**实验四 图**

PB17050941 李喆昊

1. 问题描述

题目1：图的遍历

   输入一个无向图，输出图的深度优先搜索遍历顺序与广度优先搜索遍历顺序。显然的，最后的答案会有多种可能，这里统一要求：当有多个节点可以搜索时，优先去节点编号最小的那个。

输入格式：

第一行是两个数n，m(1<n<30，1<m<300)，分别表示顶点数量和边的数量

接下来的m行每行输入两个数a、b；表示顶点a与顶点b之间有边相连，顶点编号从1到n

最后输入一个数s表示遍历的起始顶点编号。

输出格式：

   输出两行序列，第一行为深度优先搜索遍历的顺序，第二行为广度优先搜索遍历的顺序

输入输出样例：

|  |  |
| --- | --- |
| 输入样例 | 输出样例 |
| 5 6  1 5  1 4  3 1  2 1  2 5  5 3  1 | 1 2 5 3 4  1 2 3 4 5 |

说明：

   当有多个节点可以搜索时，优先去节点编号最小的那个。

题目2：求通讯网的最小代价生成树

输入一个无向铁通讯网图，用Prim和Kruskal算法计算最小生成树并输出。

输入格式：

第一行是两个数n，m(1<n<10000，1<m<100000)，分别表示顶点数量和边的数量。

接下来的m行每行输入三个数a、b、w；表示顶点a与顶点b之间有代价为w的边相连，顶点编号从1到n。

输出格式：

   输出包含一个数，即最小生成树的各边的长度之和

输入输出样例：

|  |  |
| --- | --- |
| 输入样例 | 输出样例 |
| 4 5  1 2 2  1 3 2  1 4 3  2 3 4  3 4 3 | 7 |

说明：

   本题检查时并不要求提交两份代码，只需要完成题目要求即可，但是在提交试验报告时请两种方法都提交代码并详细介绍。

题目3：铁路交通网的最短路径

输入一个无向铁路交通图、始发站和终点站，用Dijkstra算法计算从始发站到终点站的最短路径。

输入格式：

第一行是两个数n，m(1<n<100000，1<m<1000000)，分别表示顶点数量和边的数量

接下来的m行每行输入三个数a、b、w；表示顶点a与顶点b之间有长度为w的边相连，顶点编号从1到n

最后输入两个数s，t表示遍历的起始顶点编号和终点编号

输出格式：

   一个数为从起点到终点的最短路径长度

输入输出样例：

|  |  |
| --- | --- |
| 输入样例 | 输出样例 |
| 4 5  1 2 10  1 3 20  2 3 15  2 4 30  3 4 20  1 4 | 40 |

说明：

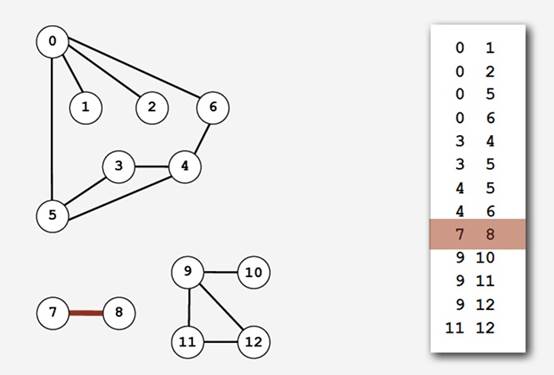
   本题要求使用Dijkstra算法，所以将会有时间要求，超过时间限制则不算正确。

附加题：显示图

   可以直接使用题目1的输入数据画出图的情况则为满足要求，要求图整体美观，可以体现出顶点的编号，边的交叉尽量少。

反例：所有点画一圈之后在中间连线。

举例：



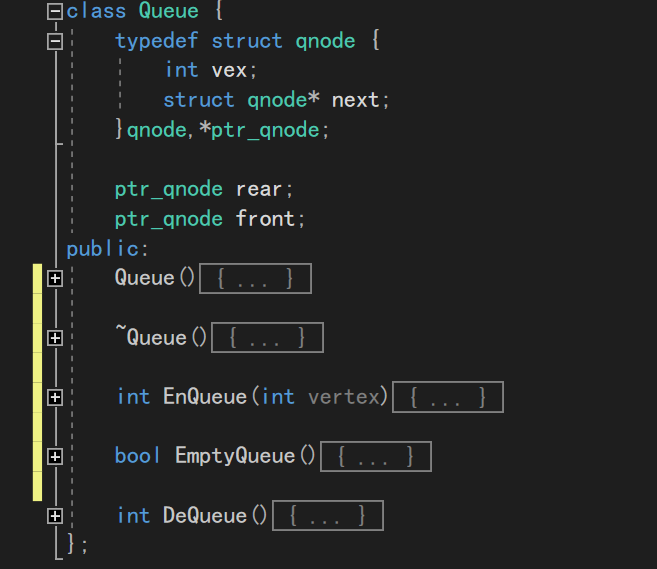
1. 算法描述
2. 数据结构描述
3. 题目1

采用邻接矩阵作为图的存储方式。逻辑结构为图状结构。

主要变量：



由于涉及到广度优先搜索，故使用了链式队列，类定义如下：





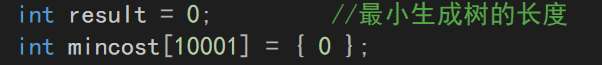
vexnum、edgenum分别是顶的个数与边的个数。G是用邻接矩阵存储的图。

1. 题目2
2. Prim算法：



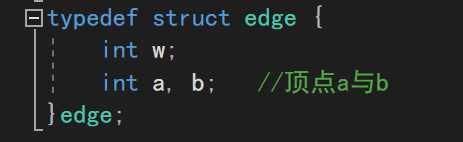
使用一个足够大的邻接矩阵存储图。



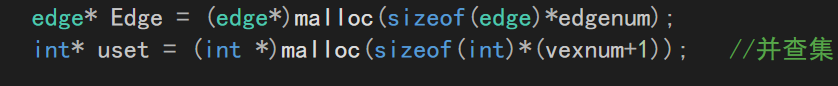


主要变量有顶和边的个数、边权、最小生成树的长度、当前选取的最小权边等。

1. Kruskal算法：



定义edge结构体用于存储边相关信息。



同时使用并查集判断找到的边是否会形成圈。

1. 题目3



同Prim算法使用一个足够大的邻接矩阵存储图。

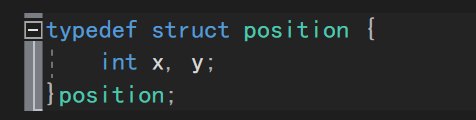


存储图中顶到起始顶距离的数组。

1. 附加题

使用邻接矩阵作为图的存储形式。

主要变量：



position结构体存储要画的顶的（x,y）坐标。



图中顶的个数、边的个数。

1. 程序结构描述
2. 题目1

使用DFS与BFS对图进行遍历。



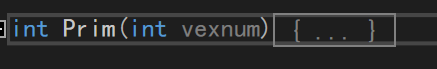
接口：begin\_vex：遍历起始顶，vexnum:顶的数目，visited数组：用于记录顶是否已经走过。



接口与DFS同，que是队列。

1. 题目2

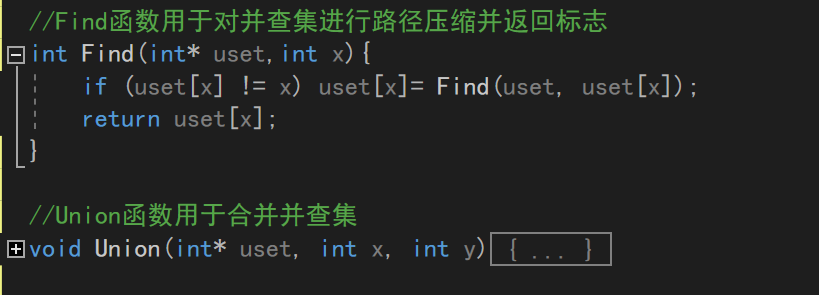
① Prim算法：



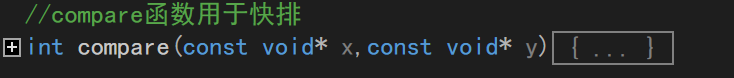
程序主体就是Prim函数。

②Kruskal算法：

并查集相关函数：



在Kruskal函数中我使用了cstdlib中的qsort函数对边根据权值从小到大排序，故定义了compare函数：

 Kruskal函数：



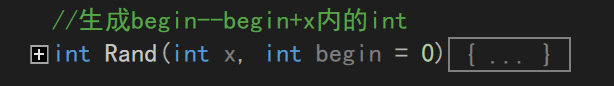
1. 题目3

程序主体就是Dijikstra函数：



1. 附加题

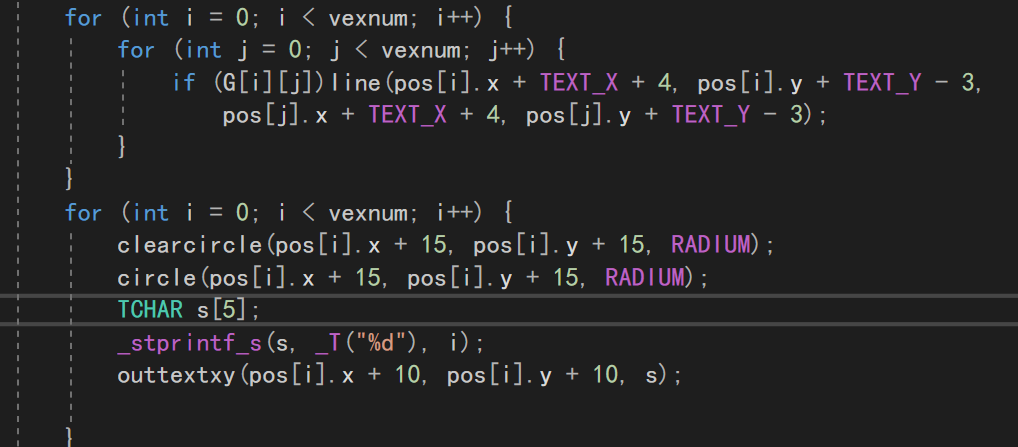
程序中画图调用了<graphics.h>这个库。



Rand函数用于生成(begin, begin+x)范围内的int随机数。



RandomPos函数用于随机生成绘制在画板上的图的num个顶的位置信息（位置信息用position结构体存储），且可以使顶的坐标之间的距离大于一个定值（为了保证图的顶画出来后不会挤在一起）。



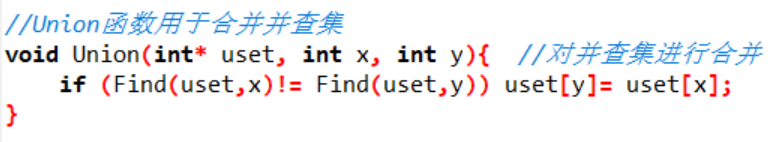
使用如上代码进行图的绘制：先画线再画圈，并且清空圈内的线头，最后绘制文字。

三．调试分析

（1）使用的是助教提供的测试数据。

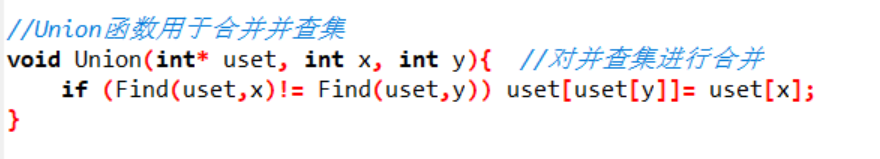
（2）程序调试中遇到的问题：

① Kruskal算法调试时的结果比答案结果小：原因是合并并查集的时候出现错误



初始代码为上图。uset[x]是x的父节点（及该并查集的标志元素）

但这个代码的问题是只修改了一个节点y的父节点，并没有实现两个并查集的合并，导致得出的生成树中有圈。改为下图代码后结果正确。



②Prim初始时使用了快排算法对边进行排序导致超时。解决方法：调整代码结构，不使用快排。

四．算法时空分析

（1）题目1

DFS：O(n^2) , BFS: O(n^2)

1. 题目2

Prim算法： O(n^2)

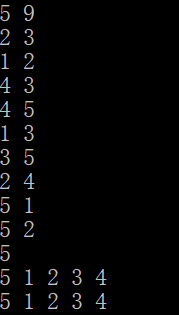
Kruskal算法：O(elog(e))

（3）题目3

Dijikstra: O(n^2)

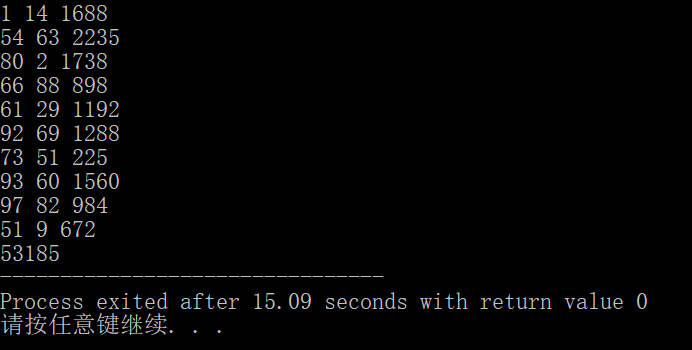
五．测试结果及分析

（1）题目1

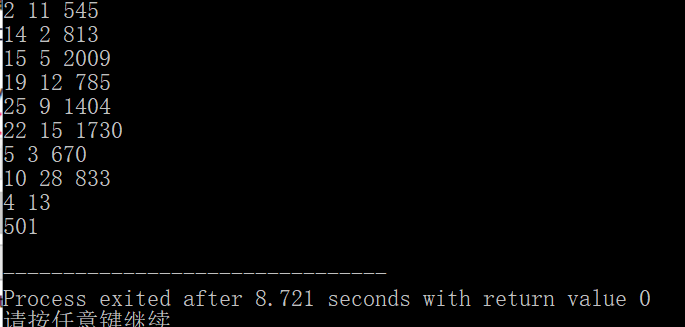


结果正确。

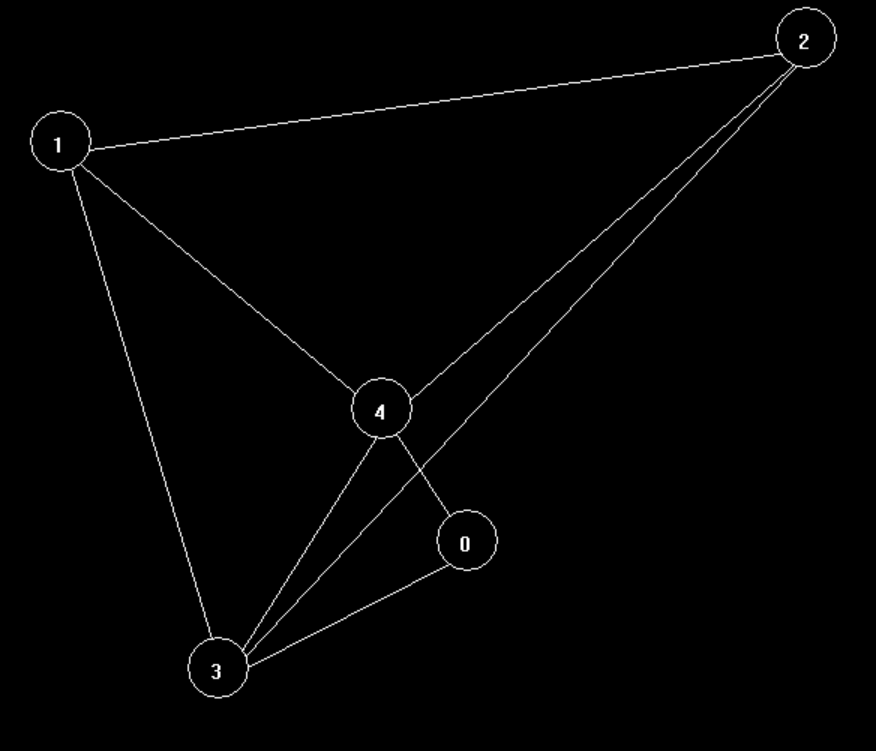
（2）题目2

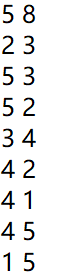


（3）题目3



（4）题目4





六．实验体会和收获

1. 题目一让我练习了DFS和BFS的写法，增进了对图的遍历的理解。
2. 题目二三验收时有时间限制，为了进行时间上的优化，我自主学习加请教别人，增加了对算法时间性能的理解。
3. 题目四让我们把图直观地画出来，这正是我感兴趣的。但是画出来的图的边不能有过多的交叉，为此我通过使用一个简单但有效的方法：控制顶之间的距离使得顶不能离得太近，从而也减少了边的交叉。