

TD 7

Exercice 1

arbre de Fibonacci : $T_0 = \emptyset, T_1 = \bullet \rightarrow$ 2 cas de base
 $T_k = (\bullet, T_{k-2}, T_{k-1})$ si $k \geq 2$

Q1 $T_2 = (\bullet, T_1, T_0)$:

$T_3 = (\bullet, T_2, T_1)$

$T_4 = (\bullet, T_3, T_2)$

$T_5 = (\bullet, T_4, T_3)$



pas H-équilibré
 car $G = \emptyset$
 $D = 1$
 $h(G) = 0$
 $h(\emptyset) = 2$

Q2 $\forall k, n(T_k) = F_{k+2} - 1, k \geq 0$

• Base : $n(T_0) = 0 = F_2 - 1$

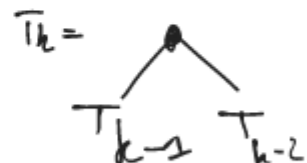
$n(T_1) = 1 = F_3 - 1$

car $F_2 = F_0 + F_1 = 1$

or $F_3 = F_2 + F_1 = 2$

Donc la base est vérifiée

• Induction : Soit $T_k = (\bullet, T_{k-2}, T_{k-1})$



On suppose que $n(T_{k-2}) = F_{k+1} - 1$
 $n(T_{k-1}) = F_k - 1$

$n(T_k) = n(T_{k-2}) + n(T_{k-1}) + 1$ (!!!)

Donc $n(T_k) = \underline{F_{k+2} - 1} + \underline{F_k - 1} + 1$

et $F_{k+2} + F_k = F_{k+2}$

Donc $\underline{n(T_k) = F_{k+2} - 1}$

Conclusion : $\forall k \geq 0 \quad n(T_k) = F_{k+2} - 1.$

(Q3)

Rappel : T est H-équilibré

ssi T est vide

ou $T = \begin{array}{c} \alpha \\ \swarrow \searrow \\ G \quad D \end{array}$

avec $|h(G) - h(D)| \leq 1$

G et D sont H-équilibrés


$P(T_k) = \begin{cases} h(T_k) = k \\ T_k \text{ est H-équilibré} \end{cases}$

• Base :
 - $h(T_0) = h(\emptyset) = 0$ et \emptyset est H-équilibré
 - $h(T_1) = h(\bullet) = 1$ et \bullet est H-équilibré (feuille)

• Induction : Soit $T_k = (\bullet, T_{k-1}, T_{k-2})$ $T_k = \begin{array}{c} \bullet \\ \swarrow \searrow \\ T_{k-1} \quad T_{k-2} \end{array}$
 On suppose que $\begin{cases} h(T_{k-1}) = k-1 \\ h(T_{k-2}) = k-2 \\ T_{k-1} \text{ et } T_{k-2} \text{ sont H-équilibrés} \end{cases}$

$h(T_k) = 1 + \max(\underbrace{h(T_{k-1})}_{=k-1}, \underbrace{h(T_{k-2})}_{=k-2}) \quad (!!)$

$h(T_k) = 1 + (k-1) = k$

De plus $T_k =$  avec:

- $|h(T_{k-1}) - h(T_{k-2})| \leq 1$ ($= 1$)

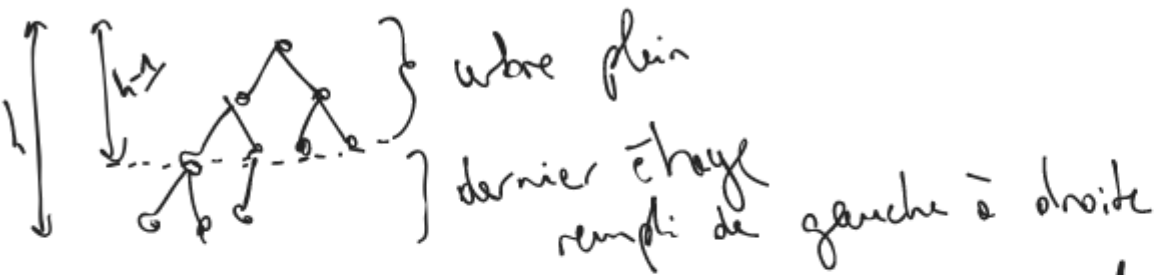
- T_{k-1} et T_{k-2} sont H-équilibrés par H.I

Donc T_k est H-équilibré.

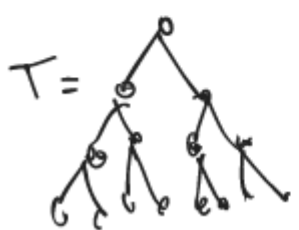
Conclusion : $\forall k \quad |h(T_k)| = k$
 T_k est H-équilibré

Exercice 3

Arbre parfait: Arbre binaire dans lequel tous les niveaux sont pleins sauf éventuellement le dernier qui est rempli de gauche à droite



(Q1) Soit h la hauteur d'un arbre parfait
 Alors $2^{h-1} \leq n < 2^h$




Arbre plein T : lien entre $n(T)$ et $h(T)$?

$$n(T) = \sum_{l=0}^{h(T)-1} 2^l = \frac{1-2^{h(T)}}{1-2} = \frac{2^{h(T)} - 1}{1-2}$$

↳ nombre d'un arbre plein.

Arbre parfait :



$$n(T) = 2^{h-2} - 1 + \underbrace{n_{\text{dernier étage}}}$$

Or.

$$1 \leq n_{\text{dernier étage}} \leq 2^{h-1}$$

$$\text{Donc } 2^{h-2} - 1 + 1 \leq n(T) \leq 2^{h-2} - 1 + 2^{h-1}$$

$\swarrow \quad \searrow$
 2^{h-1}

$$\text{Donc } 2^{h-2} \leq n(T) \leq 2^h - 1$$

$$\text{Donc } 2^{h-2} \leq n(T) \leq 2^h$$

Q2 $2^{h-2} \leq n < 2^h$ (*)

\Rightarrow isoler h ?

Comment passer de 2^h à h ?

\rightarrow (log)

Si on applique \log à (*):

$$\left(h-1 \leq \log_2 n < h \right) \quad \left| \begin{array}{l} h \leq \log_2 n + 1 \\ h > \log_2 n \end{array} \right.$$

\hookrightarrow donc $h-1 = \lfloor \log_2 n \rfloor$.

$$\text{Donc } h = \lfloor \log_2 n \rfloor + 1.$$

$$\log_2(10000) \approx 13,28$$

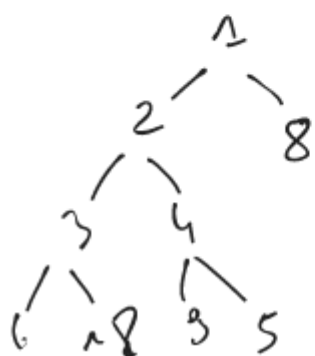
Donc $h = 14$

(pour 10000 nœuds, la hauteur n'est que de 14.)

Exercice 4

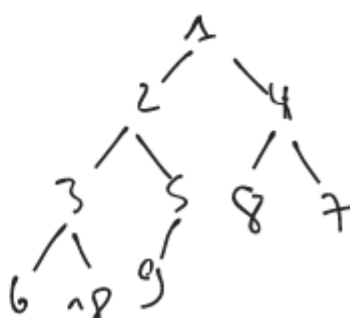
Q1

T_1



pas parfait
donc pas un b-s

T_2



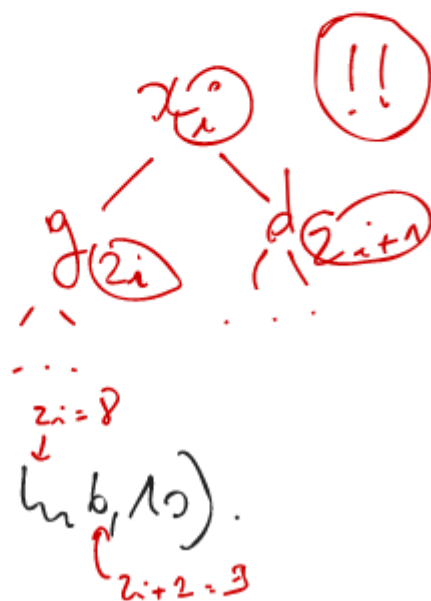
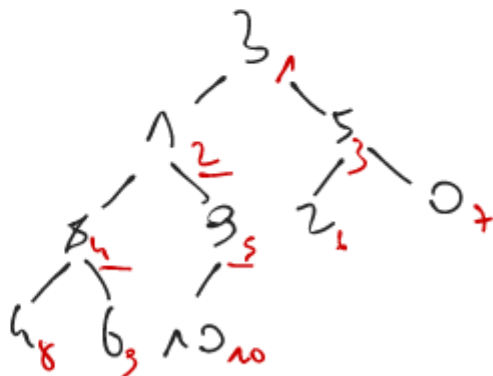
parfait
tous les chemins
racine \rightarrow feuille
sont croissants
Donc c'est un b-s

T_3



parfait
le chemin
racine \rightarrow feuille
1, 5, 2
n'est pas croissant
Donc ce n'est pas un b-s

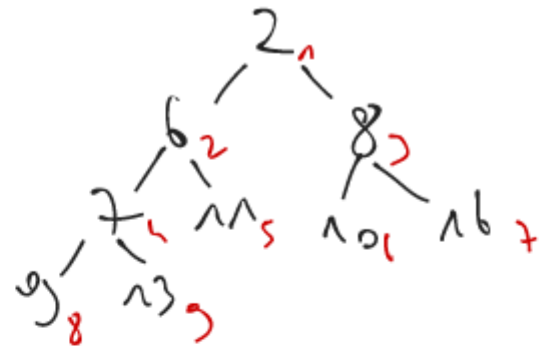
Exemple parcours par niveau:



$$A = [10, 3, 1, 5, 8, 9, 2, 0, 18, 10, 10, 10]$$

22

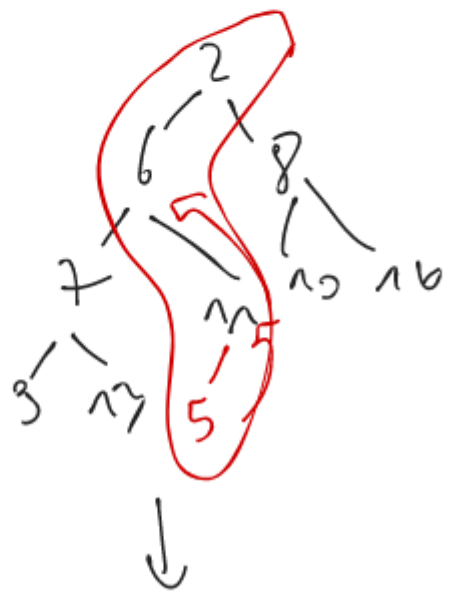
ExT



$A = [9, 2, 6, 8, 7, 11, 10, 16, 9, 13]$

1)

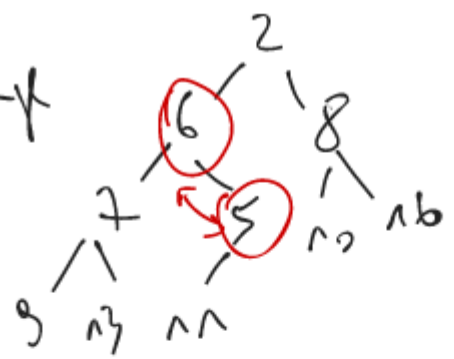
insertion



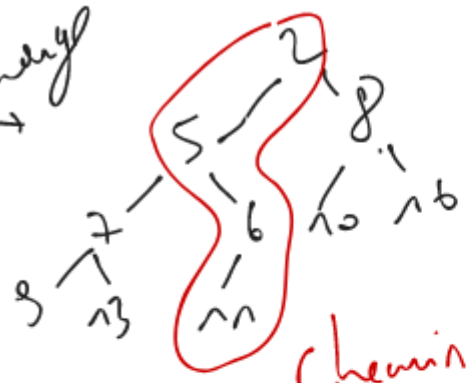
• On insère la valeur v à la fin du tableau (= à la fin de l'arbre parfait)

• On la fait remonter par la suite à la bonne place dans son chemin racine \rightarrow feuille

échange



échange



chemin croissant \checkmark

22

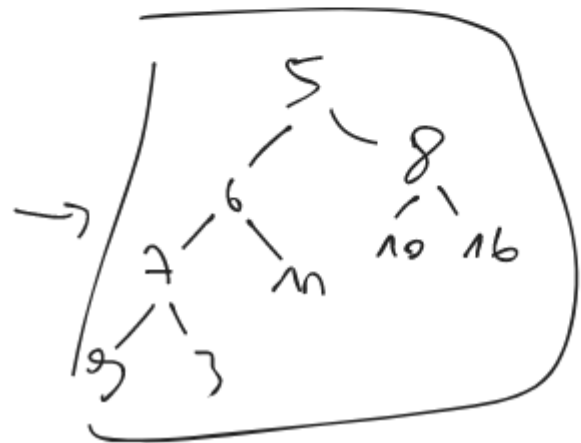


• On supprime la racine :
on échange $A[1]$ et $A[n]$
et on diminue la taille de 1

• On fait descendre la nouvelle racine à sa place



on fait descendre la racine à sa place



Q3

1). On construit le b+ en insérant les éléments de la liste 1 par 1

• On "détruit" le b+ par suppression successive des minimums

2) $L_0 = (5, 2, 3, 1, 4)$

(a) $5 \xrightarrow{2} 5 \rightarrow 2$

$3 \rightarrow 5 \rightarrow 2$

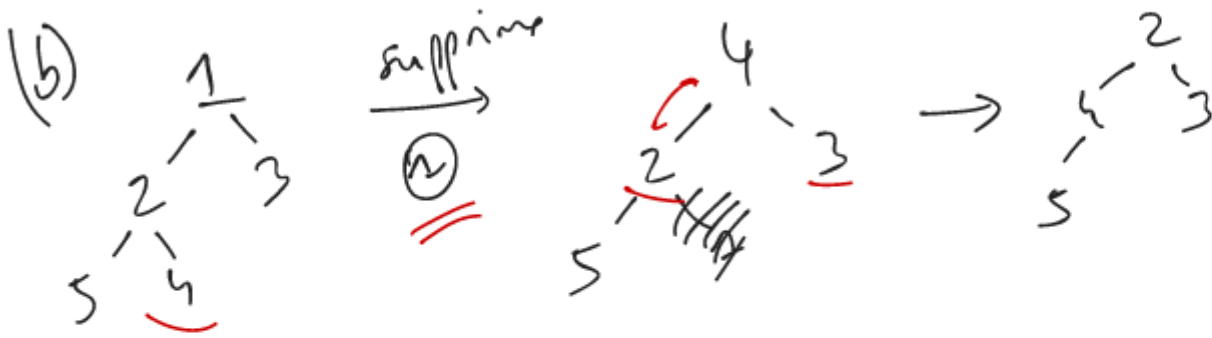
$1 \rightarrow 5 \rightarrow 2$

$2 \rightarrow 5 \rightarrow 2$

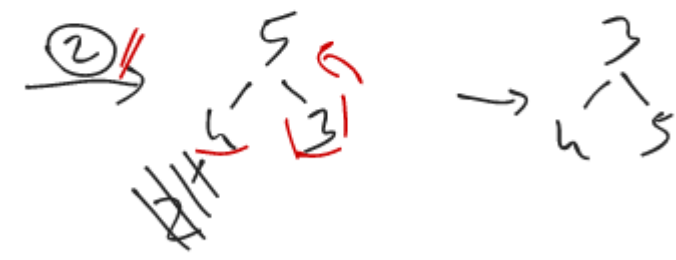
$4 \rightarrow 5 \rightarrow 2$



$$A_0 = [5, 1, 2, 3, 5, 4]$$



les suppressions
renvoient les
éléments par
ordre croissant.



Δ insertion / Δ suppression: $\mathcal{O}(\log_2 n)$
 (d'après la hauteur d'un
arbre parfait)

Bilan: n insertions: complexité $\mathcal{O}(n \log_2 n)$
 n suppressions: complexité $\mathcal{O}(n \log_2 n)$

\rightarrow tri par tas en $\mathcal{O}(n \log_2 n)$

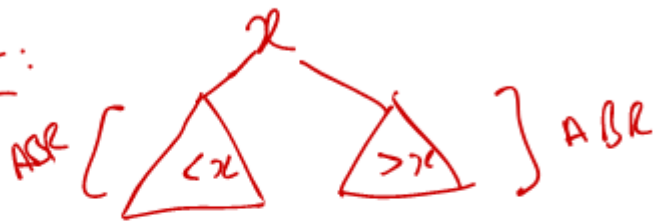
ABR rappel : \emptyset ou



avec
 $\forall g$ nœud de G
 $g < x$

$\forall d$ nœud de D
 $d > x$

ABR:



et G et D ABR

Exercice 6

Q1

B_1



pas un ABR

B_2



ABR

B_3



pas un ABR

Q2

Réduction

Exercice 7

Rappel: Parcours infixe d'un ABR
 = nœuds triés par ordre croissant

Q1 (3, 1, 4, 8, 5, 2, 7, 6)

$I = (1, 2, 3, 4, 5, 6, 7, 8)$
 (d'après le rappel)



2) Avec la même form, si on échange, on change le parcours infixe.

Or, 2 ABR sur les mêmes nœuds ont un même parcours infixe.

Donc ils ne peuvent pas avoir la même forme avec des nœuds placés différemment.

Q2 $P(T)$: I_T est trié par ordre croissant
abs T est un ABR.
parcours infixe de T

Ma $P(T)$ par récurrence forte sur la taille de T .

. Base : $T = \emptyset$, et $I_T = ()$ est trié dans tous les cas et T est un ABR.

. Induction : Soit $T = \begin{array}{c} \lambda \\ \swarrow \quad \searrow \\ G \quad D \end{array}$, supposons que

$\forall T'$ avec $n(T') < n(T)$: $I_{T'}$ trié $\Rightarrow T'$ ABR.

Supposons que I_T trié par ordre croissant


$$\underline{I_T = [I_G, \lambda, I_D]}$$

\rightarrow Donc I_G et I_D sont triés.

donc par H.I., G et D sont des ABR

$\rightarrow I_T$ est trié donc $\left(\begin{array}{l} \forall g \in I_G \quad g < \lambda \\ \forall d \in I_D \quad d > \lambda \end{array} \right)$

Donc T est un ABR (par définition)

 $P(T)$ est vrai pour tout arbre T .

Q3 1) - On veut le parcours infixe de l'arbre
- On teste si la liste du parcours est triée ou non

2) Parcours infixe, complexité en $\Theta(n)$ - $I_T = T_{\text{cjp}} \times I_D$
Test de tri d'une liste :
// caractérisation en $\Theta(1)$
// les listes chaînées

$O(n)$ (pire cas)
 $\Omega(n)$ (meilleur cas)

Donc la succession de ces deux étapes se fait en $\Theta(n)$.

Q4 Soit P un parcours préfixe

Soit T_1 et T_2 deux ABL avec
pour parcours préfixe P .

Rappel : 2 arbres qui ont le même parcours préfixe
et le même parcours infixe sont égaux
(cf TD6)

T_1 et T_2 ont le même parcours préfixe

Donc ils ont les mêmes nœuds

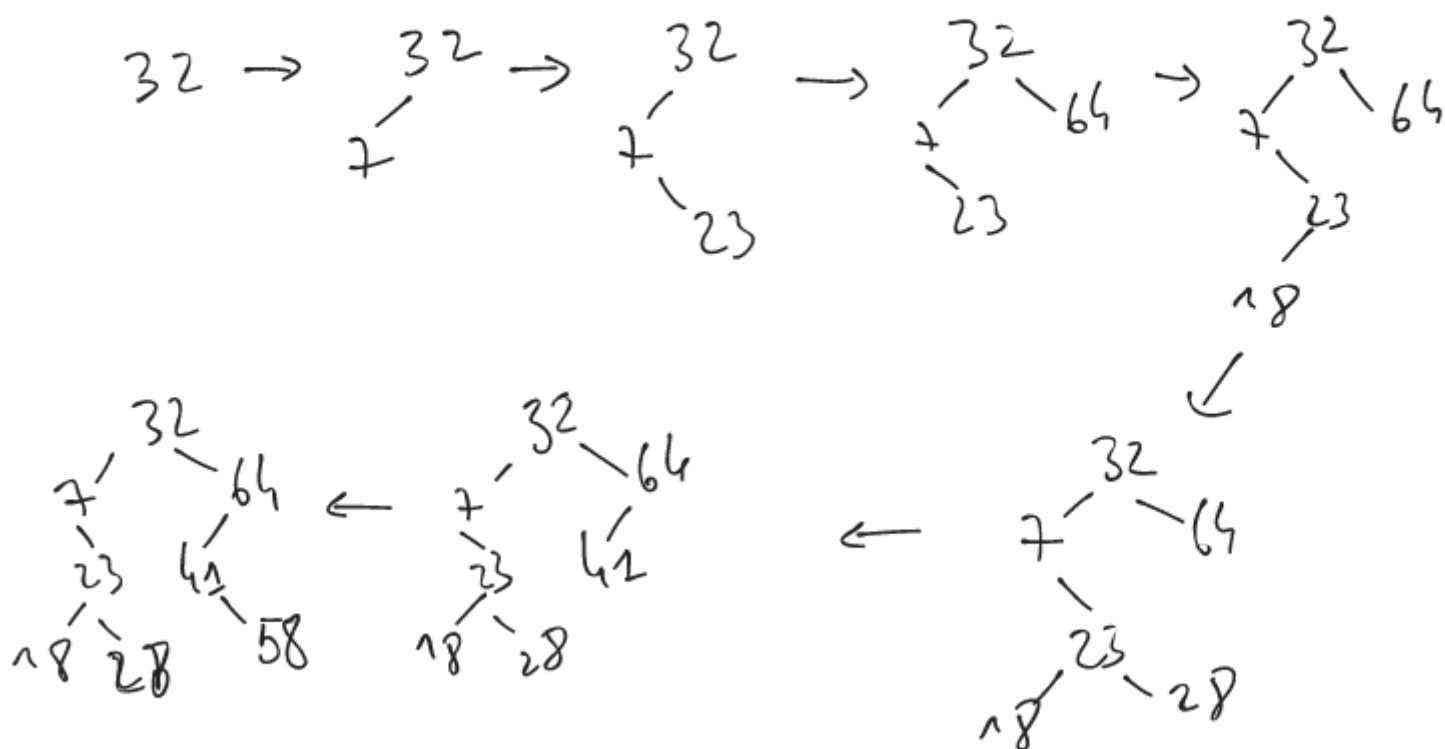
Donc ils ont le même parcours infixe (caractérisation d'un ABL)

Donc $T_1 = T_2$.

d'où l'unicité.

Exercice 9

Q1 32, 7, 23, 64, 18, 28, 41, 58



Un ABR est unique à ordre d'insertion fixé.

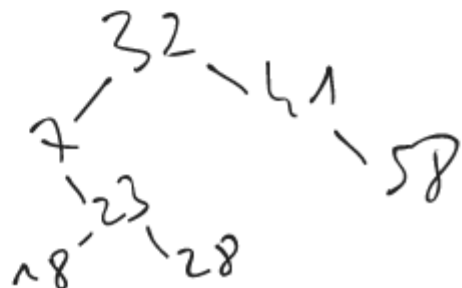
Si on change l'ordre d'insertion, on peut changer l'ABR
(par exemple si on change le 1^{er} élément)

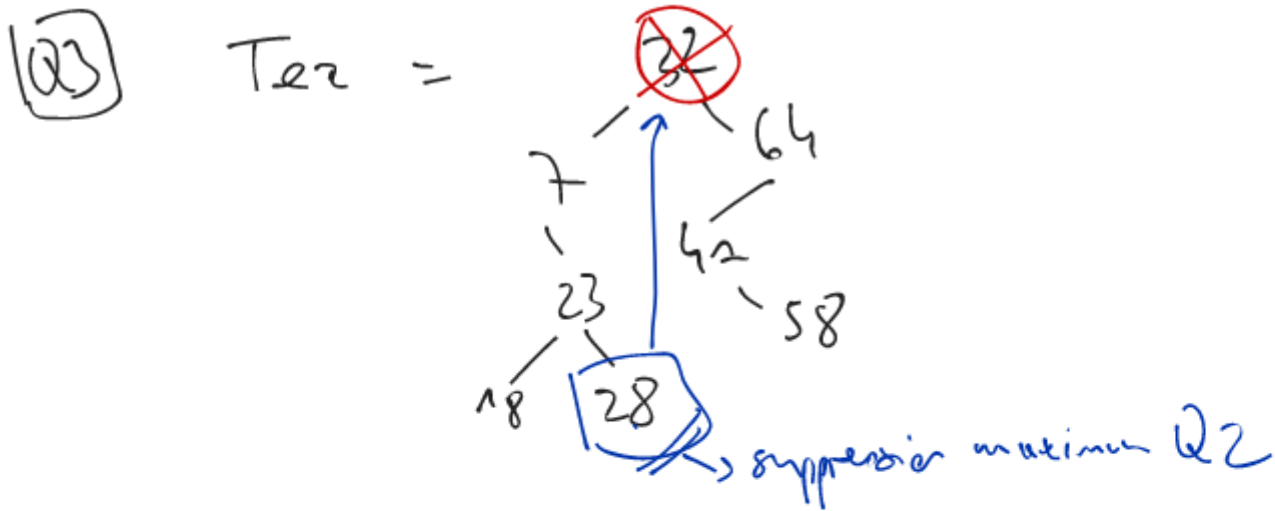


→ On supprime le nœud maximum

→ On fait remonter le fil gauche à sa place

On obtient :

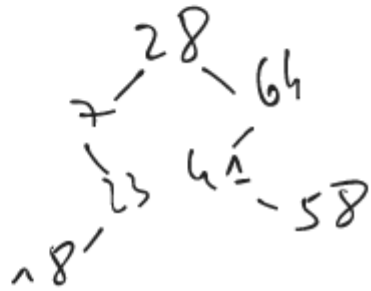




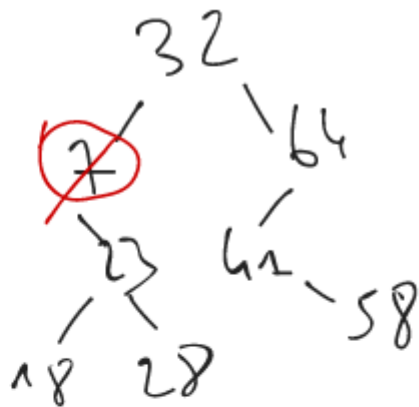
• On récupère le maximum du sous-arbre gauche en le supprimant

• On remplace la racine par ce maximum

On obtient :



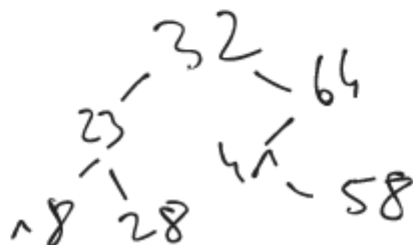
Q4) Taz



Comme pour la suppression du maximum, on fait remonter le fils gauche.

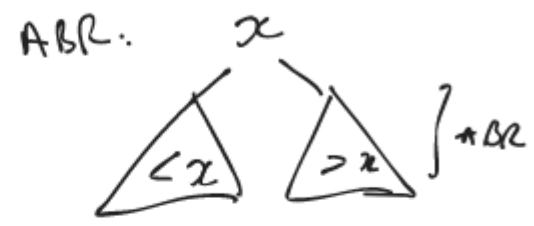
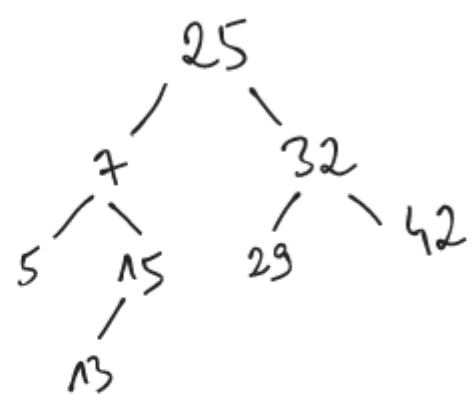
7 n'a pas de fils gauche, donc on fait remonter son fils droit

On obtient :



TD 7 exercice 10

Q1

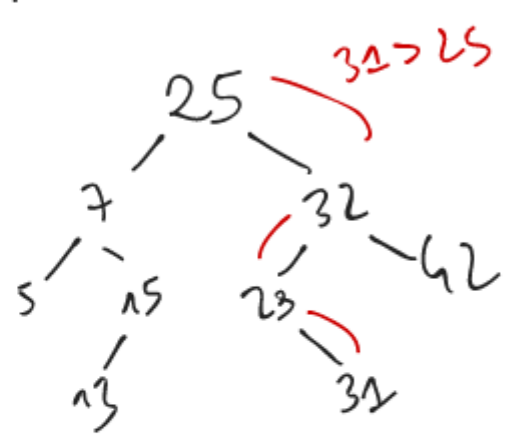


$I = [5, 7, 13, 15, 25, 29, 32, 42]$

Rappel : T est un ABR
 \Rightarrow son parcours infixe est trié par ordre croissant à retenir

Méthode : Pour mq T est ABR, on peut calculer son parcours infixe (souvent c'est très facile !)

Ajout de 31 :



$I = [5, 7, 13, 15, 25, 29, 31, 32, 42]$

Q2

$P(T)$ = Pour tout x , $ABR_{insertion}(x, T)$ se termine et renvoie un ABR dont les clés sont x et celles de T

$\forall T$ $P(T)$ est vraie par induction structurale

$T: \emptyset$ ou 


Base: $ABR_{insertion}(x, \emptyset)$ termine et renvoie la feuille x d'après le code, c'est un ABR avec les clés de \emptyset (aucune) et x , c'est juste x .

Induction: Soit $T = \begin{matrix} & r & \\ / & & \backslash \\ G & & D \end{matrix}$, on suppose $P(G)$ et $P(D)$

$\forall T$ $P(T)$

$ABR_{insertion}(x, T)$:

Si $x = r$: la fonction renvoie T qui est un ABR
 x dont les clés sont x et les clés de T (x est inséré dans les clés de T)

Si $x < r$: Si on note $G' = ABR_{insertion}(x, G)$
 Alors la fonction renvoie  $= T'$

Par H.R., G' est un ABR dont les clés sont x et les clés de G (**)

(**): les clés de T' sont x et les clés de T

(*) : $x < r$, donc G' et D sont des ABR tq $\forall g \in G' \quad g < r$
 $\forall d \in D \quad d > r$

Donc T' est un ABR

méthode induction structurale

• si $n \geq r$, on applique le même raisonnement
avec $D' = \text{Abstraction}(a, T, \text{In:V})$

Conclusion : Pour tout ABR T , $P(T)$ est vraie.