**Теорія графів**

Основні алгоритми роботи з графами:

<http://www.e-olymp.com/uk/problems/4764> - Матриця суміжності, степінь вершин

<http://www.e-olymp.com/uk/problems/4763> - Від списку ребер до матриці суміжності

<http://www.e-olymp.com/uk/problems/625> - Пошук в глибину на графах

<http://www.e-olymp.com/uk/problems/975> - Флойд (зчитування матриці)

<http://www.e-olymp.com/uk/problems/983> - Флойд (створення матриці)

<http://www.e-olymp.com/uk/problems/2968> - Флойд (Форд)

<http://www.e-olymp.com/uk/problems/1365> - Дейкстри

<http://www.e-olymp.com/uk/problems/2965> - Дейкстра

<http://www.e-olymp.com/uk/problems/981> - мінмальне остове дерево (алгоритм Прима)

<http://www.e-olymp.com/uk/problems/964> - Матриця інцендентності

**1. Основи теорії графів**

Введемо деякі основні поняття, що стосуються теорії графів.

1. Граф представляє собою не порожню множину точок і ліній, два кінці котрих належать заданій множині точок.

2. Точки 1,2,3,4,5,6 - вершини графа.

3. Відрізки 12,24,45,51,13,34,23,35 – ребра графа.

4. Вершина 6 не належить ребру і називається ізольованою (але вона частина графа).

5. Кількість ребер, які виходять з даної вершини визначають степінь вершини графа. Вершини відрізняються кількістю ребер, котрим вона належить (степінь вершини – число ребер)

# Вершина 6 має 0 степінь, а 1 – 3 степінь.

 Фігури, які складаються з ряду точок, з'єднаних між собою лініями, називаються графами. Точки є вершинами графа, а лінії –­­­ ребра графа.

 Відкриті Ейлером властивості графа:

 1. Число непарних вершин зв'язного графа завжди парне. Неможливо накреслити граф з непарним числом непарних вершин.

 2. Якщо всі вершини графа парні, то можна одним розчерком (тобто не відриваючи олівця від паперу) накреслити граф, не проводячи по кожному ребру більше одного разу. При цьому можна починати з будь-якої вершини графа і закінчувати його в тій же вершині.

3. Граф лише з двома непарними вершинами можна накреслити одним розчерком, при цьому рух потрібно почати в одній точці з цих вершин і закінчити в другій непарній вершині.



4. Граф з більше ніж двома непарними вершинами неможливо накреслити одним розчерком.

Оскільки число непарних вершин графа в задачі про кенігсбергські мости рівне 4, то такий граф неможливо зобразити одним розчерком, неможливо пройти по всіх мостах по одному разу.

Ейлеровим графом (шляхом або циклом) називається граф, що має по одному всі ребра графа. Замкнута лінія, яку можна накреслити одним розчерком, називається універсальною.

Повертаючись до задачі, зазаначимо, що можна удосконалити систему мостів, щоб здійснити прогулянку, проходячи по кожному з них тільки один раз. (Додати 1 міст або 1 міст забра­­­ти).

Зв’язний граф – граф, в якого кожні дві вершини є зв’язаними між собою ребрами.

Неорієнтований граф:



Орієнтований граф:



Орієнтований, навантажений граф:



Гамільтонів шлях проходить через кожну вершину по одному разу (по ребрах проходить декілька разів або жодного)

 Елілеровий шлях – це шлях, який ми проходимо з однієї вершини в іншу через всі ребра тільки один раз.

**2. Задачі**

**1. Ейлеровий шлях.**

Графом називається набір точок, деякі з яких з'єднані відрізками. Граф називається ейлеровий якщо його можна накреслити одним розчерком (не відриваючи руки выд паперу). Складіть алгоритм для визначення існування ейлеровго шляху.

З вхідного файлу зчитується число N<=1000 кількість вершин графу в наступних N рядках по N чисел задається матриця суміжності (0 - немає зв’язку та 1 – є зв’язок).

У вихідний файл вивести повідомлення YES, NO.

Зчитування матриці

int d[1000][100];

int n;

cin>>n;

 for(int i=0;i<n;i++)

 for(int j=0;j<n;j++)

 cin>>d[i][j];

**2. Задача про туриста (пошук в глибину)**

Турбаза мала для ночівлі N місць, з’єднаних стежками. Туристів можна вести в одну сторону. Довжина стежки – одноденний перехід. Пройти і перевірити всі M-денні маршрути, які починаються на базі K.

Орієнтований ненавантажений граф.

n=6; m=3; k=1;



Дано орієнтований ненавантажений граф. N вершин. Знайти всі маршрути довжини m=3, які виходять з вершини з номером k.

В алгоритмі розв’язування даної задачі використовується стек та рекурсивний пошук.

int a[1000][100],c[100];

int n,s,f;

void p(int i, int v)

{

c[i]=v;

if(v==f || i>=n) {

if(v==f){

 for(int j=0;j<=i;j++)cout<<c[j]<<” “;

cout<<endl;

}

}

else

for(int j=1;j<=n;j++)

if(a[v][j]>0) p(i+1,j);

}

int main()

{

 cin>>n>>s>>f;

 for(int j=1;i<=n;i++)

 for(int j=1;j<=n;j++)

cin>>a[i][j];

p(0,s);

}

3. Задача. (, алгоритм Флойда)

На вимогу класного керівника учень знайшов в Інтернеті карту міста на якій він визначив і задав в декартовій системі координат координати точок на шляху від доми до гімназії, і намалював дороги між ними (див. рис). Допоможіть учню визначити хоча б довжину найкоротшої дороги від доми до школи.

**Вхідні дані:**

Перший рядок вхідного файлу містить натуральне число N (1<=N<=100) – кількість точок на карті.

Наступні N рядків містять через проміжок координати Xi , Yi точок на карті. Значення координат по модулю менші 50000. Перші координати задають – координати доми, а останні – координати гімназії,,

Наступні рядків задають карту намальованих доріг початкова та кінцева точка.

**Вихідні дані:**

Єдиний рядок має містити дійсне число з трьома знаками після коми – дожину найкоротшої дороги.

**Приклади:**

|  |  |
| --- | --- |
| input.txt | output.txt |
| 6150 70160 90100 100170 120120 14080 1601 22 32 43 54 55 6 | 152.556 |

for (int k=0; k<n; ++k)

 for (int i=0; i<n; ++i)

 for (int j=0; j<n; ++j)

 d[i][j] = min (d[i][j], d[i][k] + d[k][j]);

3. Задача (100 балів , алгоритм Прима)

 *Збирання мита.*

Король країни Аріїв завоював N міст на території сусідніх держав.

Тепер йому необхідно створити систему збирання мита з завойованих територій. Він хоче збудувати таку систему шляхів між цими містами, щоб до будь-якого міста можна було дістатися (можливо, через інші міста) зі столиці, але у воєнному стані на транспорт виділяється дуже незначна частина фінансів, тому сумарна вартість побудованих шляхів сполучення між містами має бути мінімальною.

Вхідні дані:

Перший рядок вхідного файлу містить натуральне число N (1<=N<=100) – кількість міст у країні, а також цілі числа X та Y – координати столиці.

Наступні N рядків містять через проміжок координати Xi , Yi завойованих міст.

Значення координат по модулю менші 50000.

Вихідні дані:

Перший рядок має містити дійсне число з трьома знаками після коми – сумарну вартість побудованих доріг. Вважайте, що вартість одиниці довжини дороги дорівнює одній умовній одиниці.

Наступні рядки мають містити у довільному порядку список побудованих доріг у форматі:  =>

При цьому столицю позначте номером 0. Якщо відповідей декілька, виведіть одну довільну з них. Приклади:

|  |  |
| --- | --- |
| TALLAGE.DAT | TALLAGE.SOL |
| 6  0 0 1  1-1  1 0  2 1 -1-1 -1 0 -2 | 8.4852 33 11 00 44 66 5 |

#include <iostream>

using namespace std;

int a[100][100];

int c[100];

int n,s=10000;

void p(int k, int v)

{

 c[k]=v;

 if (v==4 || k>n)

 {if (v==4)

 {

 int s0=0;

 for(int i=1;i<=k-1;i++)//cout<<c[i]<<" ";

 s0=s0+a[c[i]][c[i+1]];

 //cout<<s0<<endl;

 //cout<<endl;

 if (s0<s)s=s0;

 }

 }

 else

 for(int i=1;i<=n;i++)

 if (a[v][i]>0 ) p(k+1,i);

}

int main()

{

 cin>>n;

for(int i=1;i<=n;i++)

 for(int j=1;j<=n;j++)

 cin>>a[i][j];

/\*for(int i=1;i<=n;i++){

 for(int j=1;j<=n;j++)

 cout<<a[i][j]<<" ";

 cout<<"\n";

}

\*/

p(1,1);

cout<<s<<endl;

 return 0;

}