Algorithmique I
Année 2007-2008

A. Benoit, B. Depardon
JF. Pineau, C. Rezvoy

TD n°2

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.

Question 1.1 Pour rechercher le plus grand et deuxième plus grand élément de n entiers, donner un algorithme naïf et sa complexité.

Solution : *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 :	$n-2$
	$2n-3$

□

Question 1.2 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 ?

Solution : *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.

	Premier maximum :	$n-1 = 2^k - 1$
Nombre de comparaisons :	*Second maximum :*	$k-1$
		$2^k + k - 2 = n + log_2 n - 2$

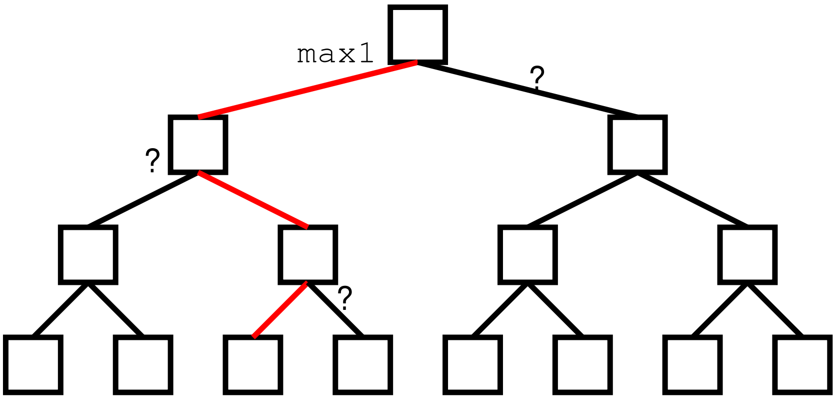

FIG. 1 – Arbre des «compétitions» effectuées

Cas général : n quelconque

On cherche k tel que $2^k \geq n$: prendre $\min\{k|2^k \geq n\} = \lceil \log_2 n \rceil$ *(voir Figure 2). Avec le raisonnement d'avant on a $n-1$ matches pour trouver le premier maximum, et on a des branches de longueur au plus égale à $\lceil \log_2 n \rceil$, donc dans le pire des cas on a à faire $\lceil \log_2 n \rceil - 1$ comparaisons entre les noeuds battus par le max. Soit un total de $n + \lceil \log_2 n \rceil - 2$.*

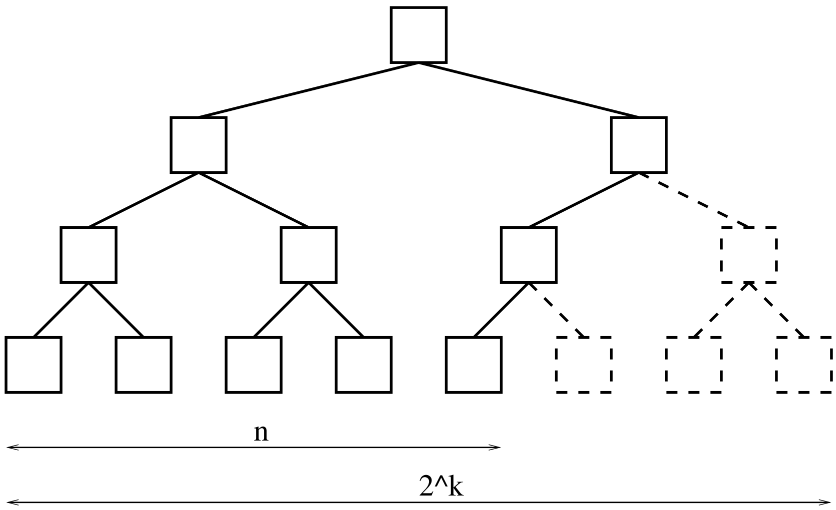

FIG. 2 – Arbre lorsque n est quelconque

Au pire, nombre de comparaisons :

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

Remarque : *plus formellement pour évaluer la hauteur maximale de l'arbre représentant le tournoi, faire une récurrence sur* $\forall d \geq 0, \forall n \geq 2 \; 2^d < n \leq 2^{d+1} \Rightarrow h(n) = d + 1$.

□

Question 1.3 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*.

Remarque : ***Arbre de décision d'un algorithme :*** *arbre représentant toutes les exécutions possibles de l'algorithme sur toutes les donnés d'une certaine taille :*
- *feuilles : résultats des différentes exécutions (plusieurs peuvent donner le même résultat)*
- *nœuds internes : tests pour aiguillage, ici nœud = comparaison (gauche : oui, droite : non) on a donc un arbre binaire.*

Remarque : ***Complexité en tests :*** *une exécution = un branche, donc le nombre de comparaisons est égal à la hauteur de la branche, soit la hauteur de l'arbre.*

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

Solution : *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 (nombre de feuilles d'un arbre binaire de hauteur $N-1$). □

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

Solution : *Par récurrence sur la hauteur h de l'arbre*

- *Pour $h = 0$: 1 feuille au plus*
- *On considère vrai jusqu'à la hauteur h : $2^h \geq f$*
- *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 feuilles 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 arbres 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.*

□

1.3.3 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.

Solution : *La Figure 3 présente un arbre de décision pour le maximum et deuxième maximum de 4 valeurs. 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. On supprime alors les nœuds où il y a un test avec $T[i]$, ces nœuds sont forcément nœud avec un seul fils (sinon ce serait une feuille). 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 :*

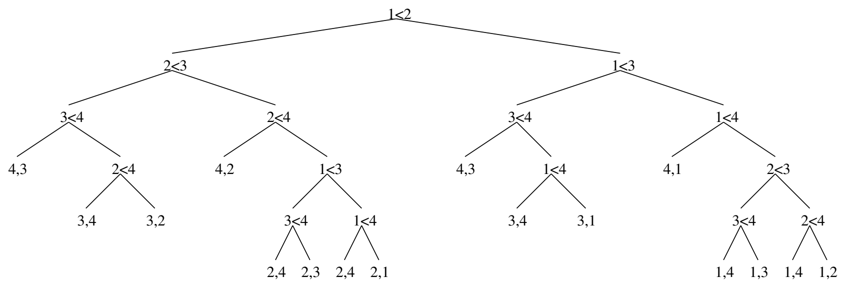

FIG. 3 – Arbre de décision pour 4 valeurs : les feuilles donnent le max et 2^{me} max

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

□

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 $2 \leq i, j \leq n$.

Question 2.1 La somme de deux matrices de Tœplitz est-elle une matrice de Tœplitz ? Et le produit ?

Solution : *Par linéarité, la somme de deux matrices de Tœplitz reste une matrice de Tœplitz. Ce n'est pas vrai pour le produit. Par exemple,*

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

□

Question 2.2 Trouver un moyen d'additionner deux matrices de Tœplitz en $\mathcal{O}(n)$.

Solution : *On n'additionne que les premières lignes et les premières colonnes, ce qui fait* $2n-1$ *opérations, car la matrice est représentée par les éléments de la première ligne et première colonne : les diagonales étant égales à leur premier élément.* □

Question 2.3 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 ?

Solution : *On ne considère que les matrices de taille* $2^k \times 2^k$.

On décompose la matrice en blocs de taille $n = 2^{k-1}$

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

Si on fait le calcul suivant on a 4 multiplications :

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

Si on pose :

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

et qu'on calcule :

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

on n'a plus que 3 multiplications.

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

$$\left\{ \begin{array}{rcl} M(1) & = & 1 \\ M(2^k) & = & 3 \cdot M(2^{k-1}) \end{array} \right.$$

$$\left\{ \begin{array}{rcl} A(1) & = & 0 \\ A(2^k) & = & \underbrace{3 \cdot A(2^{k-1})}_{\text{de } U,\ V \text{ et } W} + \underbrace{2 \cdot (2^k - 1)}_{\text{de } U \text{ et } W} + \underbrace{2^{k-1}}_{\text{de } V} + \underbrace{2 \cdot 2^{k-1}}_{\text{de } W-V \text{ et } U+V} = 3 \cdot A(2^{k-1}) + 2 \cdot (2^k - 1) + 3 \cdot 2^{k-1} \end{array} \right.$$

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

le calcul pour les multiplications :

$$\left\{ \begin{array}{l} M_s = 3M_{s-1} \\ M_0 = 1 \end{array} \right.$$

$$(E - 3)M_s = \bar{0} \Rightarrow M_s = k \cdot 3^s = 3^{\log n} = n^{\log 3}$$

puis le calcul pour les additions :

$$\left\{ \begin{array}{l} A_s = 3A_{s-1} + 7 \cdot 2^{s-1} - 2 \\ A_0 = 0 \end{array} \right.$$

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

$\square$