#include <iostream> // Prim算法(普里姆)

#include <iomanip>

using namespace std;

const int INF = 0x3fffffff;

const int N = 100;

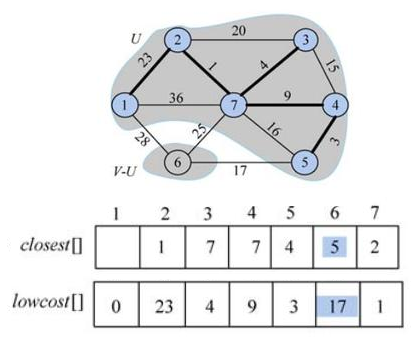
int map[N][N]; // 带权邻接矩阵

bool s[N]; // 顶点集合V，如果s[i]=true,说明顶点i已加入最小生成树的顶点集合U，

// 否则顶点i属于集合V-U，逐步在两个集合中选择最小的连接边

int closest[N]; // closest[j]表示V-U中的顶点到U中的最近邻接点

int lowcost[N]; // lowcost[j]表示V-U中的顶点到U中的最近邻接点的边值，即边(closet[j],j)的权值



void Prim(int n, int u0) // 顶点个数n、开始顶点u0

{

int i,j; // 循环变量

// (1) 初始化

s[u0] = true; // 初始时集合U中只顶点u0

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

{

if(i != u0)

{

lowcost[i] = map[u0][i];

closest[i] = u0;

s[i] = false;

}

else

lowcost[i] =0;

}

for(i = 1; i <= n; i++) // (2) 在集合V-U中寻找距离集合U最近的顶点t

{

int temp = INF;

int t = u0;

for(int j = 1; j <= n; j++) // ③

{

if((!s[j]) && (lowcost[j] < temp))

{

t = j;

temp = lowcost[j]; // temp此处更新，循环比较

}

}

if(t == u0)

break; // 找不到t，跳出循环

s[t] = true; // 否则，将t加入集合U (3) 集合更新

for(j = 1; j <= n; j++) // (4) 数组更新，更新lowcost和closest

{

if((!s[j]) && (map[t][j] < lowcost[j]))

{

lowcost[j] = map[t][j]; // 更新了上面表达式的比较标准

closest[j] = t;

}

}

}

}

void printMatrix(int n) // 输出图的带权邻接矩阵

{

cout <<"输出矩阵map："<<endl;

cout<<"\t";

int i,j; // 循环变量

for(i=1;i<=n;++i) // 输出列标题

cout<<"\t"<<i;

cout<<endl;

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

{

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

{

if(j==1) // 输出行标题

cout << "\t" << i;

if(map[i][j]==INF)

cout << "\t" << "∞";

else

cout << "\t" << map[i][j];

if(j==i)

cout<<endl;

}

}

}

void initMatrix(int& n, int& m)

{

int u, v, w;

cout <<"输入结点数n和边数m："<<endl;

cin >> n >> m;

int i,j; // 循环变量

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

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

map[i][j] = INF;

cout <<"输入结点u，v和边值w："<<endl;

for(i=1; i<=m; i++)

{

cin >> u >> v >> w;

map[u][v] = map[v][u] = w;

}

}

void output(int n)

{

int i; //循环变量

cout <<"序号、数组colesest、lowcost的值为："<<endl;

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

cout<<setw(4) << i;

cout<<"\n";

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

cout<<setw(4)<<closest[i];

cout<<"\n";

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

cout<<setw(4) << lowcost[i];

cout << endl;

int sumcost = 0;

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

sumcost += lowcost[i];

cout << "最小的花费是：" << sumcost << endl << endl;

}

int main()

test data

7

12

1 2 23

1 6 28

1 7 36

2 3 20

2 7 1

3 4 15

3 7 4

4 5 3

4 7 9

5 6 17

5 7 16

6 7 25

1

{

int n, m,u0;

initMatrix(n, m);

cout <<"输入任一结点u0："<<endl;

cin >> u0 ;

printMatrix(n);

Prim(n, u0);

output(n);

system("pause");

return 0;

}

输入任一结点u0：

1

输出矩阵map：

1 2 3 4 5 6 7

1 ∞

2 23 ∞

3 ∞ 20 ∞

4 ∞ ∞ 15 ∞

5 ∞ ∞ ∞ 3 ∞

6 28 ∞ ∞ ∞ 17 ∞

7 36 1 4 9 16 25 ∞

序号、数组colesest、lowcost的值为：

1 2 3 4 5 6 7

0 1 7 7 4 5 2

0 23 4 9 3 17 1

最小的花费是：57

1 子图：从原图中选取一些顶点和边组成的图，称为原图的子图

2 生成子图：选中一些边和所有顶点组成的图，称为原图的生成子图

3 生成树：如果生成的子图恰好是一棵树，则称为生成树；

4 最小生成树：权值之和最小的生成树，则称为最小生成树；

找出n-1条权值最小的边并要保证没有回路