TP ALGORITHMIQUE

11 Octobre 2024

Pour chaque exercice, vous allez écire un programme en C/C++ pour résoudre un problème donné. Si l'exercice contient plusieurs questions, vous devez écire une fonction correspondante à chaque question. Dans ce cas, chaque fonction doit contenir tous les codes pour répondre à la question (messages d'entrées, algorithme, messages de sorties,...).

Exercice 1

Étant donné une séquence d'entiers A = a[1..n] ($n \leq 5\ 000$, $-10000 \leq a[i] \leq 10000$). Une sous-séquence de A est une séquence contenant un certain nombre d'éléments de A qui conservent leur ordre. Trouvez la sous-séquence monotone (croissante) de A la plus longue.

Par exemple, si A = (1, 2, 3, 4, 9, 10, 5, 6, 7), la sous-séquence monotone (croissante) la plus longue est (1, 2, 3, 4, 5, 6, 7).

Entrée : fichier texte *INPMONOSEQ.TXT*

Ligne 1 : Le numéro n.

Ligne 2 : n nombres a[1], a[2], ..., a[n] séparés par au moins un espace.

Sortie : fichier texte *OUTMONOSEQ.TXT*

Ligne 1 : La longueur de la sous-séquence trouvée.

Les lignes suivantes : La sous-séquence trouvée et l'index des éléments sélectionnés dans cette sous-séquence.

INPMONOSEQ.TXT	*OUTMONOSEQ.TXT*
11 1 2 3 8 9 4 5 6 20 9 10	8 a[1] = 1 a[2] = 2 a[3] = 3 a[6] = 4 a[7] = 5 a[8] = 6 a[10] = 9 a[11] = 10

Indication: Utilisez un algorithme de programmation dymanique.

Exercice 2

Étant donné une séquence A de n ($1 \le n \le 1000$) entiers positifs a[1..n] et un entier positif k ($k \le 50$). Trouvez la sous-séquence avec le plus d'éléments de la séquence donnée telle que la somme des éléments de la sous-séquence soit divisible par k.

Entrée : fichier texte *INPDIVSEQ.TXT*

Ligne 1 : Le numéro n.

Ligne 2 : n nombres a[1], a[2], ..., a[n] séparés par au moins un espace.

Sortie : fichier texte *OUTDIVSEQ.TXT*

Ligne 1 : La longueur de la sous-séquence trouvée.

Lignes suivantes : Les éléments sélectionnés dans la sous-séquence.

Dernière ligne : La somme des éléments de cette sous-séquence.

INPDIVSEQ.TXT	*OUTDIVSEQ.TXT*
10 5 1 6 11 5 10 15 20 2 4 9	8 a[10] = 9 a[9] = 4 a[7] = 20 a[6] = 15 a[5] = 10 a[4] = 5 a[3] = 11 a[2] = 6 Sum = 80

Indication: Utilisez un algorithme de programmation dymanique.

Exercice 3

Supposons que G = (V, E) soit un graphe non orienté et connexe (il existe une chaîne qui relie deux sommets quelconques), ou V est un ensemble de sommets, E est un ensemble d'arêtes. L'arbre T = (V, F) avec $F \subset E$ est appelé arbre couvrant du graphe G. Autrement dit, si certaines arêtes de G sont supprimées pour obtenir un arbre, cet arbre est appelé un arbre couvrant.

Trouvez un arbre couvrant du graphe G.

Entrée : fichier texte *INPARBGRAPH.TXT*

Ligne 1 : Le nombre de sommets n et le nombre d'arêtes m.

Lignes suivantes : Les indices de deux sommets d'une arête du graphe.

Sortie : fichier texte *OUTARBGRAPH.TXT*

Dans chaque ligne, les indices de deux sommets d'une arête de l'arbre couvrant trouvé.

INPARBGRAPH.TXT	*OUTARBGRAPH.TXT*
11 14 1 2 1 3 2 3 2 4 2 5 3 6 3 7 4 8 5 8 5 9 6 10 6 11 7 11 10 11	1 2 2 3 2 4 3 6 4 8 5 8 6 10 7 11 10 11

On considère deux structures de données différentes pour représenter le graphe:

- *Matrice d'adjacence:* Une matrice de dimension n x n dont l'élément non diagonal aij est le nombre d'arêtes liant le sommet i au sommet j. L'élément diagonal aii est égal à 0 ($1 \leq i,j \leq n$).

- *Liste d'adjacence:* Un tableau de listes, la liste i contient des sommets adjacents au sommet i ($1 \leq i \leq n$). On suppose que les listes chaînées sont utilisées dans ce cas.

Implémentez deux versions de votre algorithme, une pour chaque structure de données.

Exercice 4

Un graphe non orienté est connexe s'il existe une chaîne reliant deux sommets quelconques. Un sous-graphe connexe maximal d'un graphe non orienté est une composante connexe de ce graphe. Déterminez si un graphe donné est connexe et, dans le cas contraire, toutes les composantes connexes du graphe.

Entrée : fichier texte *INPCONGRAPH.TXT*

Ligne 1 : Le nombre de sommets n et le nombre d'arêtes m.

Lignes suivantes : Les indices de deux sommets d'une arête du graphe.

Sortie : fichier texte *OUTCONGRAPH.TXT*

Ligne 1 : Le nombre de composantes connexes k

Lignes suivantes : Pour la composante connexe i (i=1,…,k), écrivez deux lignes:

- Ligne 1: composante connexe i.
- Ligne 2: Les indices des sommets de la composante connexe i.

INPCONGRAPH.TXT	*OUTCONGRAPH.TXT*
5 4 1 2 3 4 3 5 4 5	2 composante connexe 1 1 2 composante connexe 2 3 4 5

Implémentez deux versions de votre algorithme, une pour chaque structure de données dans l'exercice 4.

Exercice 5

Étant donné un graphe non-orienté G dont les arêtes sont munies de poids positifs.

5.1 Calculez un chemin le plus court entre deux sommets dans G. Utilisez une matrice d'adjacence pour représenter le graphe.

Entrée : fichier texte *INPDIJGRAPH.TXT*

Ligne 1 : Le nombre de sommets n, le nombre d'arêtes m, sommet de départ, sommet d'arrivée.

Lignes suivantes : Les indices de deux sommets d'une arête du graphe et le poid de l'arête.

Sortie : fichier texte *OUTCONGRAPH.TXT*

Ligne 1: La distance la plus courte trouvée.

Ligne 2: Le chemin le plus court trouvé.

INPDIJGRAPH.TXT	*OUTDIJGRAPH.TXT*
6 7 1 4 1 2 1 1 6 20 2 3 2 3 4 20 3 6 3 5 4 5 6 5 4	15 1 → 2 → 3 → 6 → 5 → 4

5.2. La même question que celle dans 6.1 en utilisant une liste d'adjacence pour représenter le graphe.

5.3. La même question que celle dans 6.1 en utilisant un tas pour stocker les nœuds de distance minimale. Vous utiliserez un tableau pour représenter le tas et deux structures de données (celles considérées dans les questions 6.1 et 6.2) pour représenter le graphe.

Quels sont les avantages des solutions dans 6.2 et 6.3 par rapport à celle dans 6.1 ?

5.4. Pour chaque sommet u qui est différent de celui de départ, déterminez le chemin le plus court entre le sommet de départ et u. Utilisez une matrice d'adjacence pour représenter le graphe.