# 1 Exercices

**Exercice 1.1.** *Plus grand et deuxième plus grand de $n$ entiers*

On s'intéresse dans cet exercice à la **complexité dans le pire des cas et en nombre de comparaisons** des algorithmes.

1 - Pour rechercher le plus grand et deuxième plus grand élément de $n$ entiers, donner un algorithme naïf et sa complexité.

2 - Pour améliorer les performances, on se propose d'envisager la solution consistant à calculer le maximum suivant le principe d'un *tournoi* (tournoi de tennis par exemple). Plaçons-nous d'abord dans le cas où il y a $n = 2^k$ nombres qui s'affrontent dans le tournoi. Comment retrouve-t-on, une fois le tournoi terminé, le deuxième plus grand ? Quelle est la complexité de l'algorithme ? Dans le cas général, comment adapter la méthode pour traiter $n$ quelconque ?

3 - Montrons l'optimalité de cet algorithme en fournissant une borne inférieure sur le nombre de comparaisons à effectuer. Nous utiliserons la méthode des *arbres de décision*.

3.1 - Montrer que tout arbre de décision qui calcule le maximum de $N$ entiers a au moins $2^{N-1}$ feuilles.

3.2 - Montrer que tout arbre binaire de hauteur $h$ et avec $f$ feuilles vérifie $2^h \geq f$.

3.3 - Soit $A$ un arbre de décision résolvant le problème du plus grand et deuxième plus grand de $n$ entiers, minorer son nombre de feuilles. En déduire une borne inférieure sur le nombre de comparaisons à effectuer.

**Correction.**

1 - Algorithme naïf : recherche du premier maximum, puis du second.

```
début
    max₁ ← T[1];
    pour i allant de 2 à n faire
        si T[i] > max₁ alors
            max₁ ← T[i];
            posmax₁ ← i;
    si posmax₁ ≠ 1 alors max₂ ← T[1] sinon max₂ ← T[2];
    pour i allant de 2 à n avec i ≠ posmax₁ faire
        si T[i] > max₂ alors
            max₂ ← T[i];
    retourner max₁, max₂;
fin
```

Nombre de comparaisons :

| | |
|---|---|
| Premier maximum, sur n valeurs : | $n - 1$ |
| Second maximum, sur n - 1 valeurs : | $n - 2$ |
| | $2n - 3$ |

2 - Cas où $n = 2^k$ :

On calcule le premier maximum à la façon d'un tournoi de tennis. On cherche ensuite le second maximum, en prenant le maximum parmi les adversaires que le premier à rencontré. Ainsi, dans l'arbre donné à la figure 1, le second maximum est nécessairement un élément contenu dans le chemin rouge.

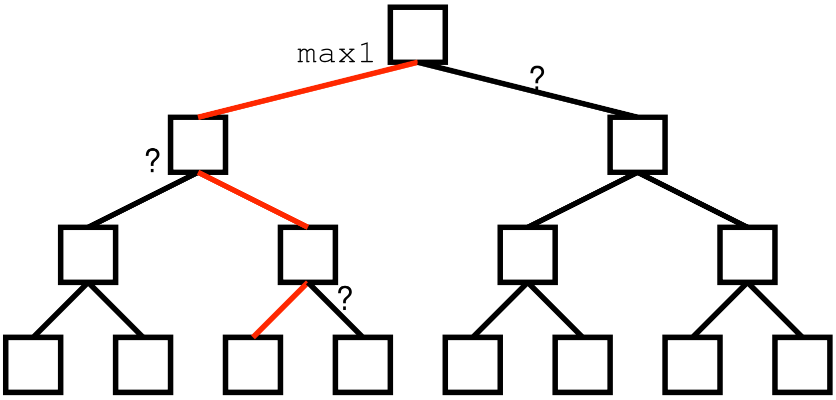


FIG. 1 – Arbre des «compétitions» effectuées

| | | |
|---|---|---|
| | Premier maximum : | $n-1=2^k-1$ |
| Nombre de comparaisons : | Second maximum : | $k-1$ |
| | | $2^k+k-2=n+log_2n-2$ |

Cas général : n quelconque

S'il n'est pas rempli et a une profondeur minimale, la branche de longueur maximale est de longueur $\lceil log_2 n\rceil$.

Au pire, nombre de comparaisons :

| | |
|---|---|
| Premier maximum : | $n-1$ |
| Second maximum : | $\lceil log_2 n\rceil - 1$ |
| | $n+\lceil log_2 n\rceil - 2$ |

3.1 - Pour chercher le maximum parmis $N$ valeurs, on doit effectuer $N-1$ comparaisons. On a donc $2^{N-1}$ feuilles dans l'arbre de décision.

3.2 - Par récurrence sur la hauteur $h$ de l'arbre

- Pour $h=0$ : 1 feuille au plus
- On considère un arbre de hauteur $h+1$ , on a alors deux cas, soit la racine a un seul fils soit il en a deux.
  - un fils : on alors le nombre de feuille qui correspond à celui de l'arbre partant du fils, qui est de hauteur $h$ , et $2^(h+1) \geq 2^h$ ce qui va bien.
  - deux fils : on a alors le nombre de feuilles qui correspond à la somme de celles des deux sous arbresz partants des deux fils : $f = f_1 + f_2$, en ayant pour chacun une hauteur maximale de h soit : $2^(h+1) \geq 2^h + 2^h \geq 2^{h_1} + 2^{h_2} \geq f_1 + f_2 \geq f$ cqfd.

3.3 - On partitionne les feuilles de $A$ selon la valeur du premier maximum pour former les $A_i$ ; $A_i$ est au final l'arbre $A$ dont on a enlevé tous ce qui n'aboutissait pas à une feuille concluant que le premier maximum était $i$. Ces $A_i$ sont des arbres donnant un maximum parmi n-1 éléments donc ils ont chacun un nombre de feuilles tel que : nombre de feuilles de $A_i \geq 2^{n-2}$ Donc en considérant $A$ comme la 'fusion' de ces arbres qui en forme une partition, on a :

$$
\begin{aligned}
nombre de feuilles de A &\geq \sum_{i=1}^{n} 2^{n-2} \\
&\geq n2^{n-2} \\
2^{hauteur} &\geq n2^{n-2} \\
hauteur &\geq \lceil \log_2(n2^{n-2}) \rceil \\
&\geq n - 2 + \lceil \log_2 n \rceil
\end{aligned}
$$

**Exercice 1.2.** *Matrices de Tœplitz*

Une *matrice de Tœplitz* est une matrice $n \times n$ $(a_{i,j})$ telle que $a_{i,j} = a_{i-1,j-1}$ pour $2 \leq i, j \leq n$.

1 - La somme de deux matrices de Tœplitz est-elle une matrice de Tœplitz ? Et le produit ?

2 - Trouver un moyen d'additionner deux matrices de Tœplitz en $\mathcal{O}(n)$.

3 - Comment calculer le produit d'une matrice de Tœplitz $n \times n$ par un vecteur de longueur $n$ ? Quelle est la complexité de l'algorithme ?

**Correction.**

1 – Par linéarité, la somme de deux Tœplitz reste une Tœplitz. Ce n'est pas vrai pour le produit. Par exemple,

$$
\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}
$$

2 – On n'additionne que les premières lignes et les premières colonnes, ce qui fait $2n - 1$ opérations.

3 – On ne considère que les matrices de taille $2^k \times 2^k$.
On décompose la matrice en blocs de taille $n = 2^{k-1}$

$$
\mathbf{M} \times \mathbf{T} = \begin{pmatrix} A & B \\ C & A \end{pmatrix} \times \begin{pmatrix} X \\ Y \end{pmatrix}
$$

On pose :

$$
\begin{aligned}
U &= (C + A)X \\
V &= A(Y - X) \\
W &= (B + A)Y
\end{aligned}
$$

et on calcule :

$$
\mathbf{M} \times \mathbf{T} = \begin{pmatrix} W - V \\ U + V \end{pmatrix}
$$

Calcul de la complexité : On note respectivement $M(n)$ et $A(n)$ le nombre de multiplications et d'additions pour un produit de matrices $n \times n$.

$$\begin{aligned} M(2^k) &= 3M(2^{k-1}) \\ M(1) &= 1 \end{aligned}$$

$$\begin{aligned} A(2^k) &= 3A(2^{k-1}) + 2(2^k - 1) + 3.2^{k-1} \\ A(1) &= 0 \end{aligned}$$

On résoud les récurrences :
on pose $M_s = M(2^s)$, et on pose $A_s = A(2^s)$.

le calcul pour les multiplications :

$$\begin{cases} M_s = 3M_{s-1} \\ M_0 = 1 \end{cases}$$

$$(E-3)M_s = \bar{0}; M_s = 3^{\log n} = n^{\log 3}$$

puis le calcul pour les additions :

$$\begin{cases} A_s = 3A_{s-1} + 7.2^{s-1} - 2 \\ A_0 = 0 \end{cases}$$

$$\begin{array}{l} (E-3)(E-2)(E-1)\{A_s\} = \bar{0} \\ A_s = i3^s + j2^s + l \\ A_0 = A(1) = 0 \longrightarrow i + j + l = 0 \\ A_1 = A(2) = 5 \longrightarrow 3i + 2j + l = 5 \\ A_2 = A(4) = 27 \longrightarrow 9i + 4j + l = 27 \\ \text{D'où } i = 6, j = -7, l = 1 \end{array}$$