(903)

D.1.

DEVELOPPEMENT

ANALYSE DE COMPLEXITE DU TRI RAPIDE.

Le tri rapide trie en $O(n^2)$ dans le pire cas mais en $O(n \log n)$ en complexité espérée.

- analyse pire cas
- analyse complexité espérée

- Pire cas : Le pivot est toujours le plus petit (ou le plus grand) élt.

$$T(n) = T(n-1) + \underbrace{T(0)}_{=0} + \Theta(n)$$

$$T(n) = \Theta(n^2).$$

- Complexité espérée :

Rq : nombre de comparaisons effectuées = nombre de comparaisons effectuées dans partition.

→ On va essayer de compter le nombre total de comparaison effectuées pdt les partitionnements

Hypothèse et notation : $A = \pi(\beta_1, \dots, \beta_n)$ où $\begin{cases} \beta_1 < \beta_2 < \dots < \beta_n \\ \pi \text{ permutation.} \end{cases}$

On note $Z_{ij} = \{\beta_i, \beta_{i+1}, \dots \beta_j\}$ pour $i < j$.

Variables aléatoires : $X_{ij} = \begin{cases} 1 & \text{si } \beta_i \text{ est comparé avec } \beta_j \\ 0 & \text{sinon} \end{cases}$ pour $i<j$

X est le nombre de comparaisons effectuées par l'algorithme. $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$.

Calcul de la complexité espérée :

$$E(X) = E\left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{ij}).$$

$E(X_{ij})$ est la probabilité que β_i soit comparé à β_j.

On peut remarquer que une fois qu'un pivot de Z_{ij} a été choisi, z_i se retrouvera dans une partition différente de z_j et ils ne seront donc pas comparés (sauf si l'un des deux est le pivot). De plus, avant que un pivot soit choisi dans Z_{ij}, z_i et z_j n'ont été comparés qu'avec les pivots précédemment choisis et donc pas entre eux.

Autrement dit :

$$\mathbb{P}\{z_i \text{ est comparé avec } z_j\}$$

ds

$$= \mathbb{P}\{z_i \text{ ou } z_j \text{ soit premier pivot dans } Z_{ij}\}$$

indépendance

$$= \mathbb{P}\{z_i \text{ soit 1er pivot dans } Z_{ij}\} + \mathbb{P}\{z_j \text{ soit premier pivot dans } Z_{ij}\}$$

$$= \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}$$

Soit $$E(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1}$$

$$\leq \sum_{i=1}^{n-1} \sum_{k=1}^{n-1} \frac{2}{k}$$

$$E(X) = O(n \log n).$$

Référence : Cormen, chap 7.4.

903

D.2

DEVELOPPEMENT

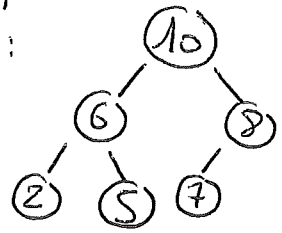
TRI PAR TAS (Heapsort)

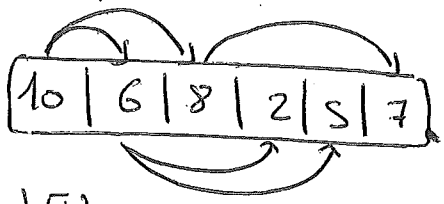
On peut, en utilisant une structure de tas, implémenter un tri qui tourne en $O(n \log n)$ comparaisons dans le pire cas.

- écriture de l'algorithme et des sous-routines
- le faire tourner sur un exemple
- analyser la complexité

• Rappel : un tas (max) = arbre binaire complet tq tout noeud soit $\geq$ à ses fils

exple :





parent[i] = $\lfloor i/2 \rfloor$, gauche[i] = 2i, droite[i] = 2i+1

• Algorithme :

Complexité

* Tri_tas (A) :=
$O(n \log n)$

```
Construit_tas (A)
Pour i = A.length à 2
    Echanger (A[1], A[i])
    A.tas.size = A.tas.size - 1
    Tassifier (A, 1)
```

$O(n \log n)$ + $n \times O(\log n)$

* Tassifier (A, i) :=

$T(n) \leq T(2n/3) + O(1)$

$\downarrow$ (master theorem)

$T(n) = O(\log n)$

```
l = gauche[i], r = droite[i], max = i
Si l ≤ A.tas.size & A[l] > A[i]
    max = l
Si r ≤ A.tas.size & A[r] > A[max]
    max = r
Si max ≠ i
    Echanger (A[i], A[max])
    Tassifier (A, max)
```

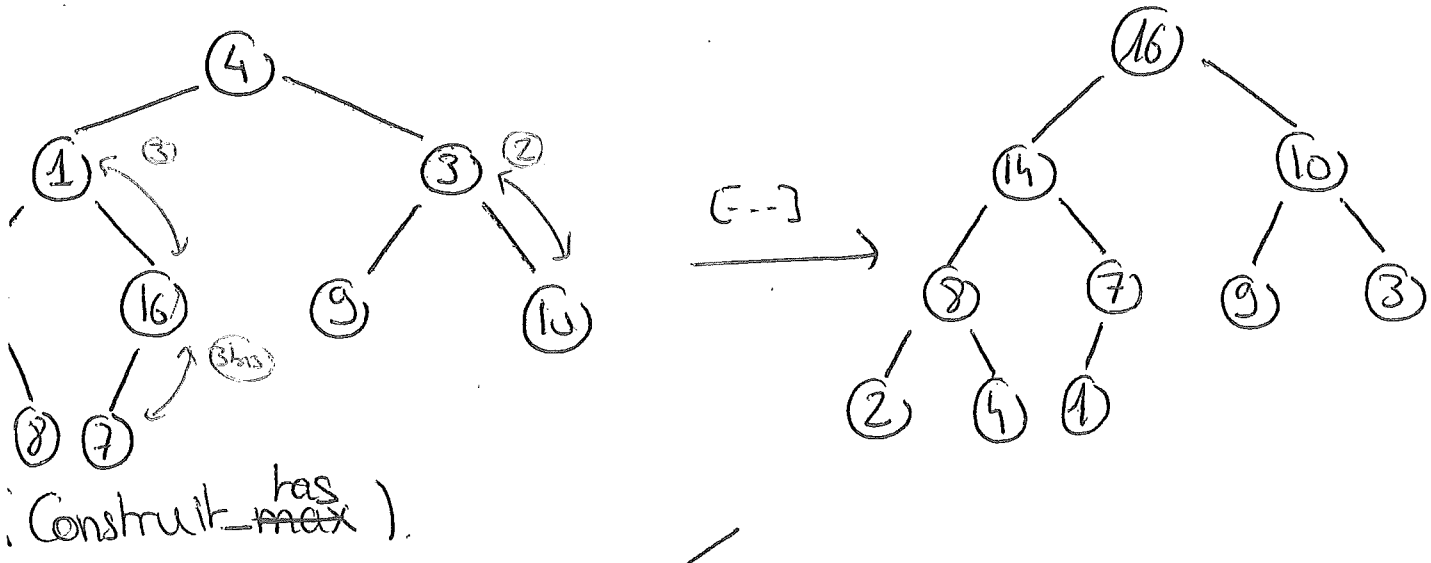
* Construit_tas :=

$T(n) \leq n \times T_{Tassifier}(n)$

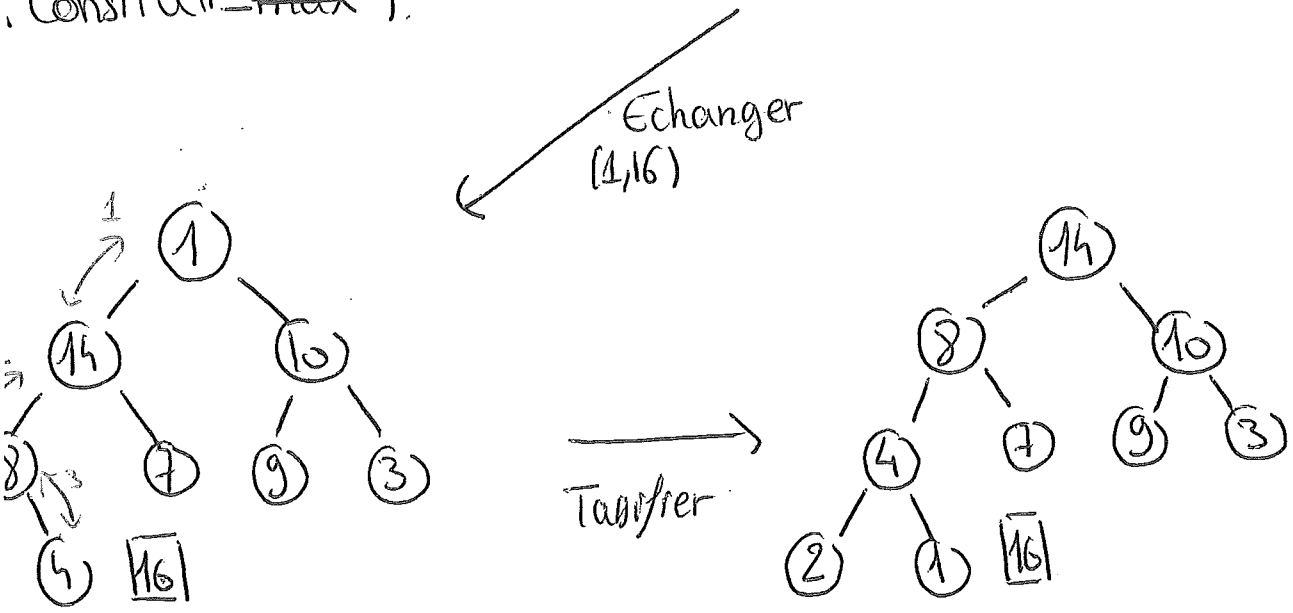
$\leq O(n \log n)$

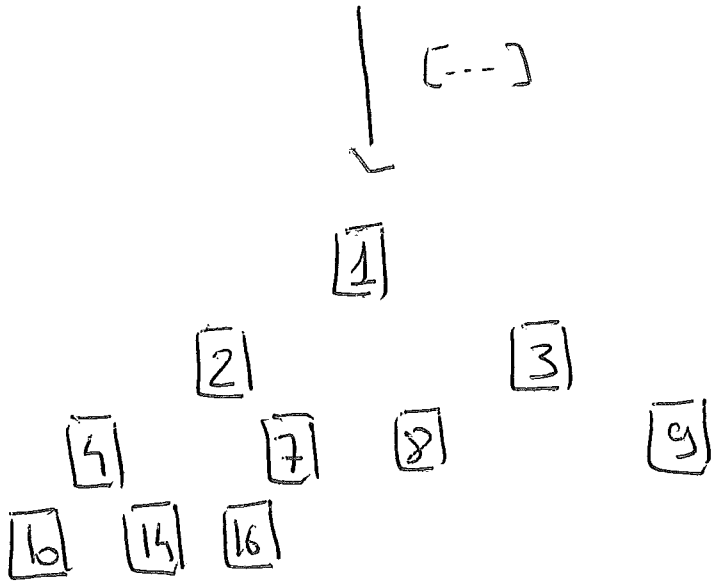
```
A.tas.size = A.length
Pour i = ⌊A.length/2⌋ à 1
    Tassifier (A, i).
```

• Exemple : A = 4-1-3-2-16-9-10-14-8-7



(Construit_~~max~~ has)





A = 1-2-3-4-7-8-9-10-14-16

Référence : Cormen, chap6.

(903)
D.3

DEVELOPPEMENT

BORNE INFERIEURE DU TRI PAR COMPARAISON.

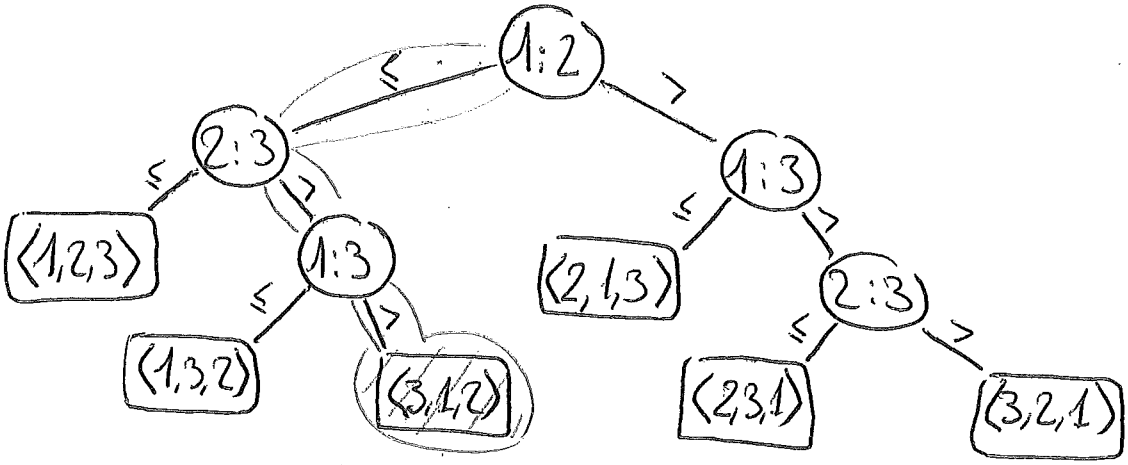
La complexité dans le pire cas d'un algorithme de tri par comparai est au moins $\Omega(n \log n)$.

- arbres de décision
- modélisation d'un tri par comparaison avec un arbre de décision, exemple
- calcul de la borne inférieure

- Un arbre de décision (ici) : arbre (binaire) entier tel que :
  * les noeuds sont étiquetés par des tests
  * les arrêtes sont étiquetés par le résultat du test du noeud au dessus
  * les feuilles sont étiquettes par les décisions.

- Un algorithme de tri par comparaison peut être vu comme un arbre de décision où :
  * les noeuds représentent les comparaisons successives.
  * les arrêtes représentent le résultat des comparaisons
  * les feuilles représentent les permutations.

Exple : Tri par insertion sur 3 éléments :



Légende :

a:b = on compare A[a] et A[b].

≤ = le résultat est a≤

> = le résultat est a>!

⟨x,y,z⟩ = la permutat le tableau trié est (A[x], A[y], A[z]).

- <u>Calcul de la borne inférieure :</u>

  * Le nombre de comparaisons effectuées par l'algorithme correspond à la longueur du chemin emprunté pour arriver à la bonne permutation.
  * Donc dans le pire cas = max {longueurs des chemins racine-feuille} = hauteur de l'arbre = h.
  * Comme toute permutation est a priori possible, le nombre de feuilles est égal au nombre de permutations = n!

    Un arbre binaire de hauteur h a au maximum 2^h feuilles, donc :

$$n! \leq 2^h$$

$$\Downarrow \leftarrow (\log \nearrow)$$

$$\underbrace{\log(n!)}_{\equiv\ \Omega(n \log n)} \leq h$$

(formule de Stirling) →

$$\left[n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right) \right]$$

<u>Référence</u> : Cormen, chap 8.1.