**Project 4实验报告**

20337263 俞泽斌

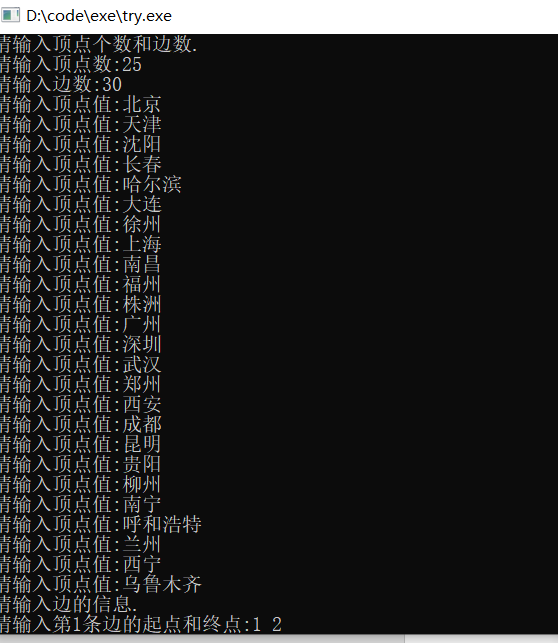
1. 程序功能简要说明。

首先是实现了ppt上的几个功能，包括领接表的输出，通过递归的和非递归的方式输出深度优先遍历序列，通过递归的方式输出广度优先遍历序列。然后是建立深度优先生成树和广度优先生成树，并将他们按树形输出，我这里用的是括号的形式，并用了前序和中序遍历来判断是否建树成功。

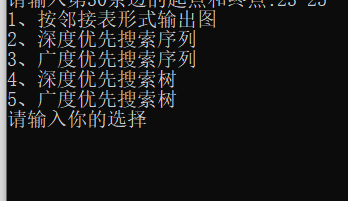
1. 程序运行截图，包括计算功能演示、部分实际运行结果展示、命令行或交互式界面效果等。



首先是图的建立，先输入顶点数和边数，然后输入顶点值和边值，具体的我都放在源.txt里面了，可以方便复制，主要要关注的可能还是边的标号是从1开始的，如果从0开始可能会导致段错误。

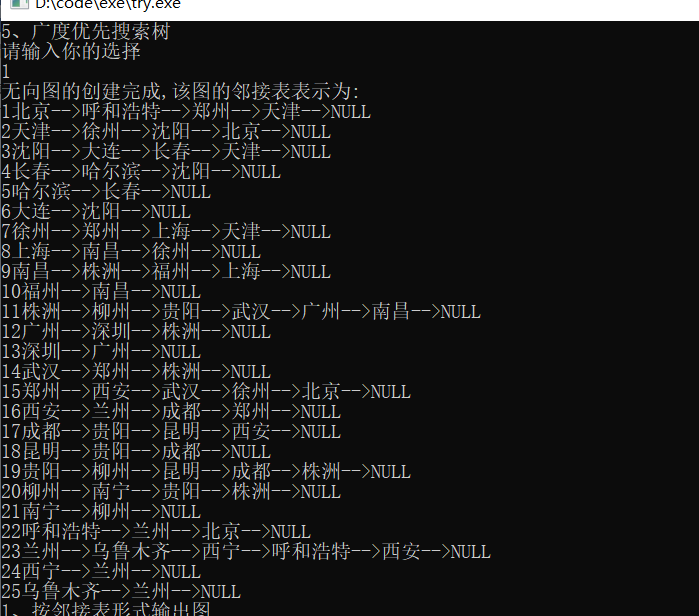


具体输出顶点值 和边值之后就会进入新的界面，也就是一个小型的菜单



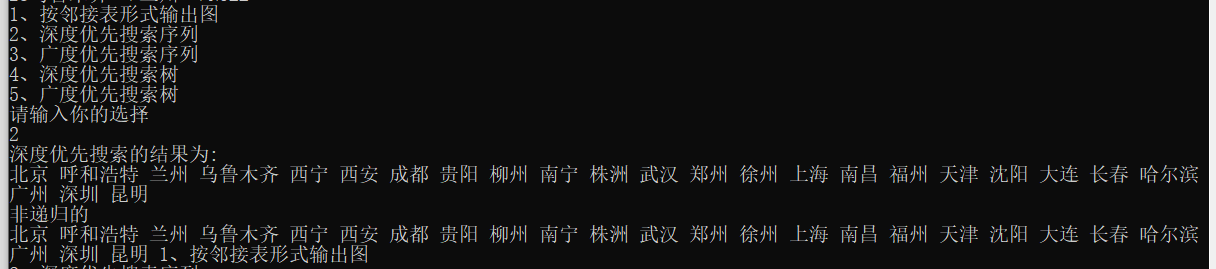
可以输入相应的选择。

首先是领接表输出图



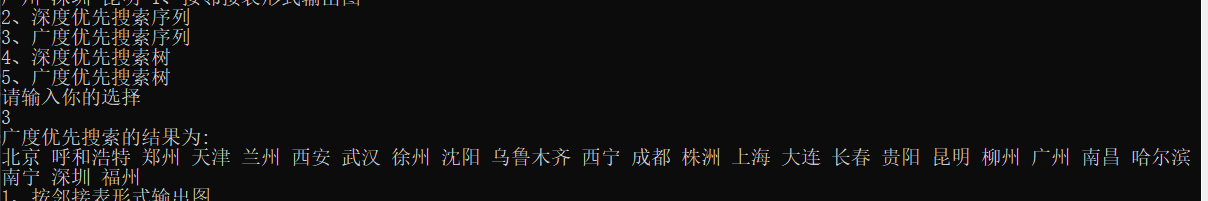
就将每一个顶点作为起点都输出了一遍领接表。

第二是深度优先序列的输出

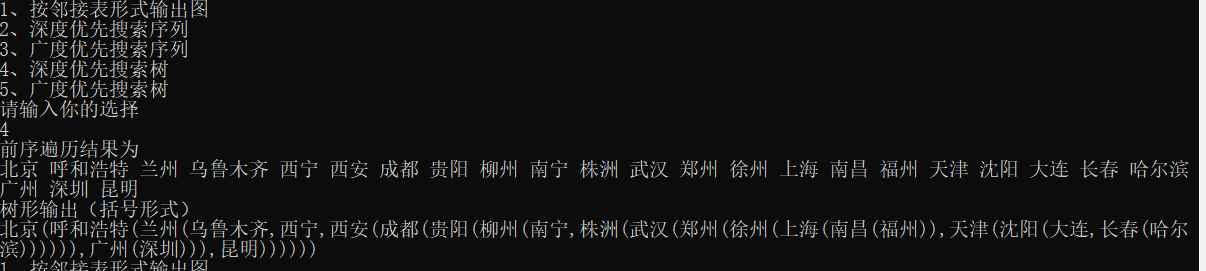


主要这里的要求有两个，一个是递归的方式，另一个是非递归的方式，我就把两个上下输出，便于关注两个是否出错。

第三是广度优先序列的输出

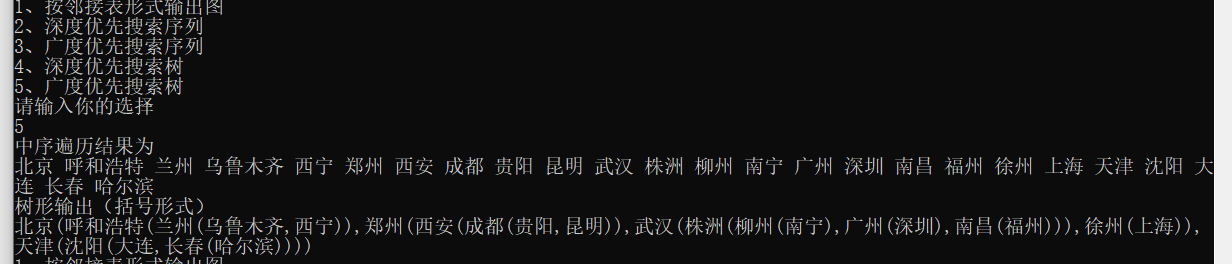


第四是深度优先搜索树的建立和输出



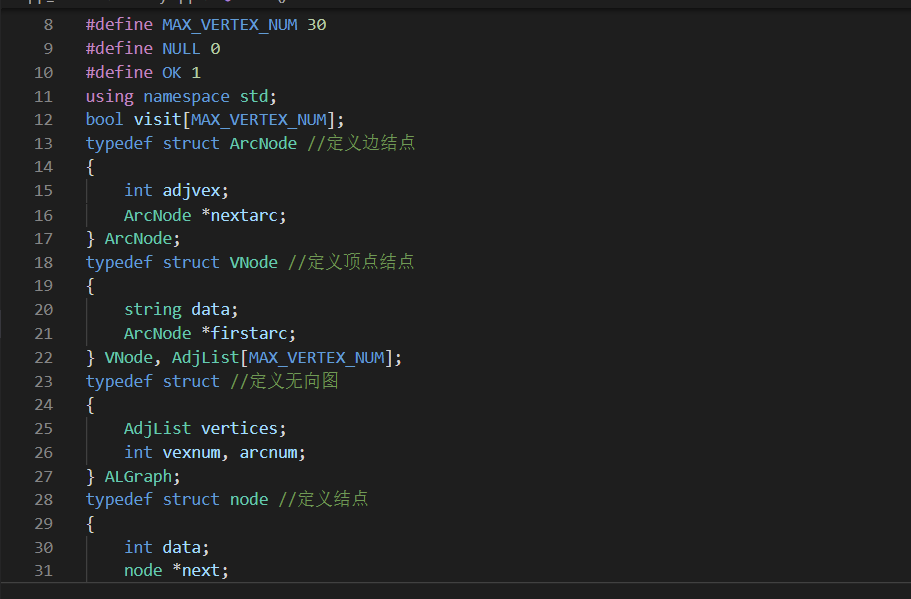
关于树形输出，我这里采用的是括号的形式来表示儿子节点的关系，括号里面的各位就是兄弟节点，然后用一个前序遍历来判断是否建树成功。

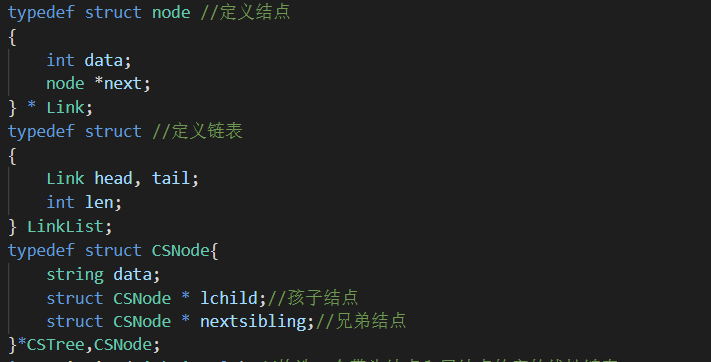
第五是广度优先搜索树的建立和输出



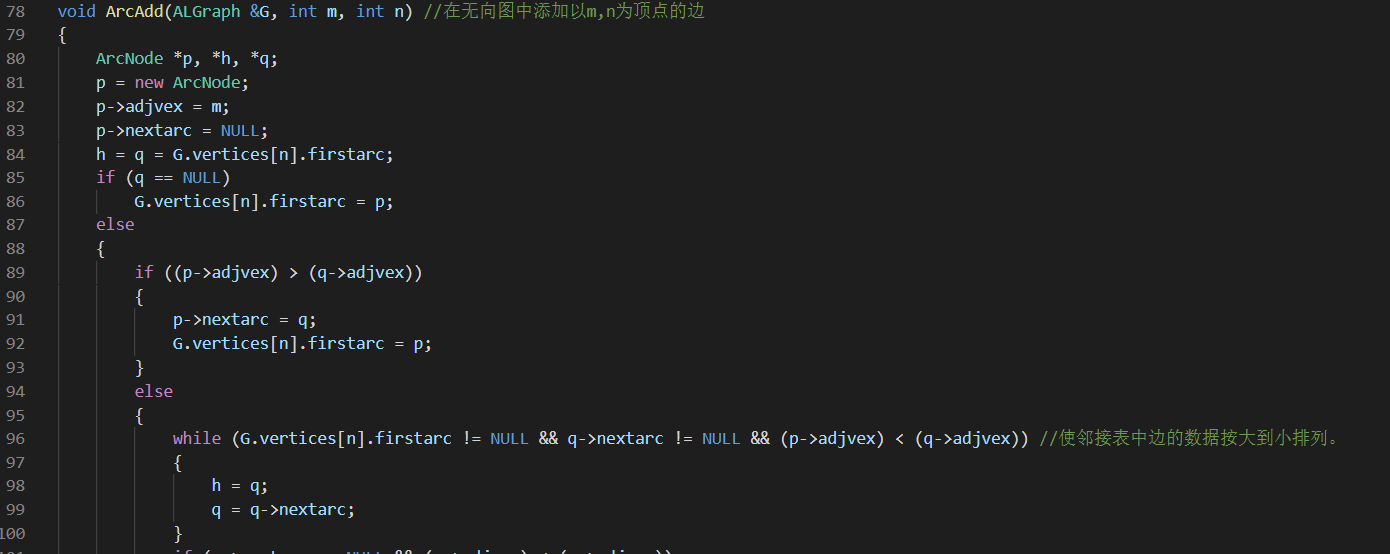
可以看到广度优先搜索树的层次并不多，也符合广度优先搜索树的特征，所以建树成功，并采用了括号的形式输出。

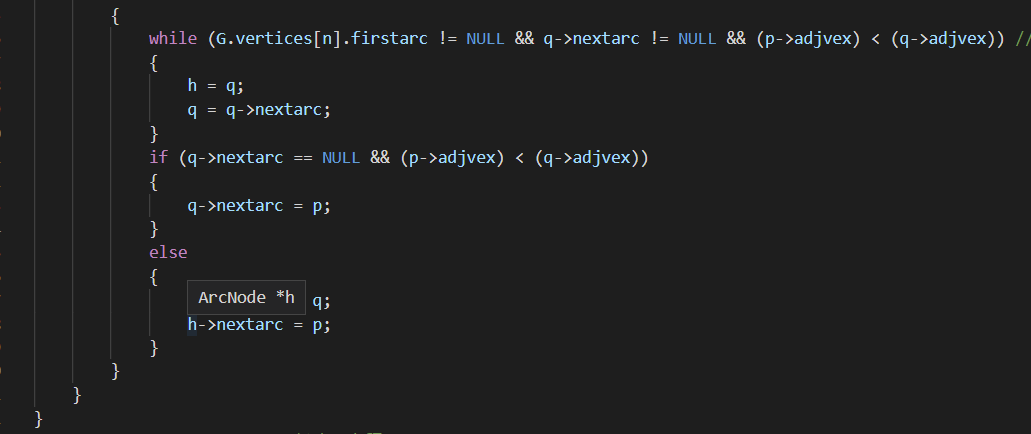
1. 部分关键代码及其说明。



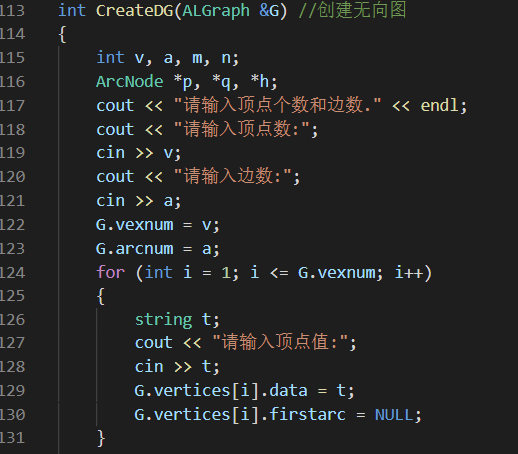


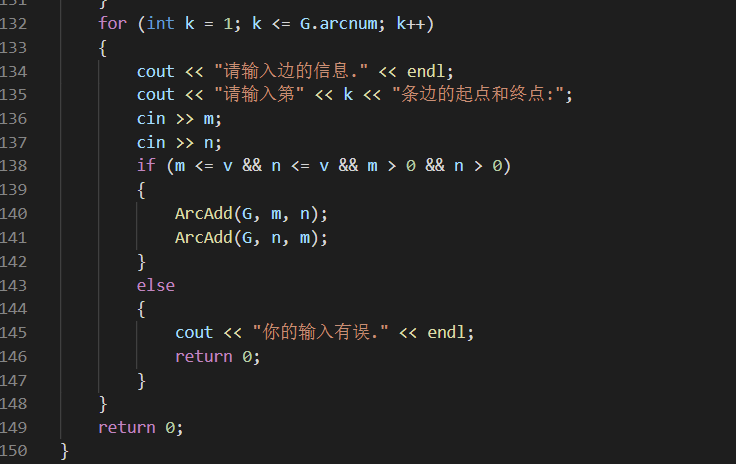
相关数据结构的定义，这里采用的是先定义顶点节点和边节点，然后顶点节点采用一个数组也就是vertices来进行表示顶点集合，然后每个顶点都有一条开始的边，这里定义为firstarc，对于每一条边，也有下一条边，定义为nextarc，所以无向图整个就可以定义成一个顶点集合和两个存储边数和顶点树的集合。



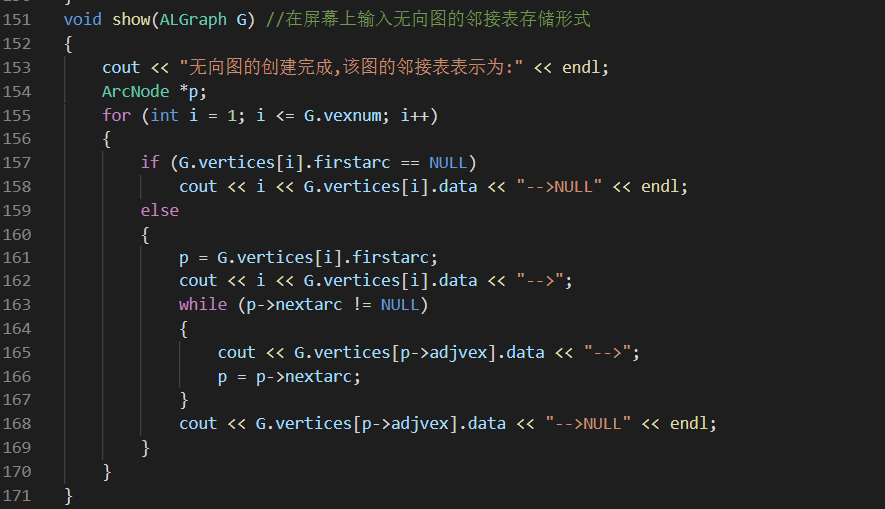


这里是建图的一个关键步骤，也就是在图中加入边，具体主要分为加入一条边，将边的起始顶点和点的开始的边，包括其他边如果有领接的话在进行更新，并且将领接表也进行一次更新。

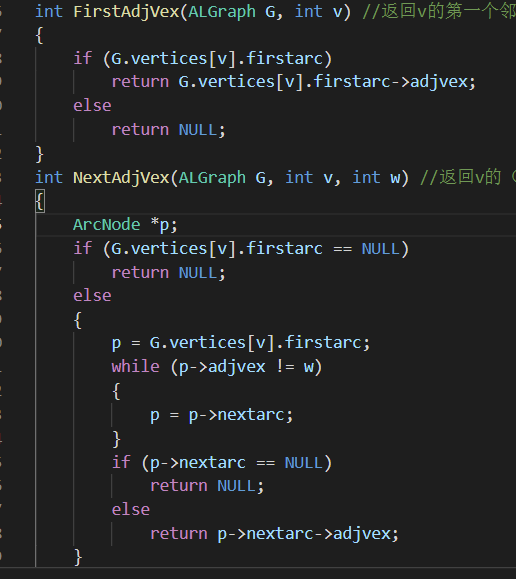




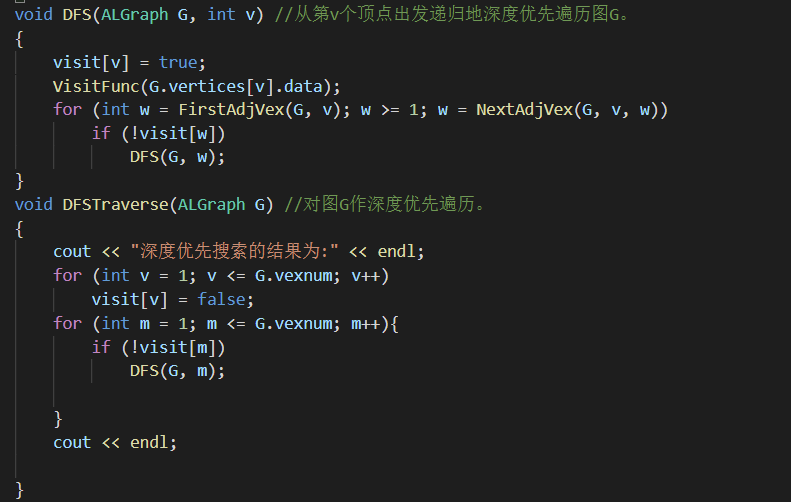
这里就是具体的表示输入提示和add上来了，其实也就是不断的调用了，一开始输入的是顶点集合，只需要将顶点数++，然后顶点的data更新，后面就是不断调用上面的增加边的函数，然后对于一些越界或者有误的情况进行判断并输出错误。



这个函数主要是输出领接表了，其实就是在每一个顶点开始进行一波顶点到边，边到领接边的不断链接了，然后注意一下到NULL的时候输出NULL，然后中间加上箭头来表示美观就好了。



两个辅助函数，具体因为定义里面没有关于顶点领接的定义，然后用函数通过边的操作来实现，方便下面的遍历和建树操作。

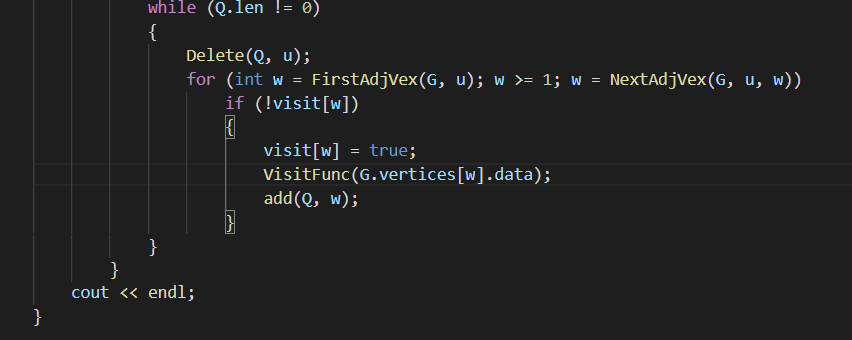


第一种深度优先遍历，采用递归的方式

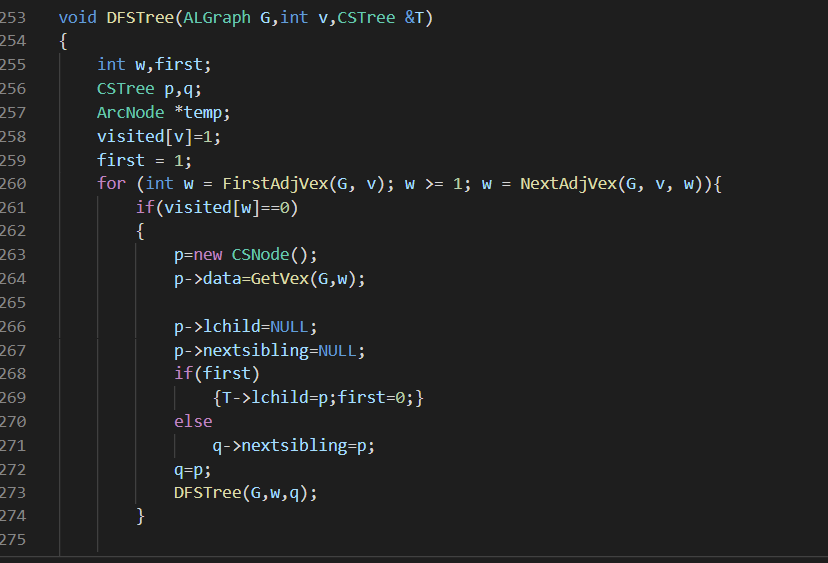


第二种深度优先遍历，采用非递归的方式，也就是栈的操作。

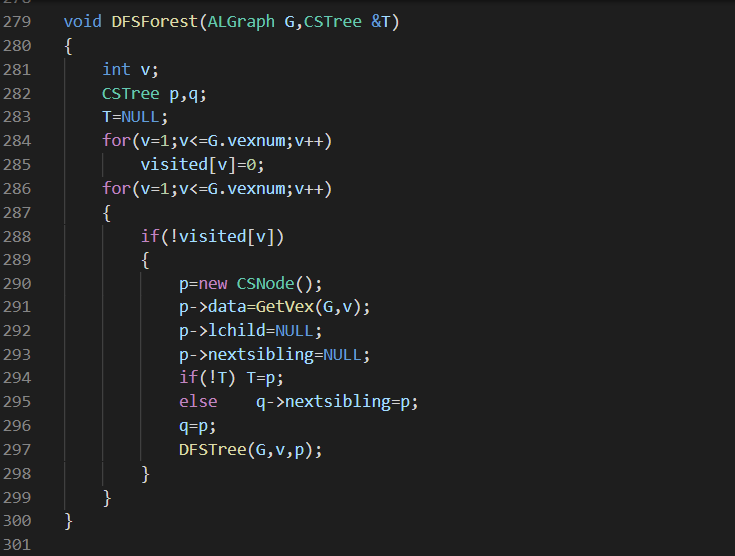


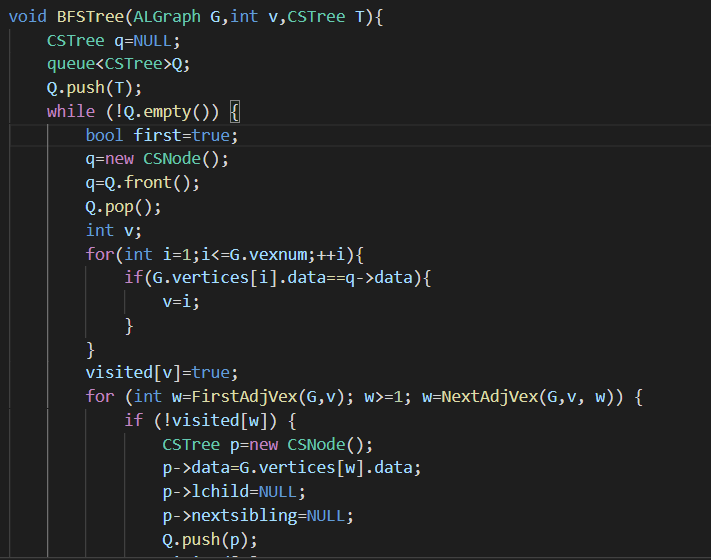


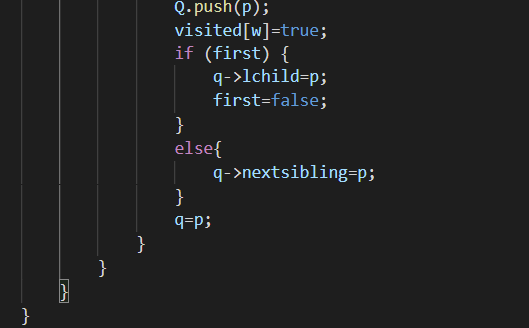
广度优先遍历，采用的是一个辅助队列的形式，具体上就是通过队列操作，判断条件是队列是否为空，然后通过对于每一个点的遍历顺应到边，然后边不断取领接的边来进队，最后通过队列来进行输出。



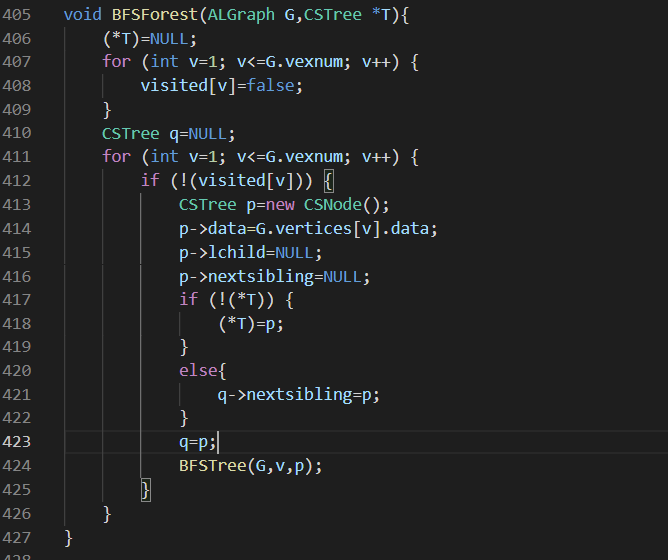
建深度优先树操作，这里首先是关于在v的位置来建树的操作，然后如果要建成森林的话就是进行一次遍历就可以了。



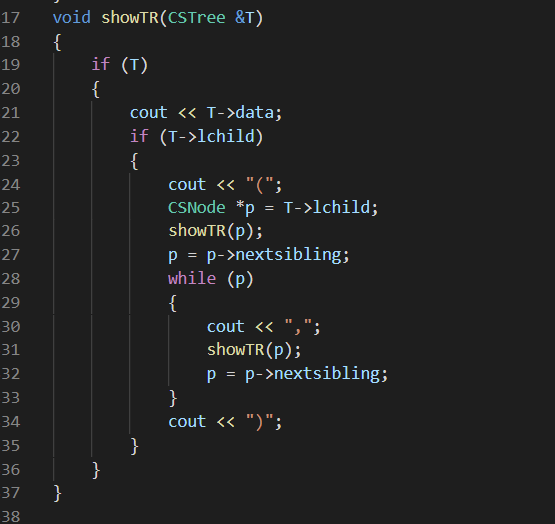




这里是具体的bfs建树的过程，主要也和上面的bfs遍历一样通过队列的操作，在遍历的同时通过层次性的关系来在树上加上儿子节点和兄弟节点，从而得到新的树，下面是加上一层遍历从而生成森林的代码



这里和上面的区别就是加上了具体的操作了，具体是把每一个visited都置为false，然后对每一个顶点集合里面的数据进行遍历，从而生成森林。



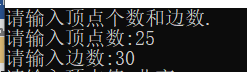
这是关于树形输出部分的代码，主要的就是递归调用了，首先是在取一个lchild，然后不断递归进行递归一层就加上一个括号，每一层之间通过nextsibling指针进行同层次的遍历操作，从而实现整个遍历。

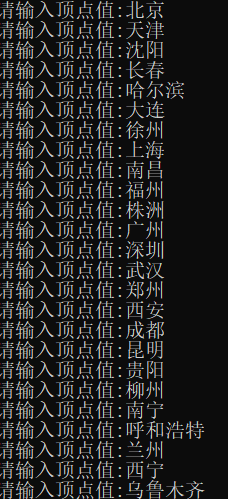
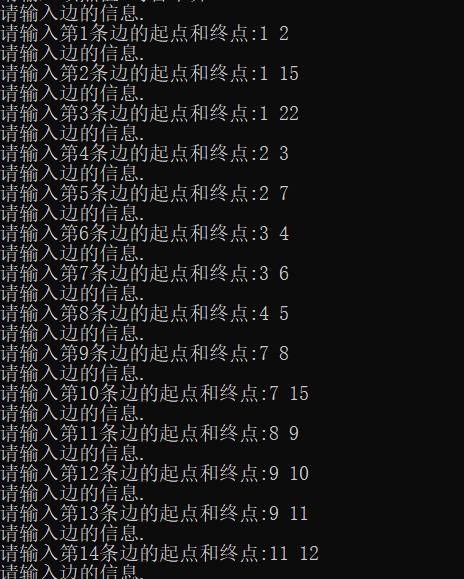




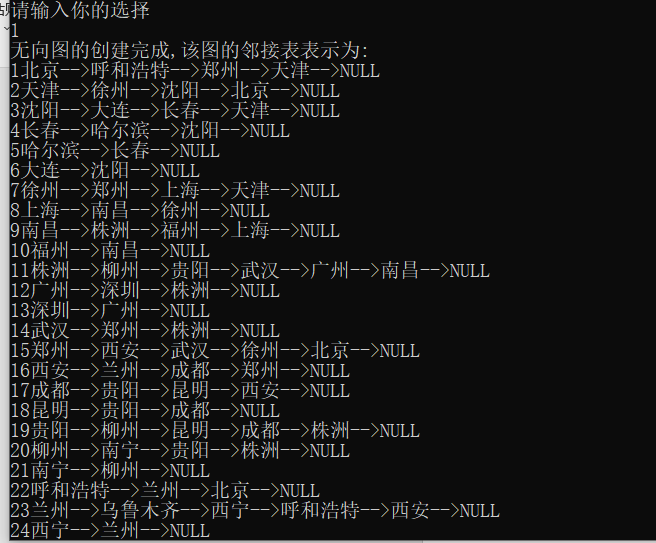
这里就是具体的主函数了，采用的也是menu的方式，开始通过都需要的输入顶点和边集合的过程然后进入新的menu，选择合适的操作，并加上中文注释输出。

4、程序运行方式简要说明。

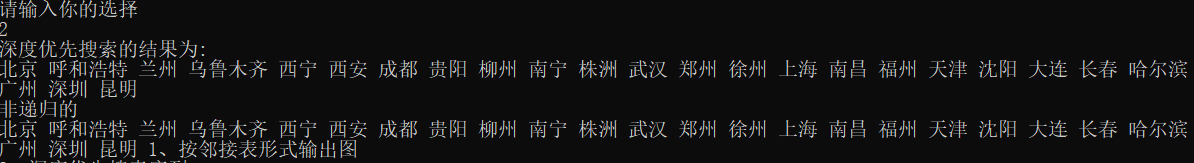


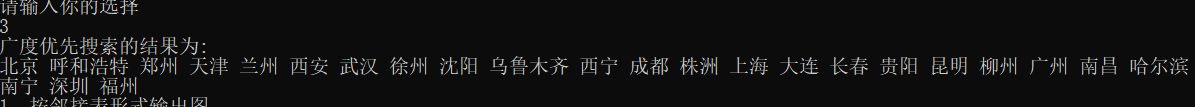
首先调用了建图的模块，先输入顶点数和边数，从而在图这个struct里面定义完成，然后调用的是对于顶点集合的赋值函数，也就是输入每个顶点集合的data，最后就是输入每一个边，调用addarc函数不断循环，生成整张图。



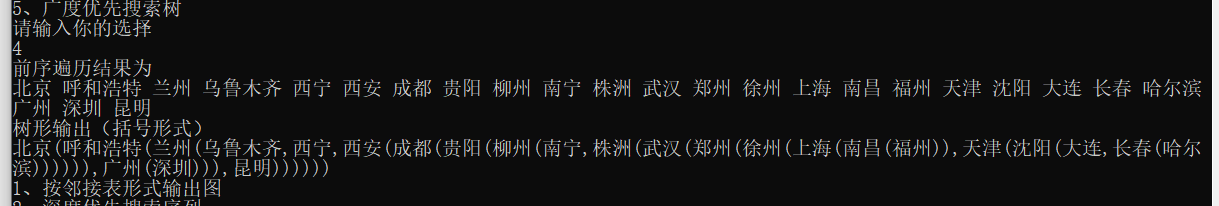
第一个选项调用的是show函数，通过对于每一个顶点的关于边的遍历，从而得到有关的链表，然后用链表的形式输出。



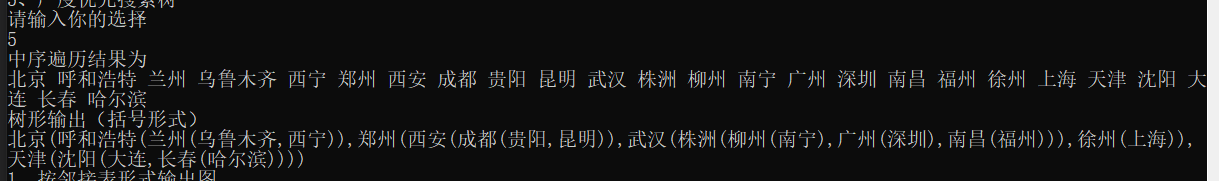
这里调用的就是dfstraverse和dfs2两个函数了，分别是递归的和非递归的版本。



Bfstraverse函数的调用



先是dfs建树，也就是dfsforest函数的调用，然后前序遍历，preordertravese函数的调用，括号的树形输出showTR函数的调用



先是bfs建树，也就是bfsforest函数的调用，然后中序遍历，inordertravese函数的调用，括号的树形输出showTR函数的调用