

□ Exercice 1 : *Maximum*

On s'intéresse au problème de la recherche du maximum des éléments de la liste non vide l .

1. Ecrire en OCaml une fonction `max int list -> int` qui renvoie le maximum de la liste donnée en argument. Donner sa complexité en nombre de comparaisons effectuées.
2. Mettre en place une stratégie *diviser pour régner* afin de résoudre ce problème.
3. En donner l'implémentation en OCaml.
4. Déterminer sa complexité en nombre de comparaisons effectuées et conclure.

□ Exercice 2 : *Algorithme de Karatsuba*

On considère deux nombres M et N ayant n chiffres en base 10, on suppose n paire et on note $k = n/2$. Ces deux nombres peuvent donc s'écrire $M = a \times 10^k + b$ et $N = c \times 10^k + d$ où a, b, c et d sont des nombres à k chiffres.

1. Si on développe "normalement" $(a \times 10^k + b)(c \times 10^k + d)$, combien de produits de nombres à k chiffres sont nécessaires pour calculer MN ?
2. Vérifier que $MN = ac \times 10^{2k} + (ac + bd - (a - b)(c - d)) \times 10^k + bd$. L'algorithme de Karatsuba, consiste à utiliser récursivement cette expression afin de calculer MN .
3. Combien de produits de nombres à k chiffres sont nécessaires dans le calcul de cette expression ?
4. On considère maintenant que les additions, soustractions et décalages d'exposant sur un nombre à 2^i chiffres s'effectue en $\mathcal{O}(2^i)$ et on note $T(2^i)$ le temps nécessaire au calcul du produit de deux nombres à 2^i chiffres en utilisant l'algorithme de Karatsuba. En déduire qu'il existe un entier A tel que $T(2^i) \leq 3T(2^{i-1}) + A2^i$
5. En divisant par 3^i et en sommant pour $i = 1$ à k , montrer que $T(2^k) \leq C3^k$
6. En déduire que l'algorithme de Karatsuba a une complexité $\mathcal{O}(n^{\log_2 3})$

□ Exercice 3 : *Recherche d'un élément dans une matrice*

On considère une matrice A de taille $m \times n$ et on suppose que chaque ligne et chaque colonne est rangée par ordre croissant. Par exemple :

$$\begin{pmatrix} 12 & 20 & 21 & 22 & 28 & 32 \\ 15 & 21 & 27 & 28 & 31 & 34 \\ 18 & 24 & 29 & 33 & 42 & 44 \\ 30 & 27 & 37 & 45 & 47 & 50 \end{pmatrix}$$

Le but de l'exercice est de mettre en place une stratégie *diviser pour régner* afin de rechercher si un entier e est présent ou non dans la matrice.

1. Montrer qu'en comparant e avec l'élément situé en ligne $m/2$, colonne $n/2$ on peut limiter la recherche à trois sous matrices de taille $m/2 \times n/2$
2. En déduire une stratégie du type *diviser pour régner* permettant de résoudre un problème (on donnera les étapes de l'algorithme sans le programmer)
3. Déterminer le coût maximal C_n en nombre de comparaison de cet algorithme dans le cas où $n = m = 2^k$ ($k \in \mathbb{N}$).
 ☛ On pourra supposer que $(C_n)_{(n \in \mathbb{N})}$ vérifie $C_n = 3C_{\frac{n}{2}} + \alpha$ où α est une constante représentant les coûts des opérations autre que les comparaisons.

□ Exercice 4 : *Parenthésage optimal de multiplications matricielles*

On rappelle que le nombre de multiplications nécessaires à la multiplication de deux matrices A de dimension $m \times n$ et B de dimension $n \times p$ est nmp .

1. On considère 4 matrices A_1 (5×2), A_2 ($2, 10$), A_3 ($10, 4$) et A_4 ($4, 1$). Pour chacun des parenthésages suivant déterminer le nombre de multiplications nécessaire :
 — $(A_1 A_2)(A_3 A_4)$
 — $A_1(A_2(A_3 A_4))$
 — $A_1((A_2 A_3)A_4)$
 — $((A_1 A_2)A_3)A_4$
 — $(A_1(A_2 A_3))A_4$

2. Montrer que le problème de la recherche d'un parenthésage minimisant le nombre de multiplication possède la propriété de sous structure optimale.
3. Se trouve-t-on dans une situation de chevauchement des sous problèmes ?
4. On note (l_i, c_i) les dimensions des matrices $(A_i)_{1 \leq i \leq N}$ et $m(i, j)$ ($1 \leq i < j \leq N$) le nombre minimal d'opérations dans le calcul du produit des $A_i \dots A_j$. Ecrire la relation de récurrence liant $m(i, j)$ aux $m(i, k)$ et aux $m(k+1, j)$ pour $i \leq k < j$

□ **Exercice 5** : *Problème du lâcher d'oeuf*

On considère un immeuble de N étages (numérotés de 1 à N), et on dispose de p oeufs. Le but du problème est de déterminer le plus petit entier k ($1 \leq k \leq N$) tel qu'un oeuf lancé depuis l'étage k se brise en touchant le sol, on appellera cet étage l'étage *critique* et on le note k_c . Si les oeufs ne se brisent pas en étant lancé de l'étage N , on posera par convention $k_c = N + 1$. On fait les hypothèses suivantes :

- Si un oeuf se brise en étant lancé depuis l'étage k alors il se brise depuis tous les étages situés au dessus
- Si un oeuf ne se brise pas en étant lancé depuis l'étage k alors il en est de même pour un lancer depuis un étage inférieur
- Un oeuf qui ne s'est pas brisé lors d'un lancer peut-être réutilisé pour un lancer suivant sans en affecter le résultat.

On veut résoudre ce problème en minimisant le nombre de lancers à effectuer pour déterminer l'étage critique k_c à partir duquel les oeufs se brisent. On note $\varphi(N, p)$ le nombre minimal de lancers à effectuer *dans le pire des cas* afin de déterminer k_c .

1. Donner les valeurs de $\varphi(0, p)$ et $\varphi(1, p)$.
2. Quelle est la seule stratégie possible dans le cas où $p = 1$? En déduire $\varphi(N, 1)$.
3. On suppose maintenant qu'on dispose de $p > 1$ oeuf, et on lance un oeuf depuis un étage k avec $1 \leq k \leq N$. En envisageant les deux possibilités (l'oeuf se casse ou non), montrer que
$$\varphi(N, p) = \min_{1 \leq k \leq N} \{1 + \max(\varphi(k-1, p-1), \varphi(N-k, p))\}.$$
4. Donner une implémentation en OCaml la fonction φ .