ARBRES BINAIRES DE RECHERCHE (ABR)

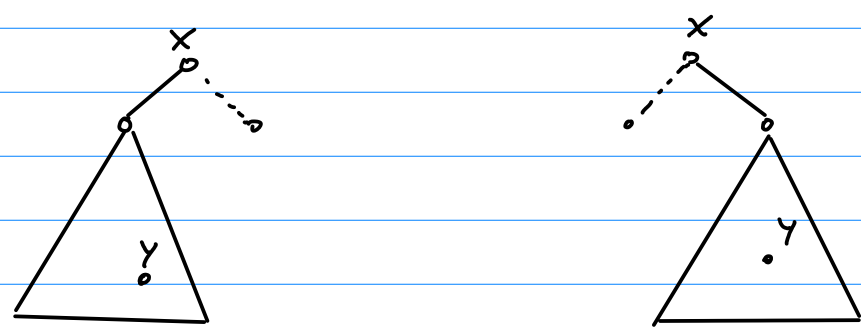
arbre binaire :

gauche, droite

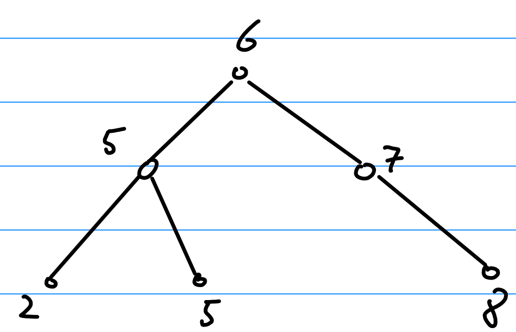
- •) chaque noeud a ≤ 2 fils, ≤ 1 parent et une valeur
- •) la racine est le seul noeud sans parent

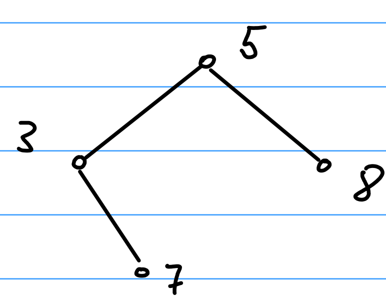
ABR : Un ABR est un arbre binaire T qui satisfait la propriété suivante :

Soient x, y deux noeuds de T.
Si y se trouve dans le sous-arbre enraciné en x.gauche (resp. x.droite) alors x.valeur $\geq y$.valeur (resp. x.valeur $\leq y$.valeur)



Exemple :





NON

racine ↓ valeur ↓

```
ABR-RECHERCHE(x, v)
    if x == None or x.valeur == v :
        return x
    if v ≤ x.valeur :
        return ABR-RECHERCHE(x.gauche, v)
    else :
        return ABR-RECHERCHE(x.droite, v)
```

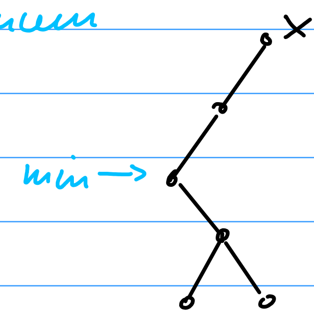
$O(h)$

(h = hauteur)

```
ABR-MINIMUM(x) :
if x == None :
    return None
if x.gauche == None :
    return x
else :
    return ABR-MINIMUM(x.gauche)
```

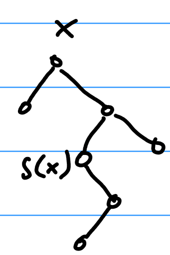
droite au lieu de gauche → maximum

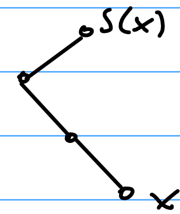
$O(h)$

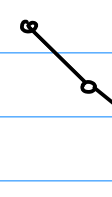



```
ABR-SUCCESSOR(x) :
    if x.droite ≠ None :
        return ABR-MINIMUM(x.droite)
    y = x.parent
    while y ≠ None and x = y.droite
        x = y
        y = y.parent
    return y
```

$O(h)$







pas de successeur

ABR-PARCOURS-INFIXE(x) $O(n)$

```
if x ≠ None :
    ABR-PARCOURS-INFIXE(x.gauche)    préfixe
    yield x (.valeur)
    ABR-PARCOURS-INFIXE(x.droite)    postfixe
```

→ iterateur sur les valeurs triées par ≤

borne inf tri

⇒ l'insertion ne peut pas être effectuée dans un temps constant

ABR-INSERTION(T, v) : $O(h)$

```
y = None
x = T.root
while x ≠ None :
    y = x
    x = x.gauche if v < x.valeur else x.droite

if y == None :
    T.racine = Noeud(v)
elif v < y.valeur :
    y.gauche = Noeud(v, parent = y)
else :
    y.droite = Noeud(v, parent = y)
```

trouver parent y du nouveau noeud

T était vide, pas de parent

La hauteur d'un ABR à valeurs $\{1, 2, \dots, n\}$ est $\geq \log_2 n$ et $\leq n$

Soit X_n la variable aléatoire egale à la hauteur d'un ABR à valeurs $\{1, 2, \dots, n\}$ construit aléatoirement

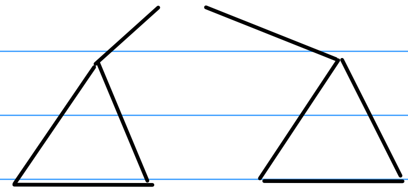
Théorème : $E[X_n] = O(\log_2 n)$

Démo : variables aléatoires

hauteur exponentielle $Y_n = 2^{X_n}$

racine $R_n = i$ choisie aléatoirement $\in \{1, 2, \dots, n\}$: $P[R_i = i] = \frac{1}{n}$

$R_n = i$



ABR aléatoire T_1 à $i-1$ valeurs

ABR aléatoire T_2 à $n-i$ valeurs

$$h(T) = 1 + \max\{h(T_1), h(T_2)\}$$

$$\Rightarrow Y_n = 2 \cdot \max\{Y_{i-1}, Y_{n-i}\}$$

$n = 1$: $Y_1 = 2^0 = 1$ (on défini $Y_0 = 0$)

$$Z_{n,i} = \begin{cases} 1 & \text{si } R_n = i \\ 0 & \text{sinon} \end{cases}, \quad E[Z_{n,i}] = \frac{1}{n} \quad (1 \leq i \leq n)$$

$$Y_{n,i} = \sum_{i=1}^{n} Z_{n,i} \cdot 2 \cdot \max\{Y_{i-1}, Y_{n-i}\}$$

On va montrer que $E[Y_n] = O(n^3)$ ce qui implique $E[X_n] = O(\log_2 n)$

On remarque que $Z_{n,i}$ et Y_{i-1}, Y_{n-i} are indépendantes

$$E[Y_n] = E\left[\sum_{i=1}^{n} Z_{n,i}\,(2\cdot \max\{Y_{i-1}, Y_{n-i}\})\right]$$

$$= \sum_{i=1}^{n} E\left[Z_{n,i}\,(2\cdot \max\{Y_{i-1}, Y_{n-i}\})\right]$$

$$\overset{\text{indépendence}}{=} \sum_{i=1}^{n} E[Z_{n,i}]\cdot E\left[2\cdot \max\{Y_{i-1}, Y_{n-i}\}\right]$$

$$= \frac{2}{n} \sum_{i=1}^{n} E\left[\max\{Y_{i-1}, Y_{n-i}\}\right]$$

à vérifier $\rightarrow$ $\leq \frac{2}{n} \sum_{i=1}^{n} \left(E[Y_{i-1}] + E[Y_{n-i}]\right)$

$$= \frac{4}{n} \sum_{i=0}^{n-1} E[Y_i]$$

Exercice :

(a) montrer que $E[Y_n] \leq \frac{1}{4}\binom{n+3}{3}$ en utilisant

$$\sum_{i=0}^{n-1} \binom{i+3}{3} = \binom{n+3}{4}$$

(b) Conclure que $E[X_n] = O(\log_2 n)$

$\square$

(a) cas initial:

$$0 = Y_0 = E[Y_0] \leq \frac{1}{4}\binom{3}{3} = \frac{1}{4}$$
$$1 = Y_1 = E[Y_1] \leq \frac{1}{4}\binom{4}{3} = 1$$

$$E[Y_n] \leq \frac{4}{n} \sum_{i=0}^{n-1} E[Y_i] \leq \frac{1}{n} \sum_{i=0}^{n-1} \binom{n+3}{3}$$

$$= \frac{1}{n}\binom{n+3}{4} = \frac{1}{n} \frac{(n+3)!}{4!\,(n-1)!}$$

$$= \frac{1}{4} \frac{(n+3)!}{3!\ n!} = \frac{1}{4}\binom{n+3}{3}$$

$f = 2^x$ convexe

(b) $$2^{E[X_n]} \leq E[2^{X_n}] = E[2^{Y_n}]$$

$$2^{E[X_n]} \leq \frac{1}{4}\binom{n+3}{3}$$

$$\leq c n^3 \quad \text{(pour tout } n \geq n_0 \text{, } n_0 \text{ bien choisi)}$$

$$\Rightarrow E[X_n] \leq c \cdot 3 \cdot \log_2 n = O(\log_2 n)$$