Informatique I

La notation $\hat{=}$ désigne l'*égalité par définition*, pour la distinguer de l'égalité $=$.

On considérera que les entiers manipulés dans le problème sont codés en binaire, et sont donc des suites de bits.

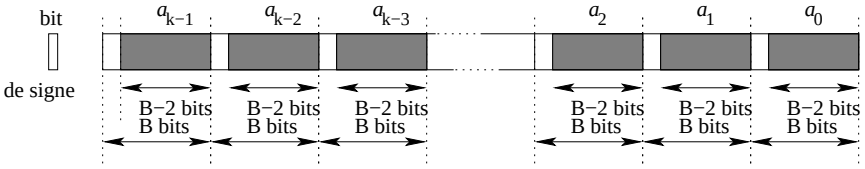
Les deux parties sont indépendantes. Notamment, tous les résultats de la partie 1 utilisés en partie 2 pourront y être admis librement.

1 Arithmétique en grande précision

On considère un modèle de machines travaillant sur des *entiers machine* à B bits (typiquement $B = 32$, mais on ne fera pas cette hypothèse). Un entier machine est donc un entier a tel que $0 \leq a < 2^B$.

Les opérations arithmétiques élémentaires (addition mod 2^B, soustraction mod 2^B, opposé mod 2^B, multiplication, division entière div et calcul du reste mod, le tout modulo 2^B) sont supposées opérer en temps constant. On supposera de plus qu'on dispose de fonctions d'allocation dynamique de mémoire opérant en temps constant.

On s'intéresse ici à des opérations sur des grands nombres, représentés sous forme de tableaux d'entiers machine. On supposera pour ceci que B est pair, $B \geq 4$, et que l'entier n est représenté sous forme d'un tableau d'entiers $< 2^{B-2}$, $a_0, \ldots, a_{k-1}$, tel que la valeur absolue $|n|$ égale $a_0 + a_1.2^{B-2} + \ldots + a_k.2^{k(B-2)}$, plus un bit de signe:



La *taille* de n est par définition k.

1. Montrer que l'on peut calculer la somme et la différence de grands nombres de tailles $\leq k$ en temps $O(k)$.

2. Proposer un algorithme en temps constant qui prend deux entiers machine n et n' plus petits que 2^{B-2}, et retourne leur produit sous forme de liste de deux entiers a_1 et a_2. Autrement dit, soient $0 \leq n, n' < 2^{B-2}$, calculer a_1 et a_2 tels que $0 \leq a_1, a_2 < 2^{B-2}$ et $nn' = a_1 + a_2 2^{B-2}$.

3. Montrer que l'on peut calculer le produit de deux grands nombres de tailles $\leq k$ en temps $O(k^\alpha)$, où α est une constante < 2 que l'on déterminera. (Indication : utiliser l'identité remarquable $(a2^\beta + b)(c2^\beta + d) = ac2^{2\beta} + ((a+b)(c+d) - ac - bd)2^\beta + bd$, et remarquer que le côté droit ne contient que 3 produits différents ac, bd et $(a+b)(c+d)$.)

4. On admettra dans la suite que l'on peut aussi calculer le quotient et le reste de la division de deux grands nombres de tailles $\leq k$ en temps $O(k^\alpha)$.

   Montrer qu'on peut calculer le produit de deux nombres n, n' tels que $0 \leq n, n' < p$ modulo un grand nombre p de taille $\leq k$, en temps $O(k^\alpha)$. (On demande donc en particulier que le résultat soit compris entre 0 inclus et p exclu.)

5. Montrer qu'on peut calculer la somme et la différence de deux nombres n, n' tels que $0 \leq n, n' < p$ modulo un grand nombre p de taille $\leq k$, en temps $O(k)$. (On demande encore que le résultat soit compris entre 0 inclus et p exclu. Noter que la complexité demandée est $O(k)$, et non $O(k^\alpha)$.)

6. On considère la fonction BEZOUT: $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ suivante:

```
1 fonction BEZOUT (n, n') {
2         si n' = 0
3           alors retourner (n, 1, 0)
4         sinon retourner (d, a, a')
5                         où (d, b, b') ≙ BEZOUT (n', n mod n'),
6                            a ≙ b',
7                            a' ≙ b − b'(n div n')
8         }
```

   Montrer que pour toute paire de deux grands nombres positifs n et n' non tous deux nuls tels que $n \geq n'$, BEZOUT (n, n') termine et retourne un triplet (d, a, a') tel que $an + a'n' = d$, où d est le pgcd de n et n'.

7. On admettra que BEZOUT (n, n') termine en temps $O(k^{(1+\alpha)})$. Proposer un algorithme MDIV qui prend trois grands nombres n, n', p tels que $0 \leq n, n' < p$ et n' est premier avec p, et retourne $n/n' \bmod p$, c'est-à-dire l'unique grand nombre m modulo p tel que $n = n'm$ modulo p. L'algorithme MDIV devra terminer en temps $O(k^{(1+\alpha)})$. Justifier.

2 Couplages parfaits

Un *graphe* désigne ici un graphe non orienté, c'est-à-dire un couple $G \hat{=} (V, E)$, où V est un ensemble fini de *sommets*, et E est un ensemble de paires $\{v_1, v_2\}$ de sommets appelées *arêtes*. Si $\{v_1, v_2\} \in E$, on dit que v_1 et v_2 sont *adjacents*, et qu'ils sont les *extrémités* de l'arête $\{v_1, v_2\}$.

Un *couplage* dans $G \hat{=} (V, E)$ est un ensemble $M \subseteq E$ tel qu'aucun sommet de V ne soit une extrémité de plus d'une arête dans M. Il est *parfait* si tout sommet de V est une extrémité d'exactement une arête dans M.

On supposera dans ce problème que les graphes G n'ont pas d'autoboucle, autrement dit un sommet n'y est jamais adjacent à lui-même.

On rappelle la formule de Cramer, pour toute matrice $A \hat{=} (a_{ij})_{1 \leq i,j \leq n}$:

$$\det A \quad = \sum_{\pi \text{ permutation de } \{1,\ldots,N\}} (-1)^{sign(\pi)} \prod_{i=1}^{N} a_{i\,\pi(i)} \tag{1}$$

Une *permutation* π de $\{1, \ldots, N\}$ est une bijection de $\{1, \ldots, N\}$ vers $\{1, \ldots, N\}$, et $sign(\pi)$ est la *signature* de π.

On rappelle aussi que A et sa transposée $A^T \hat{=} (a_{ji})_{1 \leq i,j \leq n}$ ont même déterminant, et que si $n \geq 1$ et A^{ij} est le *mineur* obtenu à partir de A en supprimant la $i$ième ligne et la $j$ième colonne alors :

$$\det A \quad = \quad a_{11} \det A^{11} - a_{12} \det A^{12} + a_{13} \det A^{13} - \ldots \pm a_{1n} \det A^{1n} \tag{2}$$

1. À tout graphe $G \hat{=} (V, E)$ on associe sa *matrice de Tutte* T_G. Numérotons $1, \ldots, N$ les sommets de V, et créons $N(N-1)/2$ variables x_{ij}, $1 \leq i < j \leq N$. Alors T_G est la matrice telle que :

$$(T_G)_{ij} \hat{=} \begin{cases} x_{ij} & \text{si } i, j \text{ adjacents et } i < j \\ -x_{ij} & \text{si } i, j \text{ adjacents et } i > j \\ 0 & \text{sinon} \end{cases}$$

   Montrer que :

$$\det T_G = \sum_{M \text{ couplage parfait de } G} \left(\prod_{\{i,j\} \in M, i<j} x_{ij}^2 \right)$$

2. Quel est le nombre maximal C_N de couplages parfaits dans un graphe à N sommets ?

3. On rappelle la formule de Stirling:
$$N! \sim N^N e^{-N} \sqrt{2\pi N}$$
lorsque N tend vers $+\infty$. En utilisant la question précédente, montrer:
$$C_N = O\left(N^{N/2} e^{-N/2} \sqrt{2}\right)$$
lorsque N tend vers $+\infty$.

4. Déduire de la question précédente qu'on ne peut pas calculer $\det T_G$ en temps polynomial en général.

5. Soit p un entier premier quelconque. Une *p-valuation* ρ est une application qui envoie chaque variable x_{ij} vers un entier $\rho(x_{ij})$ modulo p, autrement dit, dans le corps $\mathbb{Z}/p\mathbb{Z}$. On note $T_G(\rho)$ la matrice T_G où x_{ij} est remplacée par $\rho(x_{ij})$. Montrer qu'on peut calculer $\det T_G(\rho) \bmod p$ en temps polynomial en $\log p$ et la taille de $T_G(\rho)$. (On supposera — cf. partie I — que les opérations arithmétiques élémentaires, addition, soustraction, multiplication mod p prennent un temps $O(\log^\alpha p)$ et que la division mod p prend un temps $O(\log^{1+\alpha} p)$.)

6. Montrer le lemme suivant : pour tout entier p premier, pour tout polynôme $P(x)$ non nul à coefficients dans $\mathbb{Z}/p\mathbb{Z}$, à une variable, et de degré k, la proportion des éléments m dans $\mathbb{Z}/p\mathbb{Z}$ tels que $P(m) = 0 \bmod p$ est au plus k/p.

7. En déduire le lemme de Schwartz-Zippel, qui est la généralisation de la question précédente aux polynômes en un nombre arbitraire de variables : pour tout entier p premier, pour tout polynôme $P(x_1, \ldots, x_n)$ non nul à coefficients dans $\mathbb{Z}/p\mathbb{Z}$ sur les variables $x_1, \ldots, x_n$ ($n \geq 1$) de degré total k, la proportion des n-uplets $m_1, \ldots, m_n$ dans $(\mathbb{Z}/p\mathbb{Z})^n$ tels que $P(m_1, \ldots, m_n) = 0 \bmod p$ est au plus k/p.

   On rappelle que si $P(x_1, \ldots, x_n)$ s'écrit $\sum_{e_1,\ldots,e_n} a_{e_1,\ldots,e_n} x_1^{e_1} \ldots x_n^{e_n}$, le degré total de P est le maximum des $e_1 + \ldots + e_n$ apparaissant dans la sommation avec $a_{e_1,\ldots,e_n} \neq 0 \bmod p$.

8. En utilisant les questions précédentes, donner un majorant de la probabilité que $\det T_G(\rho) = 0 \bmod p$ lorsque $\det T_G \neq 0$.

9. Par la question 4, il est difficile de tester si $\det T_G$ est le polynôme nul ou non. Le but de cette question est de concevoir un algorithme qui, au lieu de tester "$\det T_G = 0$?", va tester des questions plus simples de la forme "$\det T_G(\rho) = 0 \bmod p$?", et en déduire la réponse à la question "$\det T_G = 0$?" avec forte probabilité.

   On supposera que l'on dispose d'une procédure RAND telle que pour tout grand nombre n de taille au plus k, RAND (n) retourne un nombre aléatoire m tel que $1 \leq m < n$, en temps $O(k)$.

   On supposera d'autre part que l'on dispose d'une fonction RAND-PREMIER telle que pour tout grand nombre n de taille au plus k, RAND-PREMIER (K, n) retourne en temps $O(Kk^\beta)$ un entier p tel que $n \leq p \leq 2n$, qui est premier avec probabilité au moins $1 - 1/2^K$. Ici β est une constante telle que $\beta \geq \alpha$. De plus, le paramètre K est un entier machine.

   Déduire des questions précédentes un algorithme TEST (K, G) testant si G a un couplage parfait en temps proportionnel à K et polynomial en la taille de G, retournant OUI si G a un couplage parfait avec probabilité $\geq 1 - 1/2^K$, et retournant NON si G n'a pas de couplage parfait (avec probabilité 1).

10. En déduire un algorithme construisant un couplage parfait de G ou retournant NON, en temps polynomial en K' et la taille de G, et ne se trompant qu'avec probabilité $\leq 1/2^{K'}$. On dira que l'algorithme *se trompe* s'il retourne NON alors que G a un couplage parfait. Justifier.