’ordres possibles pour l’ABR parfait de hauteur h .

Exercice 4 : recherche dans un ABR

Indiquer à quelle(s) suite(s) d’entiers $a_1 \rightarrow a_2 \rightarrow \dots$ un entier x pourrait être comparé ($x \leq a_1$, puis $x \leq a_2$, puis etc) lors de la recherche de ce x dans un ABR.

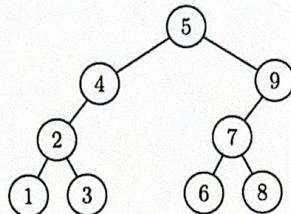
- 600 → 299 → 475 → 388 → 430 → 399 → 395 → 401
- 237 → 266 → 245 → 244 → 239 → 242 → 243 → 241
- 500 → 590 → 536 → 569 → 583 → 586 → 585 → 584
- 478 → 140 → 259 → 453 → 375 → 272 → 271 → 273
- 276 → 655 → 801 → 875 → 816 → 811 → 814 → 812





Exercice 5 : suppression dans un ABR

1. Supprimer de l'ABR ci-dessous les nœuds d'étiquette 1, puis 2, puis 5, puis 6, en appliquant l'algorithme de suppression d'un nœud dans un ABR vu en cours.
2. Recommencer en utilisant cette fois la variante de l'algorithme de suppression qui remplace un nœud supprimé par son prédécesseur.



Exercice 6 : Échauffement pour les tables de hachage

On considère une table de hachage T à adressage fermé avec résolution des collisions par chaînage, de longueur $m = 11$, utilisant la fonction de hachage $h(k) = k \bmod m$.

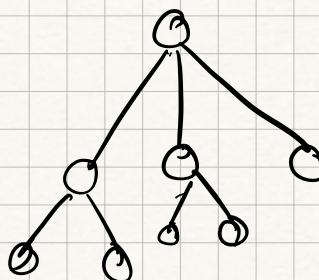
Insérer successivement les clés 5, 28, 19, 15, 20, 33, 37, 17, 26 et 10 dans T , puis supprimer ensuite les clés 19, 37 et 17.

On fixe maintenant pour la table un taux de remplissage maximal de $\frac{3}{4}$. Comment sont alors réalisées ces insertions et suppressions ?

Explication :

Arbre

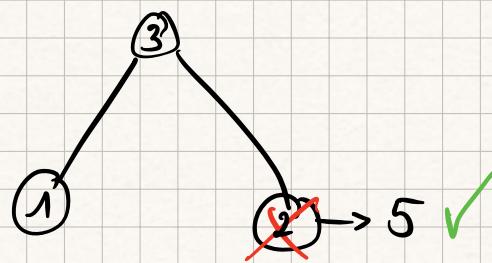
:



Binaire : au plus 2 fils

recherches : Toutes les valeurs seen arbre à gauche est plus petit que son parent en haut

Toutes les valeurs seen arbre à droite est plus grand que son parent en haut



Ex 2

1x a) Oui

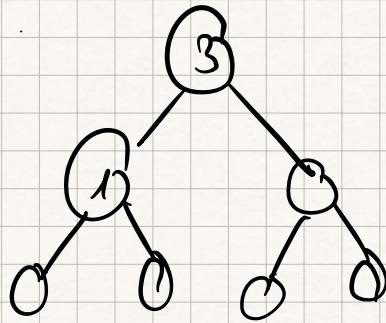
b) Non car il a plus que 2 fils

c) Non car le noeud à droite est plus petit que son parent en haut $1 < 3$

d) Non car $4 < 5$

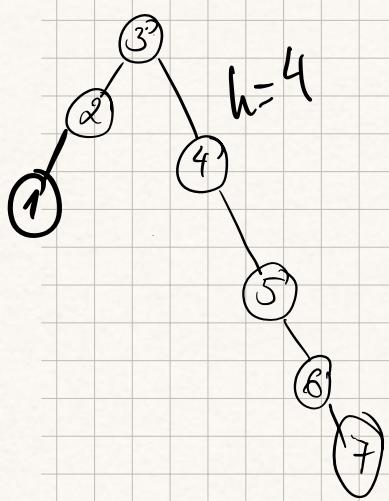
e) Oui

für Bui

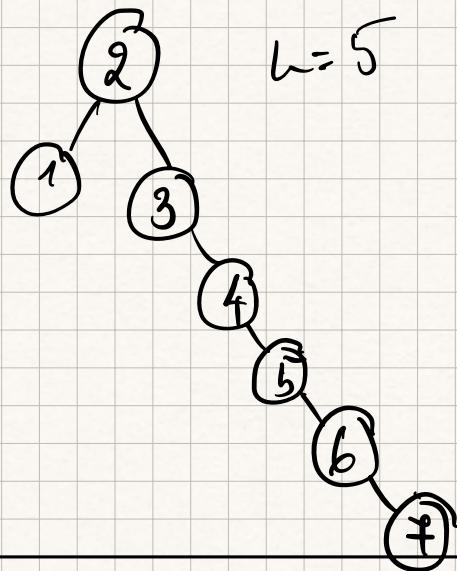


höhe = 2

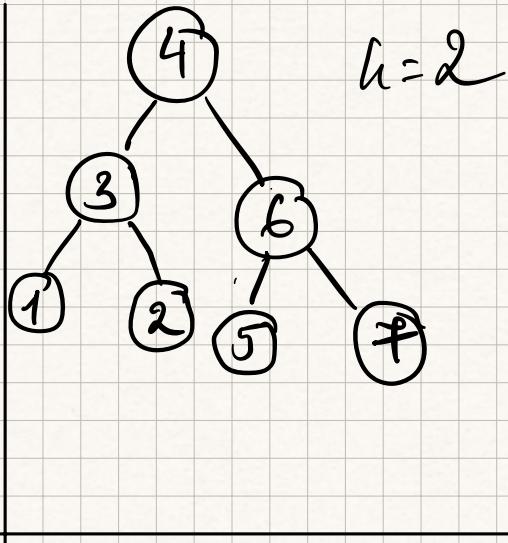
24



h=4

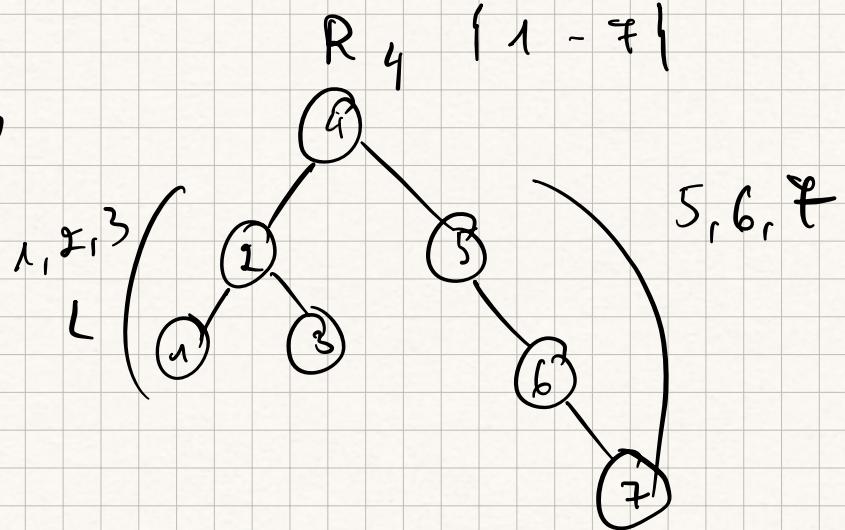


h=5



h=2

3y



R₄

{1 - 7}

1, 2, 3
L

5, 6, 7

Pour une forme fixée d'arbre de taille n ,
le sous arbre gauche de la racine contient n_1 noeuds
le sous arbre droit de la racine contient n_2 noeuds
avec $n_1 + n_2 + 1 = n$. Donc la clé de la racine doit
avoir n_1 valeurs qui lui sont strictement inférieures
et n_2 valeurs qui lui sont strictement supérieures

Comme $n = n_1 + n_2 + 1$, il y a seul choix de clé pour
la racine

Les n_1 valeurs inférieures vont dans le sous-arbre gauche
qui est aussi un ABR : on peut lui appliquer le même
raisonnement, il y a seul choix

Pareil à droite

Il y a donc un seul choix

def est_Arbre(T, mini_val, maxi_val):
 if T == None:
 return maxi_val >= mini_val
 if T.val < mini_val or T.val >= maxi_val:
 return False

return est_Arbre(T.gauche, mini_val, min(maxi_val, T.val))

and est_Arbre(T.droite, max(mini_val, T.val), maxi_val)

$\hookrightarrow \ominus(h)$

Ex 6

$$\begin{matrix} h(y) \\ h(x) \end{matrix}$$

11



$$h(k) = k \bmod m$$

$$h(x) = xr[11]$$

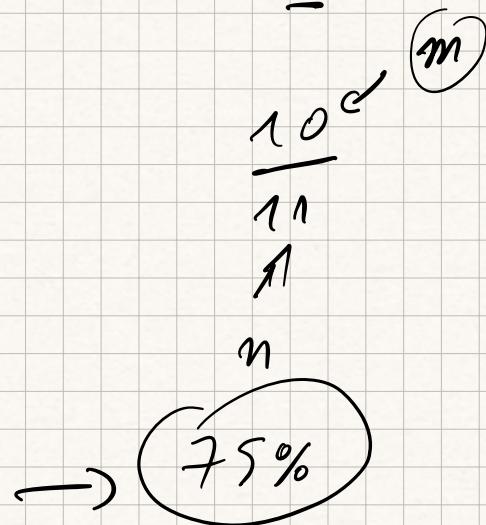
$$\begin{aligned} h(5) &= 5[11] \\ &= 5 \end{aligned}$$

$$\begin{aligned} h(8) &= 8 - 2[11] \\ &= 6 \end{aligned}$$

0	2	3	4	5	6	7	8	9	10
33				↓	↓	↓	↓	↓	↓
	*	25	28	X	20	1			

Annotations: A bracket groups the first four slots (0-3). Arrows point from 25 and 28 to their respective slots. An 'X' marks slot 4. An arrow points from 20 to slot 5. An '1' is written below slot 7.

Supp: 19 - 37 - 17



Taux de remplissage : $\frac{\text{nb éléments dans la table}}{\text{taille de la table}}$

$$\left\langle \frac{3}{4} \right\rangle$$

