S

ALLIANCE SORBONNE
UNIVERSITÉ

utc

Rapport

TP3 - AI01

Ndeye Aminata AIDA FAYE

Fatima IMGHI

Complexité des fonctions

I. Remplissage d'une matrice creuse.

II. Affichage d'une matrice creuse sous forme de tableau.

III. Afficher une matrice creuse sous forme de listes

IV. Recherche d'une valeur

V. Affectation d'une valeur à un élément d'une matrice creuse.

VI. Additionner deux matrices creuses

VII. Calcul du gain en espace

VIII. Liberer la matrice

Complexité des fonctions

I. **Remplissage d'une matrice creuse**.

Dans le pire des cas, toutes les valeurs saisies sont non nulles. Donc le coût d'une insertion est en O(M) puisque
M colonnes. On a trois boucles for. Deux boucles imbriquées, $O(M * N)$ et une autre en $O(N)$

Complexité : $O(N * M)$

II. **Affichage d'une matrice creuse sous forme de tableau**.

Cette fonction parcourte toutes les lignes et toutes les colonnes de la matrice pour afficher chaque valeur. **Complexité:** $O(N * M)$

III. **Afficher une matrice creuse sous forme de listes**

parcours de toutes les lignes et toutes les colonnes de la matrice pour afficher chaque valeur. **Complexité**: $O(N * M)$

IV. **Recherche d'une valeur**

Dans le pire des cas on recherche jusqu'à la dernière colonne de la matrice. Le meilleur cas étant si l'élément est à la première colonne (N(1))

Complexité : *0(M)*

V.

Affectation d'une valeur à un élément d'une matrice creuse.

Dans le pire des cas, l'affectation se fait à la dernière colonne.

Complexité : *0(M)*

VI. Additionner deux matrices creuses.

la boucle externe parcours N lignes, et sur chaque ligne les boucles internes parcourt N1+ N2 éléments non nuls

Complexité: *O*(N + N1 + N2)

VII.

Calcul du gain en espace

Les opérations de calcul d'espace mémoire sont élémentaires (multiplications et soustractions), donc leur complexité est *O*(1).Or, La fonction compter parcourt chaque ligne de la matrice creuse pour compter le nombre d'éléments n . DONC:

Complexité : *0(n * N)*

VIII.

Libérer la matrice.

Cette fonction parcourt une matrice creuse ligne par ligne, et pour chaque ligne, elle libère la mémoire associée

aux éléments n de la liste de chaque ligne.

Complexité: *O(n * N)*

Ajout de fonctions utiles

void viderBuffer()

Ajout de cette fonction car nous avons utilisé getchar dans le main pour supprimer les caractères résiduels de l'entrée standard.

int compter(matrice_creuse m)

Cette fonction permet de calculer le nombre d'éléments non nuls dans la matrice creuse. Ensuite, connaissant le nombre d'éléments non nuls de la matrice, on détermine la taille de la matrice avec notre représentation.