Informatique I

Correction.

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.

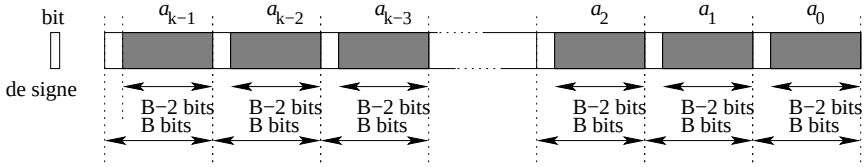
Les deux parties sont indépendantes. Notamment, tous les résultats de la partie 1 utilisés en partie 2 pourront y être admis librement.

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 $O(k)$.

   Supposons n_1 et n_2 positifs ou nuls, et représentés par les tableaux $a_{10}, \ldots, a_{1(k-1)}$ et $a_{20}, \ldots, a_{2(k-1)}$ respectivement. On calcule leur somme en calculant plus généralement $n_1 + n_2 + c$ par l'algorithme suivant PLUS (n_1, n_2, c)*, où c est la* retenue*. Soit $a_0, a_1, \ldots, a_{k-1}, a_k$ le résultat : posons $c_0 \hat{=} c$, puis pour tout i, $0 \leq i < k$, posons $b_i \hat{=} a_{1i} + b_{1i} + c_i$, et si $b_i < 2^{B-2}$ alors $a_i \hat{=} b_i$, $c_{i+1} \hat{=} 0$; sinon, $a_i \hat{=} b_i - 2^{B-2}$, $c_{i+1} \hat{=} 1$. Finalement, on pose $a_k \hat{=} c_k$.*

   De même, la soustraction se calcule en calculant $n_1 - n_2 - c$ par l'algorithme MOINS (n_1, n_2, c) *suivant.* MOINS (n_1, n_2, c) *retourne $a_0, a_1, \ldots, a_{k-1}$ et une retenue c_k telle que $n_1 - n_2 - c = a_0 + a_1.2^{B-2} + \ldots + a_{k-1}.2^{(k-1)(B-2)} - c_k 2^{k(B-2)}$. Il suffit de poser $c_0 \hat{=} c$, ensuite pour tout i, $0 \leq i < k$, $b_i \hat{=} a_{1i} - a_{2i} - c_i$, puis si $i \geq 0$, $a_i \hat{=} b_i$ et $c_{i+1} \hat{=} 0$; sinon, $a_i \hat{=} b_i + 2^{B-2}$ et $c_{i+1} \hat{=} 1$.*

   Dans le cas $c_k = 1$, on peut convertir $a_0 + a_1.2^{B-2} + \ldots + a_{k-1}.2^{(k-1)(B-2)} - c_k 2^{k(B-2)}$ en un entier dans le format prescrit, en utilisant la fonction MNORM $(a_0, a_1, \ldots, a_{k-1})$ *décrite ci-après. Soit i_0 le plus grand entier $\leq k$ tel que tous les a_i, $i < i_0$, sont nuls. Il suffit de calculer la suite*

$a'_0, \ldots, a'_{k-1}$ *telle que :* $a'_0 \hat{=} \ldots \hat{=} a'_{i_0-1} \hat{=} 0$, *et si* $i_0 < k$ *alors* $a'_{i_0} \hat{=} 2^{B-2} - a_{i_0}$ *(qui est entre* 0 *et* $2^{B-2}-1$ *car* a_{i_0} *est non nul), et* $a'_i \hat{=} 2^{B-2} - 1 - a_i$ *pour tout* i, $i_0 < i < k$. *L'entier* $n_1 - n_2 - c$ *est alors représenté avec le signe* $-$ *et la suite* $a'_0, \ldots, a'_{k-1}$.

Pour calculer l'addition de deux entiers quelconques n_1 *et* n_2, *soit ils ont le même signe* s, *et on calcule* PLUS $(|n_1|, |n_2|, 0)$ *avec le signe* s; *soit ils sont de signe opposé, et posant* s *le signe de* n_1, *on appelle* MOINS $(|n_1|, |n_2|, 0)$, *qui retourne* $a_0, a_1, \ldots, a_{k-1}, c_k$ *: si* $c_k = 0$, *alors le résultat est l'entier de signe* s *et représenté par* $a_0, a_1, \ldots, a_{k-1}$, *sinon c'est l'entier de signe* $-s$ *et représenté par la liste* MNORM $(a_0, a_1, \ldots, a_{k-1})$.

Ces algorithmes sont composées de trois parties, PLUS, MOINS, *et* MNORM, *qui prennent toutes un temps linéaire en* k.

Il y a aussi d'autres solutions possibles.

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 $nn' = a_1 + a_2 2^{B-2}$.

Calculer $n_1 \hat{=} n \bmod 2^{(B-2)/2}$, $n_2 \hat{=} n \operatorname{div} 2^{(B-2)/2}$, *et* $n'_1 \hat{=} n' \bmod 2^{(B-2)/2}$, $n'_2 \hat{=} n' \operatorname{div} 2^{(B-2)/2}$. *C'est possible car* B *est pair. Alors* $nn' = n_1 n'_1 + (n_1 n'_2 + n'_1 n_2) 2^{(B-2)/2} + n'_1 n'_2 2^{B-2}$. *Écrivons* nn' *sous la forme* $b_1 + b_2 2^{(B-2)/2} + b_3 2^{B-2}$, *avec* $b_1, b_2 < 2^{(B-2)/2}$.

Alors on a $b_1 \hat{=} n_1 n'_1 \bmod 2^{(B-2)/2}$. *Ceci se calcule sans débordement arithmétique, car* $n_1 n'_1 < 2^{B-2}$. *Donc* $b_2 2^{(B-2)/2} + b_3 2^{B-2} = (n_1 n'_2 + n'_1 n_2 + n_1 n'_1 \operatorname{div} 2^{(B-2)/2}) 2^{(B-2)/2} + n'_1 n'_2 2^{B-2}$.

Posons $c_2 \hat{=} n_1 n'_2 + n'_1 n_2 + n_1 n'_1 \operatorname{div} 2^{(B-2)/2}$. *Il s'ensuit que* $b_2 \hat{=} c_2 \bmod 2^{(B-2)/2}$. *Encore une fois,* c_2 *se calcule sans débordement arithmétique, étant* $< 2^{B-2} + 2^{B-2} + 2^{(B-2)/2} \leq 5.2^{(B-2)/2} < 2^B$. *D'autre part, ceci implique que* $b_3 = (n'_1 n'_2 + c_2 \operatorname{div} 2^{(B-2)/2})$.

Donc finalement $a_1 \hat{=} b_1 + b_2 2^{(B-2)/2}$ *(ce qui s'effectue sans débordement, puisque ceci est* $< 2^{(B-2)/2} + 2^{(B-2)}$*) et* $a_2 \hat{=} b_3$.

Le nombre d'opérations élémentaires, finalement, est constant, comme demandé.

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 $(a+b)(c+d)$.)

Ignorons les questions de signes pour l'instant. On a donc à multiplier deux nombres représentés sous forme de listes de k *coefficients* $< 2^{B-2}$. *Sans perte de généralité, supposons* $k = 2^K$, *et effectuons notre multiplication de façon récursive.*

Si $K = 0$, *la multiplication à effectuer est une multiplication d'entiers machine, qui prend un temps constant par la question précédente.*

Si $K \geq 1$, *coupons les nombres à multiplier en deux listes de même longueur* 2^{K-1}. *Les nombres en argument s'écrivent* $n_{11} 2^\beta + n_{12}$ *et* $n_{21} 2^\beta + n_{22}$, *où* $\beta \hat{=} 2^{K-1}(B-2)$. *On peut donc calculer* $n'_1 \hat{=} n_{11} n_{21}$ *(en appelant* MULT *récursivement),* $n'_2 \hat{=} n_{12} n_{22}$ *(en appelant encore* MULT *récursivement),* $n'_3 \hat{=} (n_{11} + n_{12})(n_{21} + n_{22}) - n'_1 - n'_2$ *(en appelant une troisième fois* MULT *récursivement). Alors* MULT *retourne* $n'_1 2^{2^K (B-2)} + n'_3 2^{2^{K-1}(B-2)} + n'_2$, *ce qui est facile à calculer sachant qu'on a déjà* PLUS *et* MOINS, *et que la multiplication par* $2^{2^K(B-2)}$ *consiste à ajouter* 2^K *zéros en tête de liste, et que la multiplication par* $2^{2^{K-1}(B-2)}$ *consiste à ajouter partie* 2^{K-1} *zéros en tête de liste.*

Soit t_K *le temps pris par cette procédure. Pour* $K = 0$, t_K *est une constante, disons* 1 *sans perte de généralité. Pour* $K \geq 1$, $t_K = 3t_{K-1} + \gamma 2^K$, *où* γ *est une constante.*

Donc :

$$\begin{aligned} t_K &= 3t_{K-1} + \gamma 2^K \\ &= 9t_{K-2} + 3\gamma 2^{K-1} + \gamma 2^K \end{aligned}$$

$$
\begin{aligned}
&= \ldots \\
&= 3^K + \gamma(2^K + 3.2^{K-1} + \ldots + 3^K.2^0) \\
&= 3^K + \gamma 3^K(1 + 2/3 + (2/3)^2 + \ldots + (2/3)^K) \\
&\leq 3^K + \gamma 3^{K+1} \\
&= O(3^K)
\end{aligned}
$$

Donc le temps pris par cet algorithme pour multiplier deux nombres de taille k est $O(3^K)$, où K est la partie entière supérieure de $\log_2 k$, qui est $\leq \log_2 k + 1$. Donc ce temps est $O(3^{\log_2 k}) = O(k^{\log_2 3})$. On peut donc prendre $\alpha \hat{=} \log_2 3$. Noter que $\alpha \sim 1,58$ est inférieur strictement à 2 (en effet, $\alpha < \log_2 4 = 2$).

Finalement, le cas des entiers avec signe se traite en multipliant les listes de coefficients comme ci-dessus, et en prenant comme signe du résultat, soit + si les facteurs sont de même signe, soit − sinon.

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.)

   Il suffit de calculer nn' en temps $O(k^\alpha)$, puis de calculer le reste par p, ce qui prend encore un temps $O(k^\alpha)$ par hypothèse.

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 $O(k^\alpha)$.)

   L'astuce est qu'on n'a pas besoin de prendre le reste modulo p. Dans le cas de la somme, si $n + n' < p$, alors $n + n'$ est directement la somme modulaire, sinon $n + n' - p$ est le résultat cherché car par hypothèse $n + n' - p \geq 0$, et $n + n' - p \leq (p-1) + (p-1) - p < p$. Dans le cas de la différence, soit $n \geq n'$ et $n - n'$ est clairement le résultat souhaité, soit $n < n'$ et alors $p + n - n'$ l'est: $p + n - n' < p$ par construction, et $p + n - n' > p + n - p = n \geq 0$.

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'.

   La terminaison est par récurrence complète sur n'. Comme $n \bmod n' < n'$, tout appel récursif de BEZOUT *voit son deuxième argument décroître. De plus, l'invariant selon lequel le premier argument est supérieur ou égal au deuxième argument est préservé à la ligne 5, car $n' \geq n \bmod n'$ pour tout $n \geq 0$, $n' > 0$.*

   On montre la propriété $an + a'n' = d$ aussi par récurrence complète sur n'. Si $n' = 0$ (ligne 3), $d = n$, $a = 1$, $a' = 0$, alors $an + a'n' = d$ est clair, de même que le fait que d soit le pgcd de n

et n'. Sinon (lignes 4–7), le pgcd d de n' et de $n \bmod n'$ est aussi celui de n et de n'. Il reste à vérifier que $an + a'n' = d$, où $a = b'$, $a' = b - b'(n \operatorname{div} n')$ et $bn' + b'(n \bmod n') = d$:

$$\begin{aligned} an + a'n' &= b'n + (b - b'(n \operatorname{div} n'))n' \\ &= b'n + bn' - b'(n \operatorname{div} n')n' \\ &= bn' + b'(n - (n \operatorname{div} n')n') \\ &= bn' + b'(n \bmod n') \\ &= d \end{aligned}$$

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.

   Comme n' est premier avec p, BEZOUT (p, n') *retourne un triplet $(1, a, a')$ avec $ap + a'n' = 1$ en temps $O(k^{(1+\alpha)})$. En particulier, a' est l'inverse de n' modulo p. Il ne reste plus qu'à calculer $m \hat{=} (a' \bmod p)n \bmod p$ en temps $O(k^{\alpha})$, et à retourner m.*

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 $A \hat{=} (a_{ij})_{1 \leq i,j \leq n}$:

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

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 :

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

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 :

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

On le montre par récurrence sur N.

Si $N = 0$, alors il y a exactement un couplage parfait dans G, l'ensemble vide d'arêtes. D'autre part, le déterminant de la matrice vide vaut la somme sur une permutation (vide, identité) d'un produit de 0 élément, et vaut donc 1 aussi.

Sinon :

$$\det T_G = \sum_{\{1,j\}\in E} (-1)^{j-1} x_{1j} \det T_G^{1j}$$

Notons que $j = 0$ n'apparaît pas dans la somme, puisque G n'a pas d'autoboucle. Donc T_G^{1j} a comme première colonne la première colonne de T_G, moins son premier élément $T_{G\,11} = 0$. On calcule donc $\det T_G^{1j}$ en calculant $\det \left(T_G^{1j}\right)^T$ par la formule (2) :

$$\det T_G^{1j} = \sum_{\{i,1\}\in E} (-1)^i (-x_{1i}) \det T_G^{(1,i)(1,j)}$$

où $T_G^{(1,i)(1,j)}$ est la matrice T_G de laquelle on a enlevé les lignes 1 et i (et non $i-1$), et les colonnes 1 et j. Donc :

$$\det T_G = \sum_{\{1,j\},\{1,i\}\in E} (-1)^{i+j} x_{1i} x_{1j} \det T_G^{(1,i)(1,j)}$$

Découpons cette somme en trois parties, celle où $i = j$, celle où $i < j$ et celle où $i > j$:

$$\begin{aligned}
\det T_G &= \sum_{\{1,i\}\in E} x_{1i}^2 \det T_G^{(1,i)(1,i)} \\
&\quad + \sum_{\{1,i\},\{1,j\}\in E, i<j} (-1)^{i+j} x_{1i} x_{1j} \det T_G^{(1,i)(1,j)} \\
&\quad + \sum_{\{1,i\},\{1,j\}\in E, i>j} (-1)^{i+j} x_{1i} x_{1j} \det T_G^{(1,i)(1,j)} \\
&= \sum_{\{1,i\}\in E} x_{1i}^2 \det T_G^{(1,i)(1,i)} \\
&\quad + \sum_{\{1,i\},\{1,j\}\in E, i<j} (-1)^{i+j} x_{1i} x_{1j} \det T_G^{(1,i)(1,j)} \\
&\quad + \sum_{\{1,i\},\{1,j\}\in E, i<j} (-1)^{i+j} x_{1i} x_{1j} \det T_G^{(1,j)(1,i)}
\end{aligned}$$

où on a échangé les indices i et j dans la dernière ligne. Mais la transposée de $T_G^{(1,j)(1,i)}$ est $(T_G^T)^{(1,i),(1,j)}$, et comme T_G^T est l'opposée $-T_G$ de T_G par construction, la dernière ligne est l'opposée de l'avant-dernière. Donc :

$$\det T_G = \sum_{\{1,i\}\in E} x_{1i}^2 \det T_G^{(1,i)(1,i)}$$

Or $T_G^{(1,i)(1,i)}$ est $T_{G'}$, où G' est le graphe G duquel on a supprimé les sommets 1, j ainsi que toutes les arêtes adjacentes.

Mais les couplages parfaits M de G contiennent exactement une arête $\{1,i\} \in E$, pour un certain i, et M privé de $\{1,i\}$ est nécessairement un couplage parfait M' de G'. Comme par récurrence $\det T_G^{(1,i)(1,i)}$ est la somme des $\prod_{\{i',j'\}\in M', i'<j'} x_{i'j'}^2$, le résultat s'ensuit.

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

Il est clair que $C_N = 0$ si N est impair. Supposons donc N pair.

Si $N = 0$, il y a exactement un couplage parfait, donc $C_0 = 1$. Sinon, les couplages parfaits sont donnés par une paire $(1, j)$, $j \in \{2, \ldots, N\}$ ($N-1$ choix) et un couplage parfait dans le graphe à $N-2$ sommets restants, donc $C_N = (N-1)C_{N-2}$. En conséquence, $C_N = (N-1).(N-3)\ldots 3.1 = N!/(2^{N/2}.(N/2)!)$ si N est pair.

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 $+\infty$.

Par la formule de Stirling, lorsque N est impair:
$$\begin{aligned} C_N &\sim \frac{N^N e^{-N} \sqrt{2\pi N}}{2^{N/2} (N/2)^{N/2} e^{-N/2} \sqrt{2\pi N/2}} \\ &= N^{N/2} e^{-N/2} \sqrt{2} \end{aligned}$$

On peut aussi répondre à la question sans connaître la formule de Stirling. Par exemple, remarquons que dans le produit $(N-1).(N-3)\ldots 3.1$, on peut minorer tous les $N-2k+1$ pour $k < N/3$ par $N/3+1$, et les autres par 1, ce qui donne :
$$C_N \geq \left(\frac{N}{3} + 1\right)^{N/3}$$
ce qui n'est pas non plus borné supérieurement par aucun polynôme.

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

La taille de $\det T_G$ étant un $\Omega(C_N)$ (autrement dit, le rapport de $\det T_G$ sur C_N étant borné inférieurement par une constant positive lorsque N tend vers $+\infty$), l'expression du polynôme $\det T_G$ requiert une place en $\Omega(N^{N/2} e^{-N/2})$, et n'est donc bornée supérieurement par aucun polynôme. Noter qu'on n'utilise pas seulement que $C_N = O\left(N^{N/2} e^{-N/2} \sqrt{2}\right)$, mais en fait que $C_N = \Omega\left(N^{N/2} e^{-N/2} \sqrt{2}\right)$.
Le calcul de $\det T_G$ doit nécessairement construire l'expression $\det T_G$ en mémoire, ce qui demande un espace non polynomial, donc un temps pour le construire non polynomial non plus. En effet, dans tout modèle de machine raisonnable, le temps nécessaire pour créer une structure de donnée de taille k est d'au moins k opérations.

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 $O(\log^{1+\alpha} p)$.)

$T_G(\rho)$ est une matrice $N \times N$ à coefficients dans $\mathbb{Z}/p\mathbb{Z}$. La méthode de Gauss permet de mettre $T_G(\rho)$ en forme triangulaire en $O(N^3)$ opérations élémentaires, chacune prenant un temps $O(\log^{1+\alpha} p)$ par hypothèse. Le déterminant se calcule ensuite comme produit des termes sur la diagonale, en temps $O(N \log^\alpha p)$, donc le temps total du calcul du déterminant est $O(N^3 \log^{1+\alpha} p)$.

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 $P(m) = 0 \mod p$ est au plus k/p.

   Supposons par contradiction que P est de degré k, mais il y a au moins $k+1$ racines de P, disons $m^1, \ldots, m^{k+1}$. Notons que la donnée de $k+1$ tels couples (i, m^i) suffit pour déterminer P de façon unique, par exemple comme polynôme d'interpolation de LAGRANGE. *Mais le polynôme nul est aussi, trivialement, un polynôme qui à chaque m^i associe 0. Comme le polynôme d'interpolation est unique, $P = 0$, contradiction. Il s'ensuit que P ne peut pas avoir plus de k racines, donc la proportion des racines est d'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 $P(m_1, \ldots, m_n) = 0 \mod p$ 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 $a_{e_1,\ldots,e_n} \neq 0 \mod p$.

   On raisonne par récurrence sur n. Le cas $n = 1$ est la question précédente. Supposons dans le cas récurrent que le résultat est vrai pour n, et montrons-le pour $n+1$. Soit P un polynôme quelconque sur $n+1$ variables $x_1, \ldots, x_n, x_{n+1}$. Pour chaque valeur m_{n+1} de x_{n+1}, $P(x_1, \ldots, x_n, m_{n+1})$ est un polynôme à n variables, qui vaut soit identiquement 0 soit n'a pas plus de $k'/p.p^n = k'p^{n-1}$ racines, où k' est le degré total de $P(x_1, \ldots, x_n, m_{n+1})$.

   On ne peut pas conclure directement, comptons le nombre de valeurs m_{n+1} pour lesquelles $P(x_1, \ldots, x_n, m_{n+1})$ est le polynôme nul. Pour chacune, le polynôme $x_{n+1} - m_{n+1}$ divise $P(x_1, \ldots, x_n, x_{n+1})$. En effet, c'est une conséquence directe de l'identité de Taylor :

$$P(x_1, \ldots, x_n, x_{n+1}) = P(x_1, \ldots, x_n, m_{n+1}) + \sum_{j=1}^{+\infty} \frac{(x_{n+1} - m_{n+1})^j}{j!} \frac{\partial^j P}{\partial x_{n+1}^j}(x_1, \ldots, x_n, m_{n+1})$$

   (où la somme est finie, puisque P est un polynôme). Donc $P(x_1, \ldots, x_n, x_{n+1})$ s'écrit sous forme d'un produit de q facteurs de la forme $(x_{n+1} - m)^{q_m}$, où $q_m \geq 1$, et les m sont distincts deux à deux, et d'un polynôme $Q(x_1, \ldots, x_n, x_{n+1})$ tel que $Q(x_1, \ldots, x_n, m_{n+1})$ n'est identiquement 0 pour aucune valeur m_{n+1}. Le degré total de Q est d'au plus $k - \sum_m q_m$. Par récurrence, le nombre de racines de Q est d'au plus $(k - \sum_m q_m)p^{n-1} \leq (k-q)p^{n-1}$. Et les racines de P sont ou bien des racines de Q ou bien des $(x_1, \ldots, x_n, m)$, où il y a q valeurs de m possibles et p^{n-1} valeurs de $(x_1, \ldots, x_n)$. En sommant les deux contributions, il y a donc au plus $(k-q)p^{n-1} + qp^{n-1} = kp^{n-1}$ racines de P, d'où le résultat.

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

   Par la question 1, $\det T_G$ est un polynôme à $N(N-1)/2$ variables (au plus) de degré total $\leq N$. Si $\det T_G \neq 0$, par la question précédente la proportion des p-valuations sur $N(N-1)/2$ variables telles que $\det T_G(\rho) = 0 \mod p$ est $\leq N/p$. C'est la probabilité cherchée.

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 "$\det T_G(\rho) = 0 \mod p$?", 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).

On tire un nombre premier $p \geq 2N$ en utilisant RAND-PREMIER*, avec un paramètre K' que l'on fixera plus loin. On tire alors au hasard une p-valuation ρ en utilisant* RAND*. Si $\det T_G(\rho) = 0 \mod p$, alors on retourne* OUI*, sinon on retourne* NON*. La réponse* NON *est sûre, et la réponse* OUI *n'est erronée qu'avec probabilité $\leq N/p \leq 1/2$ dès que p est premier, ce qui arrive avec probabilité $\geq 1 - 1/2^{K'}$. La probabilité d'erreur totale est donc au plus $1/2 + 1/2^{K'}$.*
Si la probabilité d'erreur était de $1/2$, il suffirait de répéter cette procédure K fois, et on obtiendrait la probabilité désirée. À cause du terme d'erreur $1/2^{K'}$, on va répéter $K+1$ fois, ce qui mène à une probabilité d'erreur de:

$$\begin{aligned}\left(1/2 + 1/2^{K'}\right)^{K+1} &= e^{(K+1)(\log(1/2)+\log(1+1/2^{K'-1}))} \\ &\leq e^{(K+1)(\log(1/2)+1/2^{K'-1})} \\ &= e^{K\log(1/2)+\log(1/2)+(K+1)/2^{K'-1}} \\ &\leq e^{K\log(1/2)} = 1/2^K\end{aligned}$$

dès que $\log(1/2) + (K+1)/2^{K'-1} \leq 0$, c'est-à-dire $K' \geq 1 + \log(K+1)/\log 2 - \log\log 2/\log 2$.
Comme chaque itération prend un temps polynomial en les tailles des paramètres d'entrée, la procédure finale est bien en temps polynomial.

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.

Si le nombre N de sommets de G est impair, retourner NON*. Ceci prend un temps polynomial à vérifier, dans la plupart des représentations standard choisies pour G.*
Sinon, on définit un algorithme récursif COUPLAGE (K, G) *comme suit. Si G est vide, retourner le couplage vide. Sinon, chercher un sommet i adjacent au sommet 1 tel que* TEST (K, G_{1i}) *retourne* OUI*, où G_{1i} est le graphe obtenu à partir de G en supprimant les sommets 1, i, et toutes les arêtes adjacentes. Si un tel i n'existe pas, alors retourner* NON*. Sinon, retourner* COUPLAGE (K, G_{1j}) *union l'arête $\{1, j\}$.*
L'algorithme termine en au plus $N/2$ appels récursifs, où $G \hat{=} (V, E)$ et $N \hat{=} |V|$. Si on prend pour G une représentation par listes d'adjacence ou par matrice d'adjacence, le parcours des sommets adjacents à 1 se fait en $O(|E|)$ itérations, la construction de G_{1i} prend un temps linéaire en la taille $|V| + |E|$ de G (on peut améliorer ceci en représentant les sous-graphes de G par G lui-même, plus une liste, ou un arbre équilibré de sommets supprimés), et TEST (K, G_{1j}) *prend un temps $O(KN^3 \log^\alpha N)$. Donc l'algorithme prend un temps $O(K(|V| + |E|)|E||V|^4 \log^\alpha |V|)$, qui est polynomial en la taille de G.*
Il est clair que si COUPLAGE (K, G) *retourne un ensemble d'arêtes, alors c'est un couplage, et qu'il est parfait.*
Estimons la probabilité de se tromper, c'est-à-dire que COUPLAGE (K, G) *réponde* NON *alors que G a un couplage parfait. Ceci arrive lorsque* TEST (K, G_{1i}) *retourne* OUI *alors qu'il n'y a aucun couplage parfait contenant l'arête $\{1, i\}$, ou bien lorsque* TEST (K, G_{1i}) *retourne* OUI*, il y a un couplage parfait contenant $\{1, i\}$, et* COUPLAGE (K, G_{1i}) *retourne* NON*. Ceci arrive avec probabilité p_{NK} telle que $p_{0K} = 0$, et $p_{NK} \leq 1/2^K + (1 - 1/2^K)p_{(N-2)K}$ si $N \geq 1$, autrement dit $p_{NK} \leq N/2^{K+1}$.*
Pour que la probabilité de se tromper soit $\leq 1/2^{K'}$, il suffit donc que $K \geq K' + \log_2 N - 1$. Notons que ceci est encore calculable en temps polynomial en la taille $\log_2 K'$ de K' (en binaire) et en la taille $\log_2 N$ de N.