ÉCOLE NORMALE SUPÉRIEURE DE LYON

Concours d'admission session 2015

Filière universitaire : Second concours

COMPOSITION D'INFORMATIQUE

Durée : 3 heures

L'usage des calculatrices de poche est autorisé, y compris les calculatrices programmables et alphanumériques ou à écran graphique, à condition que leur fonctionnement soit autonome et qu'il ne soit pas fait usage d'imprimante.

⋆ ⋆ ⋆

Ce sujet comporte trois parties indépendantes. Il est recommandé de lire l'ensemble du sujet avant de commencer la rédaction. Il est également conseillé de traiter les questions dans l'ordre de l'énoncé. On pourra cependant aborder une question en admettant les résultats des questions précédentes. Les algorithmes demandés seront écrits dans un langage de programmation au choix du candidat.

Définitions et notations

Dans ce sujet, on s'intéresse à la gestion de matrices. On note $A_{i,j}$ l'élément situé sur la $i^{\text{ème}}$ ligne et la $j^{\text{ème}}$ colonne de la matrice A. On considère que tous les indices commencent à 0 (donc l'élément en haut à gauche de la matrice A est $A_{0,0}$ et celui en bas à droite est $A_{n-1,n-1}$). Toutes les matrices considérées seront de taille $n \times n$ (n lignes et n colonnes).

On considère dans ce sujet des matrices particulières, appelées *matrices creuses*, dans lesquelles la plupart des éléments sont nuls. On rencontre souvent de telles matrices en pratique, et si on y prend garde, elles nécessitent beaucoup moins de calcul que des matrices pleines pour de nombreuses opérations. Dans ce sujet, on va concevoir des structures de données et des algorithmes spécifiques permettant de réduire la complexité de certaines opérations sur ces matrices. Ces algorithmes pourront traiter n'importe quelles matrices (creuses ou pleines), mais permettront des gains importants de complexité avec les matrices contenant beaucoup de zéros.

Pour analyser la complexité des algorithmes (ainsi que la taille des données) on utilisera d'une part la taille n de la matrice et d'autre part le nombre d'éléments non-nuls : on note $nnz(A)$ le nombre d'éléments non-nuls (ou *non-zéros*) de la matrice A. En général, $nnz(A)$ est beaucoup plus petit que n^2 : on préférera donc utiliser $nnz(A)$ à la place de n^2 dans les analyses de complexité lorsque cela est possible.

Dans la description des algorithmes, nous utiliserons la notation $a \leftarrow b$ pour l'affectation de la valeur de b à la variable a. Pour un tableau T contenant m éléments, on notera $T[i]$ le $i^{\text{ème}}$ élément de T pour i allant de 0 à $m-1$.

Partie 1 Représentations et opérations de base

Pour éviter de stocker tous les éléments non nuls d'une matrice creuse, une solution simple consiste à représenter une matrice sous la forme de trois tableaux de taille $nnz(A)$ chacun : deux tableaux `ligne` et `colonne` contenant les indices des éléments non-nuls, et un tableau `valeur` contenant leur valeur, tels que $A_{\texttt{ligne}[k],\texttt{colonne}[k]} = \texttt{valeur}[k]$. On impose que `colonne` est trié par ordre croissant, ainsi que les portions de `ligne` correspondant à une même colonne. On appelle ce format la **représentation simple triée** de la matrice.

Voici un exemple de matrice 4×4 (avec les indices de lignes et colonnes) et sa représentation simple triée.

$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 45 & 0 & 32 & 0 \\ 31 & 0 & 0 & 9 \\ 0 & 17 & 30 & 0 \\ 35 & 4 & 0 & 10 \end{pmatrix} \end{array}$$

```
colonne = [ 0,  0,  0,  1,  1,  2,  2,  3,  3]
ligne   = [ 0,  1,  3,  2,  3,  0,  2,  1,  3]
valeur  = [45, 31, 35, 17,  4, 32, 30,  9, 10]
```

Question 1. À partir d'une matrice de taille $n \times n$ donnée sous la forme d'un tableau à deux dimensions, tel que `A[i][j]` contient l'élément $A_{i,j}$, donner un algorithme qui construit sa représentation simple triée. On précisera la complexité de l'algorithme en fonction de n et $nnz(A)$.

Pour diminuer encore la taille de la représentation d'une matrice creuse, on introduit la **représentation compacte** qui comporte :

— un tableau `colnnz` de taille $n+1$, tel que `colnnz`$[0] = 0$ et pour $1 \leqslant j \leqslant n$, `colnnz`$[j]$ est le nombre d'éléments non-nuls dans les colonnes 0 à $j-1$ de A ;

— les tableaux `ligne` et `valeur` identiques à ceux de la représentation simple triée.

Voici la représentation compacte de la matrice A de l'exemple précédent (les espaces supplémentaires dans les tableaux ci-dessous sont uniquement destinés à aider la compréhension).

```
colnnz = [ 0,          3,      5,      7,     9]
ligne  = [ 0,  1,  3,  2,  3,  0,  2,  1,  3]
valeur = [45, 31, 35, 17,  4, 32, 30,  9, 10]
```

Question 2. On considère une matrice A en représentation compacte. Étant donnés k et l tels que $0 \leqslant k \leqslant n-1$ et $0 \leqslant l \leqslant$ `colnnz`$[k+1]-$`colnnz`$[k]-1$, que vaut `valeur`[`colnnz`$[k]+l$] ?

Question 3. Comment modifier l'algorithme de la question 1 pour construire la représentation compacte d'une matrice donnée sous la forme d'un tableau à deux dimensions ?

Dans la suite de ce sujet, nous considérerons que **toutes les matrices sont données sous la forme de leur représentation compacte**.

⋆ ⋆ ⋆

Dans la suite du sujet, on cherche à prédire la position ou le nombre des éléments non-nuls d'une matrice résultat d'une opération, en se basant uniquement sur la position des éléments non-nuls des opérandes. Par exemple, considérons l'opération $C \leftarrow A + B$. Étant donnée la position des éléments non-nuls de A et B, on dit qu'un élément $C_{i,j}$ est *essentiellement nul* s'il est égal à zéro quelles que soit les valeurs des éléments non-nuls de A et B. Par exemple, pour les matrices suivantes (où les X représentent les éléments non nuls) :

$$A = \begin{pmatrix} 0 & X \\ 0 & X \end{pmatrix} \qquad B = \begin{pmatrix} 0 & X \\ X & 0 \end{pmatrix}$$

$C_{0,0}$ est le seul élément essentiellement nul, même si d'autres éléments de C peuvent s'annuler en fonction des valeurs des éléments non-nuls de A et B. Dans la suite du sujet, on ne s'intéressera pas aux valeurs des éléments non essentiellement nuls, et on notera $C_{i,j} = 0$ (respectivement $C_{i,j} \neq 0$) pour dire que $C_{i,j}$ est (resp. n'est pas) essentiellement nul.

Question 4. Écrire un algorithme qui, étant données deux matrices A et B, calcule le nombre d'éléments non essentiellement nuls dans la somme de matrices $A + B$ et dont la complexité est $O(nnz(A) + nnz(B))$.

Question 5. On considère le produit de matrices $C = AB$ où A , B et C sont trois matrices carrées de taille $n \times n$. On note $\mathcal{A}_j$, $\mathcal{B}_j$ et $\mathcal{C}_j$ l'ensemble des indices de lignes des éléments non (essentiellement) nuls de la $j^{\text{ème}}$ colonne de A, B et C. En particulier : $\mathcal{A}_j = \{i \text{ tels que } A_{i,j} \neq 0\}$. Montrer que

$$\mathcal{C}_j = \bigcup_{k \in \mathcal{B}_j} \mathcal{A}_k$$

Question 6. Donner un algorithme qui calcule le nombre d'éléments non essentiellement nuls du produit de deux matrices A et B. Donner sa complexité dans le pire cas, et dans le cas où les éléments non nuls de A sont répartis équitablement entre ses lignes et les éléments non nuls de B sont répartis équitablement entre ses colonnes.

Partie 2 Résolution d'un système triangulaire

On considère une matrice triangulaire inférieure L (c'est-à-dire telle que $L_{i,j} = 0$ pour $i < j$), et un vecteur b de taille n. On cherche x tel que $Lx = B$. On suppose que $L_{i,i} \neq 0$ pour tout $i = 0 \dots n-1$. On peut utiliser l'algorithme 1 pour effectuer cette résolution.

Algorithme 1 : Résolution de $Lx = b$

$x \leftarrow b$
pour $i = 0 \dots n-1$ **faire**
  $x_i \leftarrow \dfrac{x_i}{L_{i,i}}$
  pour $j = i+1 \dots n-1$ **faire**
    $x_j \leftarrow x_j - x_i \times L_{j,i}$

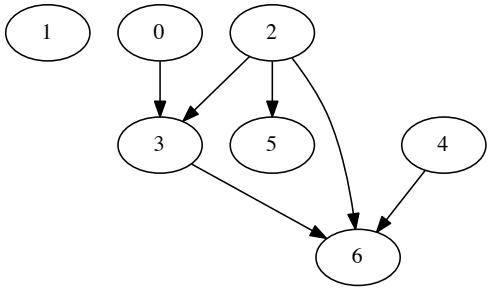
Question 7. Expliquer comment adapter l'algorithme 1 pour améliorer la complexité lorsque $nnz(L)$ est beaucoup plus petit que n^2, et que L est donnée par sa représentation compacte et b sous la forme d'un tableau de n éléments. Quelle est la complexité obtenue ? Enfin, que penser de cet algorithme si b contient beaucoup d'éléments nuls (on pourra regarder le cas $b_n = 1$ et $b_i = 0$ pour $i < n$).

On se concentre maintenant sur le cas où b contient également un grand nombre d'éléments nuls. On appelle **structure** d'une matrice (ou d'un vecteur) la position de ses éléments non-nuls (ou non essentiellement nuls dans le cas d'un résultat). On cherche à prévoir la structure de x en fonction des structures de L et b.

Question 8. On considère la valeur des coefficients de x après l'exécution de l'algorithme 1. Donner une condition nécessaire et suffisante pour que $x_i \neq 0$ en fonction de la structure de L et des x_j pour $j < i$.

Étant donnée une matrice triangulaire inférieure L, on définit le **graphe orienté** de la transposée de la matrice L, le graphe noté $G(L^T)$ où les sommets sont les entiers de 0 à $n-1$ et il existe un arc (j, i) pour tout $L_{i,j} \neq 0$ (on remarquera l'inversion des indices), que l'on pourra aussi noter $j \to i$. On donne ci-dessous l'exemple d'une matrice triangulaire inférieure L (avec indices des lignes/colonnes, les X représentant les éléments non nuls) et le graphe orienté de sa transposée $G(L^T)$.

$$L = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} X & & & & & & \\ & X & & & & & \\ & & X & & & & \\ X & & X & X & & & \\ & & & & X & & \\ & & X & & & X & \\ & & X & X & X & & X \end{pmatrix} \end{array}$$



Question 9. On appelle cycle dans un graphe orienté une suite d'arcs (u_1, u_2), (u_2, u_3), …, (u_{k-1}, u_k), (u_k, u_1) où $k \geqslant 2$. Montrer que le graphe $G(L^T)$ ne contient pas de cycle.

Question 10. Donnez une caractérisation des nœuds i du graphe $G(L^T)$ tels que x_i n'est pas essentiellement nul. En déduire un algorithme calculant l'ensemble des indices i tels que $x_i \neq 0$.

Question 11. On suppose connus les indices i tels que $x_i \neq 0$. Donner un algorithme calculant la solution de $Lx = b$, pour b creux. L'algorithme prendra en entrée les données suivantes :

- un tableau b_ligne contentant les indices (triés par ordre croissant) des éléments non-nuls des éléments de b,
- un tableau b_valeur contentant les valeurs de ces éléments dans le même ordre,
- un tableau x_ligne contentant les indices (triés par ordre croissant) des éléments non-nuls de x, calculés grâce aux questions précédentes.

L'algorithme doit calculer un tableau x_valeur contenant les valeurs des éléments non essentiellement nuls de x. On donnera la complexité de l'algorithme proposé.

Partie 3 Décomposition de Cholesky

On suppose dans cette partie que la matrice A est symétrique (pour tout couple i, j, $A_{i,j} = A_{j,i}$). Sous certaines conditions (si la matrice est définie positive), on peut décomposer A sous la forme du produit d'une matrice triangulaire inférieure L et de sa transposée (notée L^T) : $A = LL^T$. On suppose que $A_{j,j} \neq 0$ pour tout $j = 0 \ldots n-1$. On utilise l'algorithme 2 pour effectuer ce calcul.

Algorithme 2 : Décomposition de Cholesky $A = LL^T$

pour $j = 0, \ldots, n-1$ **faire**

$$L_{j,j} \leftarrow \sqrt{A_{j,j} - \sum_{k=0}^{j-1} {L_{j,k}}^2}$$

pour $i = j+1, \ldots, n-1$ **faire**

$$L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=0}^{j-1} L_{i,k} L_{j,k} \right)$$

Question 12. On considère la matrice L après l'exécution de l'algorithme 2. Pour un coefficient $L_{i,j}$ avec $i > j$, donner une condition nécessaire et suffisante pour que $L_{i,j} \neq 0$ en fonction de la structure de A et des $L_{k,l}$ pour $l < j$.

⋆ ⋆ ⋆

Question 13. On considère la matrice A ci-dessous (X désigne un élément non-nul). Donner la structure du résultat L obtenu par l'algorithme 2 appliqué à A.

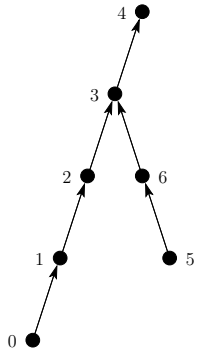
$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} X & X & X & X & X \\ X & X & & & \\ X & & X & & \\ X & & & X & \\ X & & & & X \end{pmatrix} \end{array}$$

Pour une permutation de σ de $0, \ldots, n-1$, on appelle **permutation de A par σ** et on note $\sigma(A)$ la matrice telle que $\sigma(A)_{i,j} = A_{\sigma(i),\sigma(j)}$.

Question 14. Pour la matrice A de la question précédente, proposer une permutation σ permettant de réduire le nombre d'éléments non-nuls du résultat L obtenu par l'algorithme 2 appliqué à $\sigma(A)$.

⋆ ⋆ ⋆

Pour $0 \leqslant j < n-1$, on note $\mathit{parent}(j) = \min\{i > j \text{ tel que } L_{i,j} \neq 0\}$ (pour la matrice L calculée par l'algorithme 2). Si pour j donné, il n'existe pas de $i > j$ tel que $L_{i,j} \neq 0$, alors on définit $\mathit{parent}(j) = -1$. La fonction *parent* permet de définir l'**arbre généalogique** des entiers $0, \ldots n-1$ en fonction de la matrice L : on dit que i est un **ancêtre** de j si $i = \mathit{parent}(j)$ ou $i = \mathit{parent}(\mathit{parent}(j))$, etc. Sur la figure ci-contre, la notation $j \longrightarrow i$ signifie $i = \mathit{parent}(j)$, les ancêtres de 0 sont 1, 2, 3 et 4 et $\mathit{parent}(4) = -1$.



Question 15. Montrer que si $L_{i,j} \neq 0$, alors i est un ancêtre de j.

Question 16. Soit $i > j$. Montrer que $L_{i,j} \neq 0$ si et seulement si il existe $k \leqslant j$ tel que $A_{i,k} \neq 0$ et j est soit un ancêtre de k, soit k lui-même.

On note $\mathcal{L}_i$ l'ensemble des indices de colonne des éléments non essentiellement nuls de la $i^{\text{ème}}$ ligne de L. Formellement :

$$\mathcal{L}_i = \big\{j \text{ tels que } L_{i,j} \neq 0\big\}$$

Question 17. Pour i donné, connaissant l'ensemble des éléments $A_{i,k}$ non essentiellement nuls, que représente $\mathcal{L}_i$ dans l'arbre généalogique ? En déduire un algorithme qui, étant donnée la fonction *parent*, calcule le nombre d'éléments non essentiellement nuls dans chaque **colonne** de L.

On considère l'arbre généalogique construit en ne considérant que les k premières lignes et colonnes de la matrice L, et on note parent_k la fonction associée : $\mathit{parent}_k(j) = \min\{i > j \text{ tel que } i \leqslant k \text{ et } L_{i,j} \neq 0\}$ et $\mathit{parent}_k(j) = -1$ s'il n'existe pas d'indice i tel que $j < i \leqslant k$ et $L_{i,j} \neq 0$.

Question 18. Étant donné $parent_{k-1}$ et la structure de A, comment construire $parent_k$? En déduire un algorithme pour calculer $parent(j)$ pour $j = 0, \ldots, n-1$.

⋆ ⋆

⋆