Fiche d'exercices I: Algorithmes et Complexité

Exercice 1 (Conversion) *La représentation d'un entier n en base décimale se fait comme suit: $n = \sum_i (a_i 10^i)$ avec $a_i \in \{0, ..., 9\}$ pour $i \geq 0$. Écrire un algorithme de conversion entier$\leftrightarrow$décimal. Complexité?*

Exercice 2 (Calcul approché) *En utilisant le développement limité de $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$, donnez une approximation à ϵ prés de e^x pour $x \in [0, 1]$.*

Exercice 3 (Méthode SX) *On désire calculer x^n pour $n \in \mathbb{N}$. En écrivant n en binaire, déduire un algorithme efficace pour le calcul de x^n. Que faire si $n \in \mathbb{R}$? Étudier la complexité de votre algorithme. Peut-on améliorer cet algorithme? Essayer $n = 15$ et comparez votre algorithme avec cette factorisation $(x^3)^5$.*

Exercice 4 (Fonctions récursives) *Calculer $\binom{n}{p}$. Quelle est la complexité de votre algorithme? Soit la suite de Fibonacci définie par:*

- *$f(0) = 0$ et $f(1) = 1$.*
- *$f(n) = f(n-1) + f(n-2)$ pour $n \geq 2$.*

Calculer $f(n)$ grâce à un algorithme récursif. Dérécursivez les algorithmes précedents.

Exercice 5 (Min/Max) *Soit $S = \{a_1, ..., a_n\}$ n éléments distincts de $\mathbb{R}$. Calculer $\min S$, $\max S$. Le deuxième plus grand élément deux$(S) = \min(S \backslash \{\min S\})$.*

Exercice 6 (La fausse piéce) *On considére un ensemble de $p = 8$ piéces de monnaie de $10F$. L'une d'elles se démarque par son poids (plus lourd ou plus léger). On dispose d'une balance à plateau. Donner une solution pour trouver la pièce différente. Même exercice mais avec $p = 12$.*

Exercice 7 (Transposée) *On considére une matrice $(A)_{i,j}$ de $\mathcal{M}_{\mathbb{N}}(n, m)$. Donnez un algorithme qui calcule A^t (puis une version récursive).*

Exercice 8 (Crible d'Érathosthène) *Le crible d'Érathosthène permet de calculer les nombres premiers de 2 à n. L'algorithme est le suivant:*

début
Mettre les nombres de 2 *à* n *dans le crible*;
répéter
Sélectionner dans le crible le plus petit nombre p*. Ce nombre est premier.*
Retirer du crible tous les multiples de p*.*
jusqu'à *Vide(Crible)*
fin

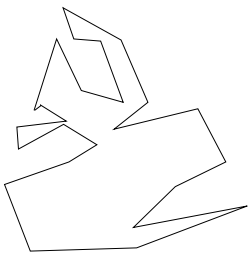
Quelle est la complexité de cet algorithme en fonction de n*, puis en fonction de la taille de la donnée, c'est-à-dire le nombre de bits* b *qu'il faut pour coder* n *(on pourra utiliser* $\sum_{p \leq n \text{ premier}} \frac{1}{p} = O(\log \log n)$*).*

Exercice 9 (Que fais-je?) *On considére un tableau* T *de* n *entiers et l'algorithme suivant:*

début
$i \leftarrow 1$;
tant que $i < n$ **faire**
début
pour j **de** n **à** $i+1$ **faire**
si $T[j] < T[j-1]$ **alors** $T[j] \leftrightarrow T[j-1]$; **fsi**
$i \leftarrow i+1$;
fin
fin

- *Étudier la complexité de l'algorithme en fonction du nombre de comparaisons.*
- *Étudier la complexité de l'algorithme en fonction du nombre d'affectations.*

Exercice 10 (Triangulation de Mickey) *Soit* P *un polygone simple du plan. Une triangulation de* P *est un ensemble de triangles tels que les sommets de ces triangles sont les sommets du polygone, leur réunion est* P *et toute paire de triangles est d'intérieur disjoint. Donner plusieurs triangulations du polygone représenté ci-dessous:*



On appelle **oreille** *de P tout triangle ABC où ABC sont trois sommets consécutifs de P tel que $ABC \subseteq P$. Montrez que tout polygone simple posséde au moins 2 oreilles. En déduire un algorithme pour trianguler P. Quelle est sa complexité?*

Exercice 11 (mot de Dick) *Soit Σ un ensemble fini de symboles appelé* **alphabet**. *Un* **mot** *sur Σ est une suite finie de symboles de Σ. On appelle* **préfixe** *d'un mot $\sigma_1 ... \sigma_n$ sur Σ tout mot de la forme $\sigma_1 ... \sigma_p$ pour $1 \leq p \leq n$.*
On appelle mot de Dick sur l'alphabet $\Sigma = \{x, \overline{x}\}$, tout mot sur cet alphabet tel que le nombre d'occurences de x est le même que $\overline{x}$ et le nombre d'occurences de x est toujours supérieur ou égal à celui de $\overline{x}$ pour tout préfixe du mot.

- *Donner un algorithme qui teste si un mot est un mot de Dick.*
- *Écrire un algorithme qui calcule tous les mots de Dick de longueur $2p$*

Exercice 12 (Multiplication rapide) *Soit $X = (x_{ij})$ un matrice de $\mathcal{M}_{\mathbb{N}}(m, n)$ et $Y = (y_{ij})$ un matrice de $\mathcal{M}_{\mathbb{N}}(n, s)$. On désire calculer $Z = X \times Y = (z_{ij})$ où Z est une matrice de $\mathcal{M}_{\mathbb{N}}(m, s)$*

Algorithme direct. *Donner l'algorithme "direct". Montrer que le nombre de multiplication est mns, le nombre d'addition $ms(n-1)$.*

Algorithme de S. Winograd (1967). *On considére le schéma algorithmique suivant:*

$$z_{ik} = \sum_{1 \leq j \leq \frac{n}{2}} (x_{i,2j} + y_{2j-1,k})(x_{i,2j-1} + y_{2j,k}) - a_i - b_k + c_{ik}$$

avec

$$a_i = \sum_{1 \leq j \leq \frac{n}{2}} x_{i,2j} x_{i,2j-1}$$

$$b_k = \sum_{1 \leq j \leq \frac{n}{2}} y_{2j-1,k} y_{2j,k}$$

et

$$c_{i,k} = \begin{cases} 0 & n \text{ pair} \\ x_{i,n} y_{n,k} & n \text{ impair} \end{cases}$$

Montrer que l'on diminue le nombre de multiplications en augmentant le nombre d'additions.

Algorithme de V. Strassen(1968). *Basé sur le fait que la multiplication de matrice 2×2 peut se faire en 7 multiplications:*

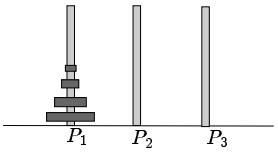
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} =$$

$$\begin{pmatrix} aA+bB & w+(c+d)(C-A)+(a+b-c-d)D \\ w+(a-c)(D-C)-d(A-B-C+D) & w+(a-c)(D-C)+(c+d)(C-A) \end{pmatrix},$$

avec $w = aA-(a-c-d)(A-C+D)$. *Généraliser cette méthode aux matrices* $2^d \times 2^d$ *et montrer que le nombre d'opérations est passé à* $O(n^{\log 7})$.

Coppersmith et. al *Algorithme en* $O(n^{2.5161})$.

Exercice 13 (Tour de Hanoï) *L'un des algorithmes récursifs les plus célébres. On dispose de trois piquets notés* P_1, P_2, P_3. *Initialement sont empilés* n *disques par taille croissante sur* P_1. *On désire faire passer les* n *disques de* P_1 *à* P_3 *tel que l'ordre croissant des disques doit toujours être respecté.*



- *Écrire un algorithme récursif qui résoud ce problème.*
- *Calculer sa complexité (*$n = 3$, $n = 10$, $n = 100$*).*
- *Généraliser cette méthode à* p *piquets.*

Exercice 14 ($O(\cdot)$-notation) *Montrer que si* $\lim_{+\infty} f = +\infty$ *alors on a* $O(nf(n)) \neq O(n)$.

Exercice 15 (fonctions de $\mathbb{N}^{\mathbb{N}}$) *Trouver deux fonctions* f *et* g *de* $\mathbb{N} \to \mathbb{N}$ *tels que* $f(n) \notin O(g(n))$ *et* $g(n) \notin O(f(n))$.

Exercice 16 (Dichotomie et complexité)

- *une fonction* f *de* $\mathbb{N}$ *dans* $\mathbb{R}^+$ *est "finalement" croissante ssi* $\exists n_0,\ \forall n \geq n_0,\ f(n) \leq f(n+1)$. *Si de plus pour un entier* $b \geq 2$ *on a* $f(bn) = O(f(n))$ *la fonction* f *est dites* b-lisse. *Montrez que si* f *est* b*-lisse alors elle est* c*-lisse pour tout* $c \geq 2$. *Donner des exemples de fonctions lisses et non-lisses.*
- *Soit* P *un prédicat sur* $\mathbb{N}$. *On définit la classe* $O(f(n)|P(n))$ *par:*

$$O(f(n)|P(n)) = \{g, \exists c \in \mathbb{R}^+\ \forall n \in \mathbb{N}\ P(n) \Rightarrow g(n) \leq cf(n)\}$$

Soit $b \geq 2$ *un entier et* f *une fonction* b*-lisse de* $\mathbb{N}$ *dans* $\mathbb{R}^+$. *Soit* t *une fonction de* $\mathbb{N}$ *dans* $\mathbb{R}^+$ *finalement croissante telle que* $t(n) \in O(f(n)|n$ n *est une puissance de* $b)$. *Montrez que* $t(n) \in O(f(n))$.

- *En déduire que la fonction t définie par:*

$$t(1) = a$$

$$\forall n > 1,\ t(n) = t(\lfloor \frac{n}{2} \rfloor) + t(\lceil \frac{n}{2} \rceil) + bn$$

  où $(a, b) \in \mathbb{R}^2$, appartient à $O(n \log n)$.

- *Avez-vous des algorithmes dont la complexité s'inscrit dans ce contexte?*

Exercice 17 ("Diviser pour Régner") *Le paradigme "Diviser pour Régner" est basé sur la division dichotomique des données: pour résoudre un problème de taille n, on le divise en deux sous-problèmes de taille $\lfloor \frac{n}{2} \rfloor$ et $\lceil \frac{n}{2} \rceil$, on résoud les sous-problèmes et on fusionne les deux résultats afin d'obtenir la solution finale. Utiliser l'algorithme précedent pour montrer que si la fusion peut se faire en temps linéaire alors l'algorithme à une complexité en $O(n \log n)$ pour traiter un problème de taille n.*