**实 验八：利用Prim算法求无向网的最小生成树（2学时）**

**[问题描述]**

如要在n个城市之间建设通信网络，只需架设n-1条线路即可。如何以最低的经济代价建设这个通信网，是求一个无向网的最小生成树问题。

**[实验目的]**

1. 掌握图的各种存储结构和基本操作。
2. 对于实际问题的求解会选用合适的存储结构。
3. 通过Priml算法理解如何求无向网的最小生成树。

**[实验内容及要求]**

（1）构造具有n个顶点的无向网，并利用Priml算法求网的最小生成树。

（2）以文本形式输出所求得的最小生成树中各条边以及它们的权值。

/\*

 \* @?: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* @Author: Weidows

 \* @Date: 2020-12-21 21:05:58

 \* @LastEditors: Weidows

 \* @LastEditTime: 2020-12-31 23:20:45

 \* @FilePath: \Weidows\C++\homework\ing\大二上\实验八源代码\实验8.cpp

 \* @Description:

 \* 1、带权无向图，给出节点个数以及所有边权值，用Prim算法求最小生成树。

 \* 2、2个for循环都是从2开始的，因为一般我们默认开始就把第一个节点加入生成树，因此之后不需要再次寻找它。

 \* 3、lowcost[i]记录的是以节点i为终点的最小边权值。初始化时因为默认把第一个节点加入生成树，因此lowcost[i] = graph[1][i]，即最小边权值就是各节点到1号节点的边权值。

 \* 4、mst[i]记录的是lowcost[i]对应的起点，这样有起点，有终点，即可唯一确定一条边了。初始化时mst[i] = 1，即每条边都是从1号节点出发。

 \* @!: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \*/

**#include** <stdio.h>

**#define** MAX 100

**#define** INF **0x**3f3f3f3f

***int*** graph[MAX][MAX];

***int*** Prim(***int*** *graph*[][MAX], ***int*** *n*)

{

***int*** lowcost[MAX], mst[MAX]; //lowcost[i]记录以i为终点的边的最小权值，当lowcost[i]=0时表示终点i加入生成树,mst[i]记录对应lowcost[i]的起点，当mst[i]=0时表示起点i加入生成树

***int*** i, j, min, minid, sum **=** 0;

**for** (i **=** 2; i **<=** *n*; i**++**) //默认选择1号节点加入生成树，从2号节点开始初始化

  {

    lowcost[i] **=** *graph*[1][i]; //最短距离初始化为其他节点到1号节点的距离

    mst[i] **=** 1;               //标记所有节点的起点皆为默认的1号节点

  }

  mst[1] **=** 0; //标记1号节点加入生成树

**for** (i **=** 2; i **<=** *n*; i**++**) //n个节点至少需要n-1条边构成最小生成树

  {

    min **=** INF;

    minid **=** 0;

**for** (j **=** 2; j **<=** *n*; j**++**) //找满足条件的最小权值边的节点minid

    {

**if** (lowcost[j] **<** min **&&** lowcost[j] **!=** 0) //边权值较小且不在生成树中

      {

        min **=** lowcost[j];

        minid **=** j;

      }

    }

    printf("%c - %c : %d\n", mst[minid] **+** 'A' **-** 1, minid **+** 'A' **-** 1, min); //输出生成树边的信息:起点，终点，权值

    lowcost[minid] **=** 0;                                                   //标记节点minid加入生成树

**for** (j **=** 2; j **<=** *n*; j**++**) //更新当前节点minid到其他节点的权值

    {

**if** (*graph*[minid][j] **<** lowcost[j]) //发现更小的权值

      {

        lowcost[j] **=** *graph*[minid][j]; //更新权值信息

        mst[j] **=** minid;               //更新最小权值边的起点

      }

    }

  }

**return** 0; //返回最小权值和

}

***int*** main()

{

***int*** m, n, weight;

***char*** chx, chy;

  printf("请输入节点数与边数 ");

  scanf("%d %d", **&**m, **&**n);

**if** (m **<=** 2 **||** n **<=** 0)

  {

    printf("输入错误，结束程序\n");

**return** 0;

  } //读取节点和边的数目

  printf("请输入节点与权值(节点要输入为大写英文，权值为阿拉伯数字)\n");

  setbuf(stdin, NULL);

**for** (***int*** i **=** 1; i **<=** m; i**++**) //初始化图，所有节点间距离为无穷大

**for** (***int*** j **=** 1; j **<=** m; j**++**)

      graph[i][j] **=** INF;

**for** (***int*** k **=** 0; k **<** n; k**++**) //读取边信息

  {

    scanf("%c %c %d", **&**chx, **&**chy, **&**weight);

    setbuf(stdin, NULL);

***int*** i **=** chx **-** 'A' **+** 1, j **=** chy **-** 'A' **+** 1;

    graph[i][j] **=** weight;

    graph[j][i] **=** weight;

  }

  Prim(graph, m);

**return** 0;

}

**[测试数据]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 4 | 3 |  |  |
| B | 4 | 0 | 5 | 5 | 9 |
| C | 3 | 5 | 0 | 5 | 0 |
| D |  | 5 | 5 | 0 | 7 |
| E |  | 9 |  | 7 | 0 |

5个节点,7条边

对角线取其中一边

