DIVISER POUR REGNER : TRI RAPIDE [CLRS, ch. 7]

objectif : trier un sous-tableau $A[p, \dots, r]$

DIVISER : Le tableau $A[p, \dots, r]$ est partitionné en sous-tableaux $A[p, \dots, q-1]$ et $A[q, \dots, r]$ tels que chaque élément de $A[p, \dots, q-1]$ soit $\leq A[q]$ et $A[q]$ est $\leq$ à chaque élément de $A[q+1, \dots, r]$

RÉGNER : trier $A[p, \dots, q-1]$ et $A[q+1, \dots, r]$ par des appels récursifs

COMBINER : $A[p, \dots, r]$ est trié ; aucune fusion n'est nécessaire

```
TRI-RAPIDE(A, p, r) :
    si p < r :
        q = PARTITION(A, p, r)
        TRI-RAPIDE(A, p, q-1)
        TRI-RAPIDE(A, q+1, r)
```

```
PARTITION(A, p, r) :
    pivot = A[r]
    i = p-1
    pour p ≤ j ≤ r-1 faire :
        si A[j] ≤ pivot :
            i += 1
            A[i], A[j] = A[j], A[i]
    A[i+1], A[r] = A[r], A[i+1]
    retourner i+1
```

$\Theta(1)$

$\Theta(n)$

au total : $\Theta(n)$

Exemple : PARTITION([1, 4, 8, 2, 7, 6, 3, 5], 0, 7)

pivot = 5   A[j]

j	A à la fin de l'étape
0	[1, 4, 8, 2, 7, 6, 3, 5]
1	[1, 4, 8, 2, 7, 6, 3, 5]
2	[1, 4, 8, 2, 7, 6, 3, 5]
3	[1, 4, 2, 8, 7, 6, 3, 5]
4	[1, 4, 2, 8, 7, 6, 3, 5]
5	[1, 4, 2, 8, 7, 6, 3, 5]
6	[1, 4, 2, 3, 7, 6, 8, 5]

dernière permutation : [1, 4, 2, 3, 5, 6, 8, 7]

Performances du tri rapide :

•) pire cas : TRI-RAPIDE([1, 2, ..., n], 0, n−1) coûte

$$T(n) = T(n-1) + T(0) + \Theta(n)$$
$$= T(n-1) + \Theta(n)$$
$$= \Theta(n^2) \quad \text{(pourquoi ?)}$$

•) meilleur cas : partitionnement le plus equilibré
→ deux sous-problèmes de taille $\leq \frac{n}{2}$

$$T(n) \leq \underset{a}{2}\, T\big(\tfrac{n}{\underset{b}{2}}\big) + \underset{f(n)}{\Theta(n)}$$

$n^{\log_b a} = n = \Theta(f(n))$ ⟹ cas 2) du master theorem

$\overset{MT}{\Longrightarrow} T(n) = \Theta(n \log n)$

TRI-RAPIDE-RAND : appeler PARTITION-RAND à la place de partition

PARTITION-RAND :

i = randint (p, r)

$A[r], A[i] = A[i], A[r]$

retourner PARTITION(A, p, r)

Théorème : Soit A un tableau de taille n contenant une permutation de $1, 2, \dots, n$. Alors l'espérance $C(n)$ du nombre de comparaisons effectuées lors de l'exécution de TRI-RAPIDE-RAND est au plus $2n \ln n$.

On va utiliser

(1) Soit X une variable aléatoire avec valeurs $x_1, \dots, x_n$ et probabilités $p_1, \dots, p_n$, on a

$$E[X] = \sum_{i=1}^{n} x_i \cdot p_i$$

(2) Pour deux variables aléatoires X, Y on a

$$E[X+Y] = E[X] + E[Y]$$

Démo :

Le nombre de comparaisons effectuées par PARTITION-RAND est égal à $n-1$.

Pour $0 \leq i \leq n-1$ on obtient deux sous-tableaux de tailles i et $n-i-1$ avec probabilité $\frac{1}{n}$ (selon le choix aléatoire de l'élément pivot)

(1), (2)

Donc $$C(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}\left(C(i) + C(n-i-1)\right)$$

$C(0)=0$

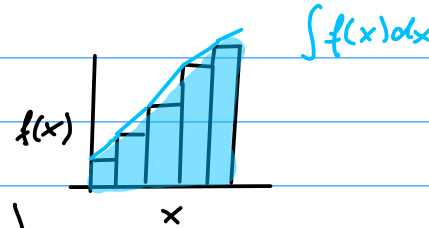
$$= (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} C(i)$$

On suppose que $C(i) \leq c\, i \ln i$ pour tout $1 \leq i \leq n-i$
et montre alors que $C(n) \leq c\, n \ln n$

$$C(n) \leq (n-1) + \frac{2}{n}\sum_{i=1}^{n-1}\left(c\, i \ln i\right)$$

f croissante

$$\leq (n-1) + \frac{2}{n}\int_1^n (cx \ln x)\, dx$$



$\int x \ln x = \frac{x^2}{2}\ln x - \frac{x^2}{4} + C$

$$= (n-1) + \frac{\cancel{2}c}{\cancel{n}}\left(\frac{n^{\cancel{2}}}{\cancel{2}}\ln n - \frac{n^2}{\cancel{4}} + \frac{1}{\cancel{4}}\right)$$

$$\leq (n-1) + c\, n \ln n - c\frac{n}{2} + \frac{c}{2n}$$

Pour $c=2$ on obtient $C(n) \leq 2 n \ln n$.

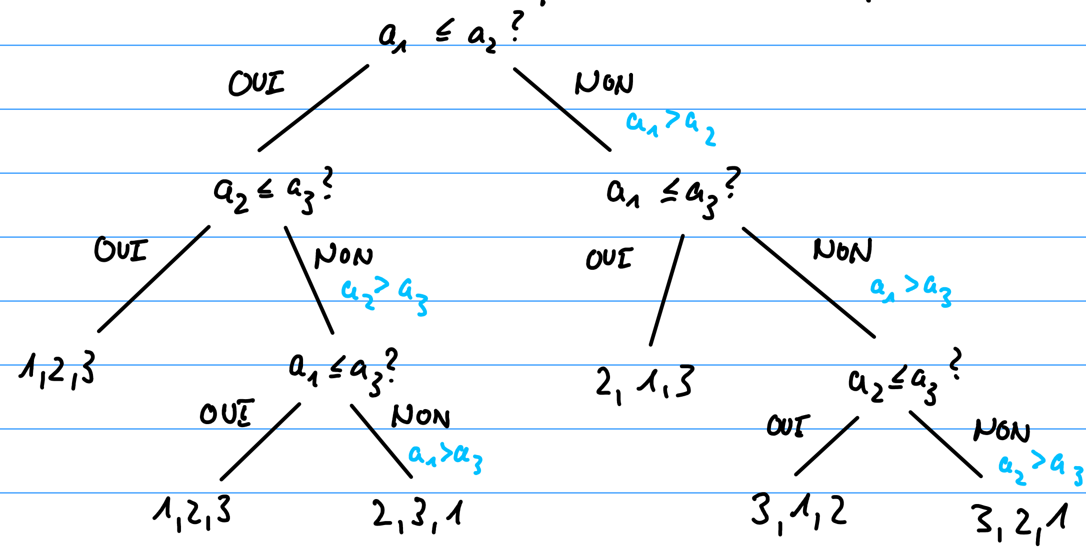
Il reste à vérifier le cas de base pour notre choix de la constante c :

$C(1) = 0 \leq 2 \cdot 1 \cdot \ln(1) = 0$ ✓

□

Trier par comparaison "$a \le b$? OUI / NON" **[CLRS 8.1]**

Exemple : Soit a_1, a_2, a_3 une permutation de $\{1,2,3\}$



Chaque permutation correspond à une feuille de l'arbre de décision. Un algo de tri doit effectuer suffisamment de comparaisons pour distinguer les $n!$ permutations !

Théorème : Tout algorithme (déterministe) de tri par comparaison exige $\Omega(n \log n)$ comparaisons en pire cas.

Démo : L'exemple précédent montre qu'il suffit de déterminer la hauteur d'un arbre de décision dans lequel chaque permutation apparaît en tant que feuille.

On a donc un arbre binaire de hauteur h avec $n! \le 2^h$ feuilles

$$\Rightarrow h \ge \log(n!) = \log(n) + \log(n-1) + \dots + \log(2)$$
$$\le n \cdot \log(n)$$

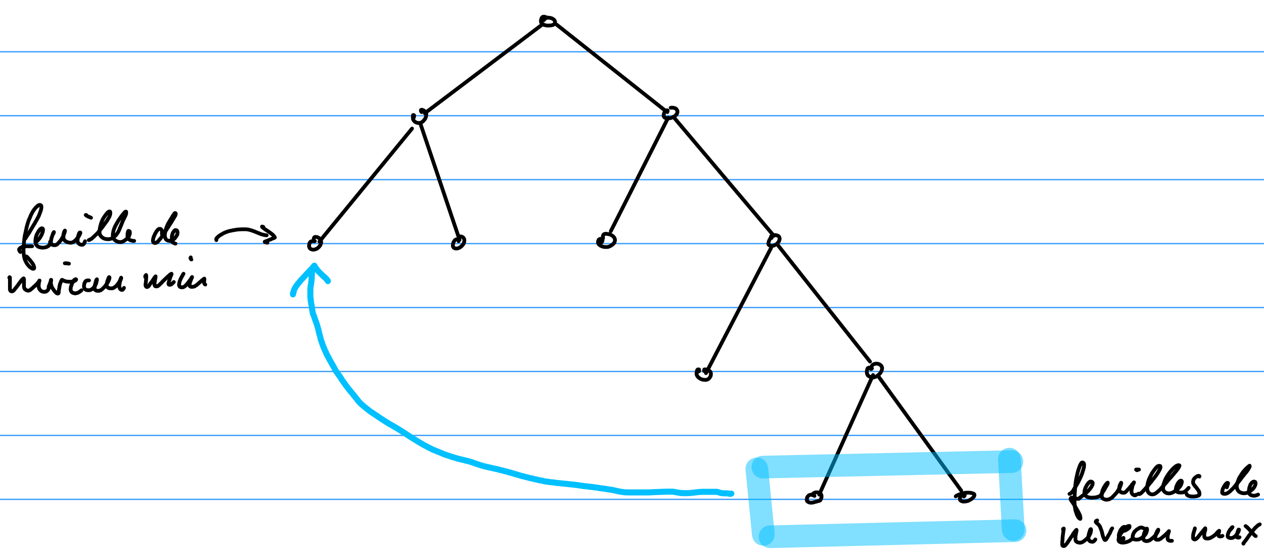
(hauteur = nombre max de comparaisons) $\square$

Théorème : Tout algo déterministe de tri par comparaisons exige $\lfloor \log_2 n! \rfloor$ comparaisons en moyenne.

idée : arbre de décision avec $n!$ feuilles
niveau d'une feuille = # de comparaisons effectuées par l'algo

pour un arbre équilibré le niveau de toute feuille est de $\lceil \log_2 n! \rceil$ ou $\lfloor \log_2 n! \rfloor$ (*)

pour un arbre non-équilibré on considère l'opération suivante



cette opération diminue le moyen niveau des feuilles de l'arbre

⇒ un arbre de décision équilibré minimise le moyen niveau des feuilles

⇒ la borne (*) s'applique :

moyen niveau des feuilles $\geq \lfloor \log_2 n! \rfloor$

$\square$

Théorème : Tout algo randomisé de tri par comparaisons exige $\lfloor \log_2 n! \rfloor$ comparaisons en moyenne.

Démo : $C(I)$: # de comparaisons effectuées en moyenne par un algo randomisé de tri A afin de trier un tableau I de taille n

A_s : algo **déterministe** obtenu à partir de A pour un choix s de bits aléatoires

$C_s(I)$: # de comparaisons effectuées par A_s pour traiter l'entrée I

$$C(I) = \sum_s \Pr[s] \times C_s(I)$$

Donc le coût de A en moyenne sur toutes permutations de $1, 2, \dots, n$ est de

$$\sum_{\text{permutation } I} \Pr[I] \cdot C(I) = \sum_I \Pr[I] \left(\sum_s \Pr[s] \times C_s(I) \right)$$
$$= \sum_s \Pr[s] \left(\sum_I \Pr[I] \times C_s(I) \right)$$
$$\overset{\text{Thm précédent}}{\geq} \sum_s \Pr[s] \lfloor \log_2 n! \rfloor$$
$$= \lfloor \log_2 n! \rfloor$$

□

$\Rightarrow$ TRI-RAPIDE est optimal en moyenne asymptotiquement