Algorithmique

1 Calculs sur les entiers

Exercice 1 Programmer RAPIDEMENT une fonction récursive fournissant des coefficients de Bezout associés à deux entiers premiers entre eux.

Exercice 2 Montrer que si $0 \leq a, b \leq F_N$ (N-ième terme de la suite de Fibonacci), alors l'algorithme d'Euclide pour calculer $a \wedge b$ demande moins de $N + 10$ divisions euclidiennes.

Exercice 3 *Algorithme de Solovay-Strassen*

Si $(a, b) \in \mathbb{Z}^2$, $b \neq 0$, le symbole de Jacobi $\left(\frac{a}{b}\right)$ est un entier (qui vaut 0,1 ou -1) qui se calcule en temps polynomial en $\ln a$ et $\ln b$ tel que :

1. si n est premier, alors $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2}\ [n]$ pour tout $a \in [\![1, n-1]\!]$;
2. si n n'est pas premier, alors il y a au moins la moitié des $k \in [\![1, n-1]\!]$ premiers avec n qui vérifient : $\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2}\ [n]$

Déterminer un algorithme prenant en entrée un entier n, et retournant en temps "poly-log(n)" un booléen b tel que :

1. si n est premier, b vaut toujours `true` ;
2. si n n'est pas premier, b vaut `false` avec une probabilité supérieure à $1 - \frac{1}{2^{50}}$.

Cet algorithme est-il satisfaisant ? Le comparer au test déterministe.

2 Diviser pour régner

Exercice 4 Considérons deux matrices $A, B \in \mathcal{M}_2(\mathbb{A})$, où $\mathbb{A}$ est un anneau *pas forcément commutatif*. On note $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, $B = \begin{pmatrix} e & g \\ f & h \end{pmatrix}$, et $C = AB = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$. On a $r = ae + bf$, $s = ag + bh$, $t = ce + df$ et $u = cg + dh$.

1. Evaluer le nombre de sommes et de multiplications d'éléments de $\mathbb{A}$ nécessaires à la multipication de deux matrices de $\mathcal{M}_2(\mathbb{A})$ par la méthode naïve.

   Si on note $p_1 = a(g-h)$, $p_2 = (a+b)h$, $p_3 = (c+d)e$, $p_4 = d(f-e)$, $p_5 = (a+b)(e+h)$, $p_6 = (c-d)(g+h)$ et $p_7 = (a-c)(e+g)$, on a alors $s = p_1 + p_2$, $t = p_3 + p_4$, $r = p_5 + p_4 - p_2 + p_6$, et $u = p_5 + p_1 - p_3 - p_7$.
2. Evaluer le nombre de sommes et de multiplications d'éléments de $\mathbb{A}$ nécessaires à la multiplication de deux matrices de $\mathcal{M}_2(\mathbb{A})$ grâce aux formules précédentes, dues à Strassen.

3. En appliquant ces formules lorsque $\mathbb{A} = \mathcal{M}_n(\mathbb{B})$, montrer qu'on peut ainsi multiplier des éléments de $\mathcal{M}_{2n}(\mathbb{B})$.
4. En appliquant récursivement cette méthode, évaluer le nombre de multiplications et sommes nécessaires dans $\mathbb{A}$ pour multiplier deux matrices (n, n) (commencer par $n = 2^k$).

EXERCICE 5

1. Proposer un algorithme (en temps polynomial en n) pour déterminer l'enveloppe convexe d'un ensemble E de n points donnés par leurs coordonées. Montrer que sa complexité est (au moins) en n^2 "opérations élémentaires".
2. On propose de diviser les points en deux sous-ensembles disjoints E_1 et E_2 de taille égale (± 1) situés respectivement à droite et à gauche d'une ligne médiane : combien d'opérations sont nécessaires pour cette phase ?
3. Connaissant les enveloppes de E_1 et E_2, comment trouver celle de E ?
4. Trouver un contre-exemple à votre proposition précédente, puis proposer une autre solution (à chaque passage, incrémenter un compteur initialisé à 0 en début d'exo : lorsqu'il vaut 3, passer à la question suivante).
5. Quel complexité obtient-on pour l'algorithme "diviser pour régner" ainsi obtenu ?

EXERCICE 6 *Médiane d'un ensemble*

La médiane d'un ensemble fini d'entiers E est l'unique élément $x_0 \in E$ tel que $\{y \in E \mid y < x_0\}$ est de cardinal $= \left\lfloor \dfrac{|E|}{2} \right\rfloor$. Par exemple, la valeur médiane de$\{1, 12, 11, 2\}$ est 11.

1. Décrire un algorithme (très !) simple calculant la médiane d'un ensemble donné en $O(n^2)$ comparaisons.
2. Donner un algorithme permettant d'obtenir la valeur médiane de la réunion de deux ensembles de même taille n **disjoints et déjà triés** en $O(\ln n)$ comparaisons dans le pire des cas. *On pourra dichotomiser...*
   Traiter l'exemple : $E_1 = [1, 2, 3, 6, 7, 8, 9, 15]$ et $E_2 = [4, 5, 10, 11, 12, 13, 14, 16]$.
3. Quelle est l'amélioration ?

3 Programmation dynamique

EXERCICE 7 Un sac-à-dos peut contenir des objets pour un poids total de P_{max}. On cherche à optimiser le contenu, avec des objets caractérisés par un poids p_k et une valeur v_k. **On suppose les p_k entiers**[1]**.**

1. Proposer un ensemble d'objets pour lequel la méthode gloutonne consistant à prendre l'objet de plus grande "valeur massique" (et recommencer) n'est pas optimal.
2. Proposer un algorithme permettant de choisir, parmi n objets (p_k, v_k), le sous-ensemble de plus grande valeur tenant dans le sac-à-dos. Le temps d'exécution doit être polynomial en n.

[1] dans le cas où les poids ne sont pas entiers, on obtient un problème "NP-complet"

Exercice 8 Donner un algorithme permettant de trouver la taille de la plus grande sous-suite croissante d'une suite de n entiers. On devra effectuer $O(n^2)$ comparaisons et opérations arithmétiques élémentaires.

Comment modifier l'algorithme précédent pour trouver effectivement cette plus grande sous-suite (et non seulement sa taille) sans stocker plus d'information et en seulement $O(n)$ opérations supplémentaires ?

Exercice 9 On cherche à allouer (de façon injective !) m paires de skis de longueur $l_1, \ldots, l_m$ à n skieurs de tailles $t_1, \ldots, t_n$ $(m \geq n)$, en minimisant la différence entre la longueur des paires de ski et la hauteur des skieurs. Cela revient à chercher une injection $a : [\![1, n]\!] \to [\![1, m]\!]$ qui minimise :

$$A[a, m, n] = \sum_{k=1}^{n} \left| t_k - l_{a(k)} \right|.$$

On note $A[m, n]$ ce minimum. De même, $A[i, j]$ désigne le minimum du même problème, mais avec les i premiers skis et les j premiers skieurs.

Les t_i et l_i sont supposés triés dans l'ordre croissant.

1. Montrer qu'il existe une allocation a optimale *croissante*. C'est une tellle allocation que l'on recherche par la suite.
2. Montrer :

$$A[i, j] = \mathrm{Min}\Big(A[i, j-1], A[i-1, j-1] + |l_i - t_j|\Big).$$

   En déduire un algorithme permettant d'allouer les skis en $O(n \ln n + m \ln m + (m-n)n)$ opérations élémentaires (sommes, différence, comparaisons).
3. Quid lorsque $m = n$?

Exercice 10 On cherche à multiplier N matrices $A_k \in \mathcal{M}_{n_k, m_k}(\mathbb{A})$ (avec $m_k = n_{k+1}$ pour tout $k \in [\![1, N-1]\!]$).

1. Comment faire dans le cas où A_1 et A_3 sont des matrices lignes, et A_2 est une matrice colonne ?
2. Poser le problème.
3. Y répondre avec un algorithme en N^3 !
4. Implémenter sa solution en Caml.

Exercice 11 On veut construire une tour la plus haute possible avec des briques parallélipédiques de n types différents de tailles (x_i, y_i, z_i), sachant qu'une brique doit toujours être posée sur une brique de tailles strictement plus petites (on compare les tailles des faces qui se voient).

Donner un algorithme d'un coût $O(n^2)$ permettant de calculer la plus grande tour réalisable.

Exercice 12 Une triangulation d'un polygone convexe $P = [M_0, ..., M_{n-1}]$ est un ensemble de cordes qui ne se coupent pas à l'intérieur du polygone et qui le divisent en triangles. On va s'intéresser aux triangulations optimales, c'est-à-dire celles minimisant un poids défini comme la somme des poids des traigles, ceux-ci pouvant être définis de différentes façons (aire, périmètre, plus grand coté, etc...) : ce poids est une fonction $p(T)$ (où T est un triangle).

1. Montrer que toute triangulation d'un polygone à n cotés possède $(n-3)$ cordes et fait intervenir $n-2$ triangles.
2. Coût de l'algorithme naif ?
3. Pour $0 \leq i < j \leq n$, on note $T[i,j]$ comme le plus petit poids d'une triangulation optimale du polygone $[M_{i-1}, M_i, ..., M_j]$ (avec $M_n = M_0$).

   Proposer une relation simple récursive vérifiée par T, et en déduire un algorithme pour le problème de la triangulation optimale.
4. Comment trianguler de façon optimale si le poids d'un triangle est égal à son aire ?

4 Recherche de motifs

Pour les deux prochains exercices, on s'intéresse qu problème de recherche d'un motif dans un texte : on fixe ici deux mots $m = a_1 \dots a_k$ (motif) et $t = t_1 ... t_n$. L'objectif est de trouver la liste des i tels que $t_i ... t_{i+k-1} = m$.

Exercice 13 Donner le coût (en terme de comparaison de lettres) de l'algorithme naïf dans le pire des cas (on pourra rechercher $a^{n-1}b$ dans a^m).

Dans les deux exercices suivant on améliore cela en construisant un automate pertinent. La première construction (naïve) sera amélioré dans le second exercice.

Pour $w \in A^*$, $S_m(w)$ désigne le plus grand suffixe de w qui est un préfixe de m.

Exercice 14 On construit un automate déterministe complet $\mathcal{A}_m$ de la façon suivante : les états sont les préfixes de m (il y en a $k+1$; on note Q cet ensemble d'états) ; l'état initial est ε, et si $e \in Q$ et $\alpha \in A$, on définit $\delta(e, \alpha) = S_m(e\alpha)$.

1. Quel est le coût de la construction d'un tel automate ?
2. Construire $\mathcal{A}_m$ lorsque $m = aaaaab$, $m = abbbbb$, et $m = abcdabce$.
3. Que représente l'état dans lequel on se trouve après avoir lu un mot ? Conclure.

Exercice 15 On reprend les notations de l'exercice précédent, et on va construire l'automate $\mathcal{A}_m$ en temps $O(|A|\,|m|)$.

Soit $q_2 = q_1\alpha \in Q$ avec $q_1 \neq \varepsilon$. On suppose connu q_1' le plus grand suffixe STRICT de q_1 qui est préfixe de m.

1. Soit q_2' le plus grand suffixe STRICT de q_2 qui est préfixe de m. Montrer que $q_2' = \delta(q_1', \alpha)$.
2. Montrer que si $\beta \in A$, alors $\delta(q_2, \beta) = q_2\beta$ si $q_2\beta$ est un préfixe de m, et $\delta(q_2', \beta)$ sinon.
3. En déduire un algorithme plus efficace qu'à l'exercice précédent pour construire $\mathcal{A}_m$.
4. Reprendre la construction des automates de l'exercice précédent, avec cette nouvelle méthode.