Rapport de TP - INF 231 : Structure de Dononnées II

Introduction

Ce rapport présente la résolution d'une série d'exercices donnés dans le cadre du cours INF 231_EC2 : Structure de Données II. L'objectif est de mettre en pratique des algorithmes classiques en programmation C, tout en étudiant leur complexité et leur mise en œuvre pratique. Chaque exercice est détaillé avec son énoncé, l'algorithme proposé, la complexité et un exemple d'exécution.

Exercice 1 : Somme de matrices

Algorithme :

Additionner deux matrices de même taille en additionnant élément par élément.

Entrée : deux matrices A et B de taille n × m
Sortie : une matrice C = A + B

Algorithme :
Pour i de 0 à n-1 faire
  Pour j de 0 à m-1 faire
    C[i][j] ← A[i][j] + B[i][j]
Fin Pour
Fin Pour
Retourner C

Complexité :

O(n × m)

Exemple d'exécution :

Exemple :
A=[[1,2],[3,4]]
B=[[5,6],[7,8]]
C=[[6,8],[10,12]]

Exercice 2 : Produit de matrices

Algorithme :

Multiplier une matrice A (n×m) par une matrice B (m×p) en utilisant la formule classique.

Entrée : deux matrices A (n×m) et B (m×p)
Sortie : matrice C = A × B de taille n×p

Algorithme :
Pour i de 0 à n-1 faire
  Pour j de 0 à p-1 faire
    C[i][j] ← 0
    Pour k de 0 à m-1 faire
      C[i][j] ← C[i][j] + A[i][k] × B[k][j]
    Fin Pour
  Fin Pour
Fin Pour
Retourner C

Complexité :

O(n × m × p)

Exemple d'exécution :

Exemple :
A=[[1,2],[3,4]], B=[[5,6],[7,8]]
C=[[19,22],[43,50]]

Exercice 3 : Recherche séquentielle

Algorithme :

Parcourir le tableau et comparer chaque élément avec la valeur recherchée.

Entrée : un tableau T de taille n et un élément x
Sortie : position de x si trouvé, sinon -1

Algorithme :
Pour i de 0 à n-1 faire
  Si T[i] = x alors retourner i
Fin Pour
Retourner -1

Complexité :

O(n)

Exemple d'exécution :

Exemple :
T=[4,7,1,9], x=1 → position=2

Exercice 4 : Multiplication a × b avec seulement +1

Algorithme :

Répéter l'addition de a exactement b fois.

Entrée : deux entiers a et b (a,b>0)
Sortie : produit a × b

Algorithme :
résultat ← 0
Pour i de 1 à b faire
  résultat ← résultat + a
Fin Pour
Retourner résultat

Complexité :

O(b)

Exemple d'exécution :

Exemple : a=3, b=4 → résultat=12

Exercice 5 : Tester si un tableau est trié

Algorithme :

Vérifier que tab[i] ≤ tab[i+1] pour tout i.

Entrée : tableau T de taille n
Sortie : vrai si T trié croissant, sinon faux

Algorithme :
Pour i de 0 à n-2 faire
  Si T[i] > T[i+1] alors retourner faux

Fin Pour
Retourner vrai

Complexité :
O(n)

Exemple d'exécution :
Exemple : [1,2,3,4] → trié ; [3,1,2] → non trié

Exercice 6 : Trouver le médian d'un tableau

Algorithme :
Trier le tableau puis choisir l'élément du milieu (ou la moyenne de deux éléments centraux si n est pair).

Entrée : tableau T de taille n
Sortie : médian

Algorithme :
1. Trier le tableau T
2. Si n impair, retourner T[n/2]
3. Si n pair, retourner (T[n/2 - 1] + T[n/2]) / 2

Complexité :
O(n log n) (tri dominant)

Exemple d'exécution :
Exemple : [7,1,3] → trié=[1,3,7], médian=3

Exercice 7 : Inverser un tableau

Algorithme :
Échanger tab[i] et tab[n-1-i] jusqu'au milieu.

Entrée : tableau T de taille n
Sortie : T inversé

Algorithme :
Pour i de 0 à n/2-1 faire

échanger T[i] et T[n-1-i]
Fin Pour

Complexité :
O(n)

Exemple d'exécution :
Exemple : [1,2,3,4] → [4,3,2,1]

Exercice 8 : Produit vectoriel (3D)

Algorithme :
Appliquer la formule : (a2b3 - a3b2, a3b1 - a1b3, a1b2 - a2b1).

Entrée : deux vecteurs A=(a1,a2,a3), B=(b1,b2,b3)
Sortie : C = A × B

Algorithme :
C[0] = a2*b3 - a3*b2
C[1] = a3*b1 - a1*b3
C[2] = a1*b2 - a2*b1

Complexité :
O(1)

Exemple d'exécution :
Exemple : A=(1,2,3), B=(4,5,6) → C=(-3,6,-3)

Exercice 9 : Produit vecteur × matrice

Algorithme :
Multiplier un vecteur ligne par une matrice pour obtenir un nouveau vecteur.

Entrée : vecteur V (1×n), matrice M (n×m)
Sortie : vecteur R (1×m)

Algorithme :
Pour j de 0 à m-1 faire
  R[j] ← 0

```
  Pour i de 0 à n-1 faire
    R[j] ← R[j] + V[i] × M[i][j]
  Fin Pour
Fin Pour
Retourner R
```

Complexité :

O(n × m)

Exemple d'exécution :

Exemple : V=[1,2], M=[[3,4],[5,6]] → R=[13,16]

Répartition du travail

Le groupe de 5 membres s'est réparti les tâches de la manière suivante :
- Membre 1 : Somme et produit de matrices

- Membre 2 : Recherche séquentielle, multiplication avec +1

- Membre 3 : Vérification tri, médiane

- Membre 4 : Inversion de tableau, produit vectoriel

- Membre 5 : Produit vecteur × matrice, compilation et tests

Points d'amélioration

- Optimiser les algorithmes de tri pour améliorer la recherche du médian.
- Explorer des implémentations plus efficaces (par exemple recherche dichotomique au lieu de séquentielle).
- Modulariser davantage le code en C pour une meilleure réutilisation.

Conclusion

Ce TP nous a permis de mettre en pratique des concepts fondamentaux de la programmation et des structures de données. Nous avons étudié des algorithmes portant sur les matrices, les vecteurs et les tableaux, tout en analysant leurs complexités. Ce travail en groupe a favorisé la compréhension collective et a montré l'importance d'une répartition efficace des tâches.