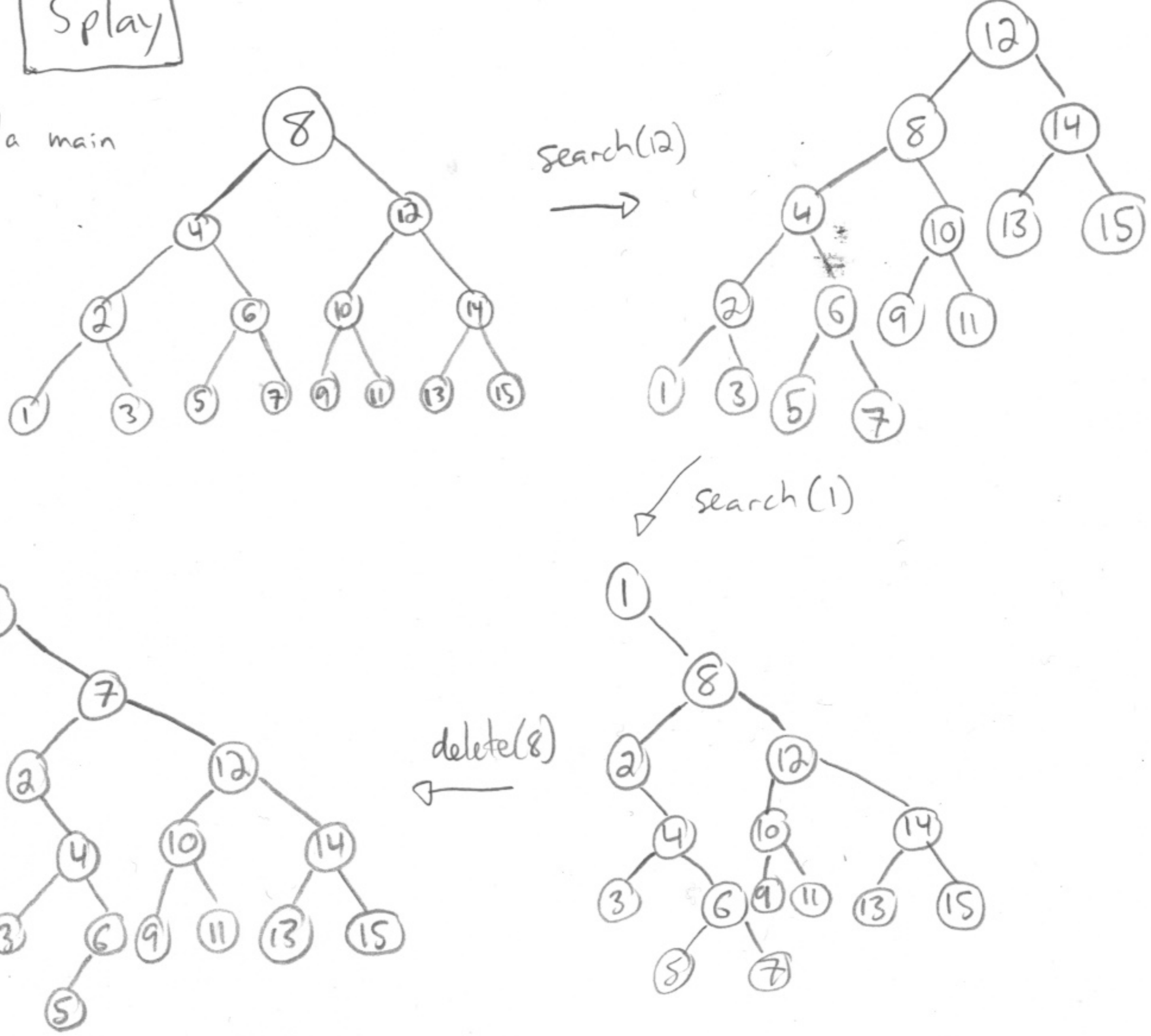
verification RN

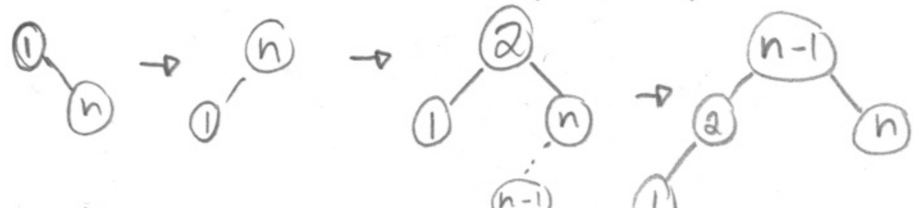
```
coloriageRN(root):
    if root.col == red:
    | return false
    result = coloriageRN(root, black)
    if result == false:
    | return false
    return true

coloriageRN(N, col_parent):
    if N == null:
    | return 0
    if N.col == red:
    |   if (N.left == null || n.right == null): # requis pour l'exercice mais
    |                                           # pas selon définition du cours
    |   | return false
    |   if col_parent == red:
    |   | return false
    left = coloriageRN(N.left, N.col)
    if left == false:
    | return false
    right = coloriageRN(N.right, N.col)
    if right == false || left != right :
    | return false
    if N.col == black:
    | return 1 + left
    return left
```

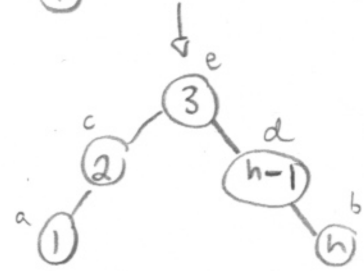
Splay

- à la main



- séquence alternante : voici le résultat des quelques premières insertions:





Lemme: l'insertion du $i^{ème}$ élément se fait au niveau $n \leq 2$ et implique au maximum 2 rotations.

Cas de base: $i = 1$. Une seule opération: insertion de la racine.

- Séquence alternante (suite)

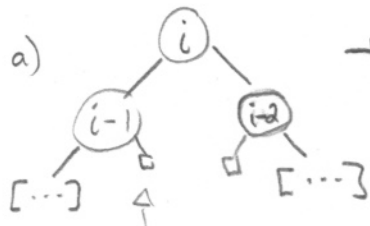
H.I. Le $i^{\text{ème}}$ élément s'insère au niveau $n \leq 2$ et son déploiement comprend au maximum 2 rotations.

Cas inductif $(i+1)$:

2 cas possibles pour cette séquence d'insertions :

* $S(i)$ dénote la valeur de la séquence à l'indice i.

a)

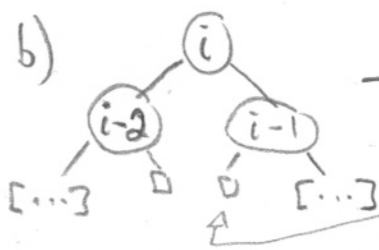


$\rightarrow S(i-1) < S(i)$

Dans ce cas, $S(i+1) < S(i)$ et $S(i+1) > S(i-1)$.

Donc, l'élément $i+1$ est inséré comme enfant droit du noeud $i-1$ (niveau 2) et on doit faire deux rotations (zig-zag) pour le déployer à la racine.

b)



$\rightarrow S(i-1) > S(i)$

Dans ce cas, $S(i+1) > S(i)$ et $S(i+1) < S(i-1)$

$i+1$ est inséré comme enfant gauche du noeud $i-1$ (niveau 2) et on doit faire deux rotations (zag-zig) pour le déployer à la racine. $\square$

Comme on a un nombre d'opérations borné par une constante pour cette séquence d'insertion, il va de soi que l'insertion est $O(\log n)$. Aussi, on voit que l'arbre dégénère en une liste dont le milieu est la racine, d'où la hauteur $\Theta(i)$, ou $\sim \frac{1}{2} n$

Ougandais

b) On définit la hauteur noire (ou rang) des arbres ougandais comme celle des rouge-noir.

On peut la borner ainsi pour le sous-arbre enraciné au noeud x: $h(x) \leq 3 \cdot \text{rang}(x)$

Preuve : En pire cas, on a une alternance (noir-jaune-rouge)..

Pour le nombre de noeuds, on peut le borner comme pour les rouge-noir (tous les arbres rouge-noir sont des arbres ougandais et les arbres ougandais ne font qu'ajouter des noeuds pour un même rang):

$$n(x) \geq 2^{\text{rang}(x)} - 1$$

Donc,

$$n(x) + 1 \geq 2^{\text{rang}(x)}$$

$$\log(n+1) \geq \text{rang}(x)$$

$$\underbrace{\log(n+1)}_{\text{hauteur d'un arbre binaire complet}} \leq h(x) \leq 3\log(n+1) \quad \square$$

$\boxed{\text{Gamme}}$ (dans brebis.pdf)

a)
```
gamme(n):
  if n.parent == null:
  | return (-∞, ∞)
  if n.key < n.parent.key :
  | return (gamme(n.parent)[0], n.parent.key)
  return (n.parent.key, gamme(n.parent)[1])
```

b)
```
affichageGamme(n): # on appelle sur la racine
  affichageGamme(n, (-∞, ∞))

affichageGamme(n, g):
  if n == null:
  | return
  print(g)
  affichageGamme(n.left, (g[0], n.key))
  affichageGamme(n.right, (n.key, g[1]))
```