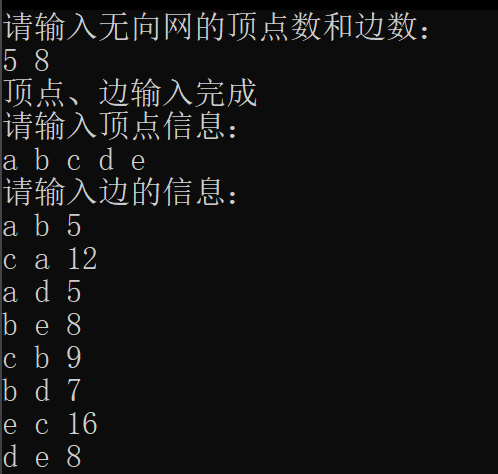
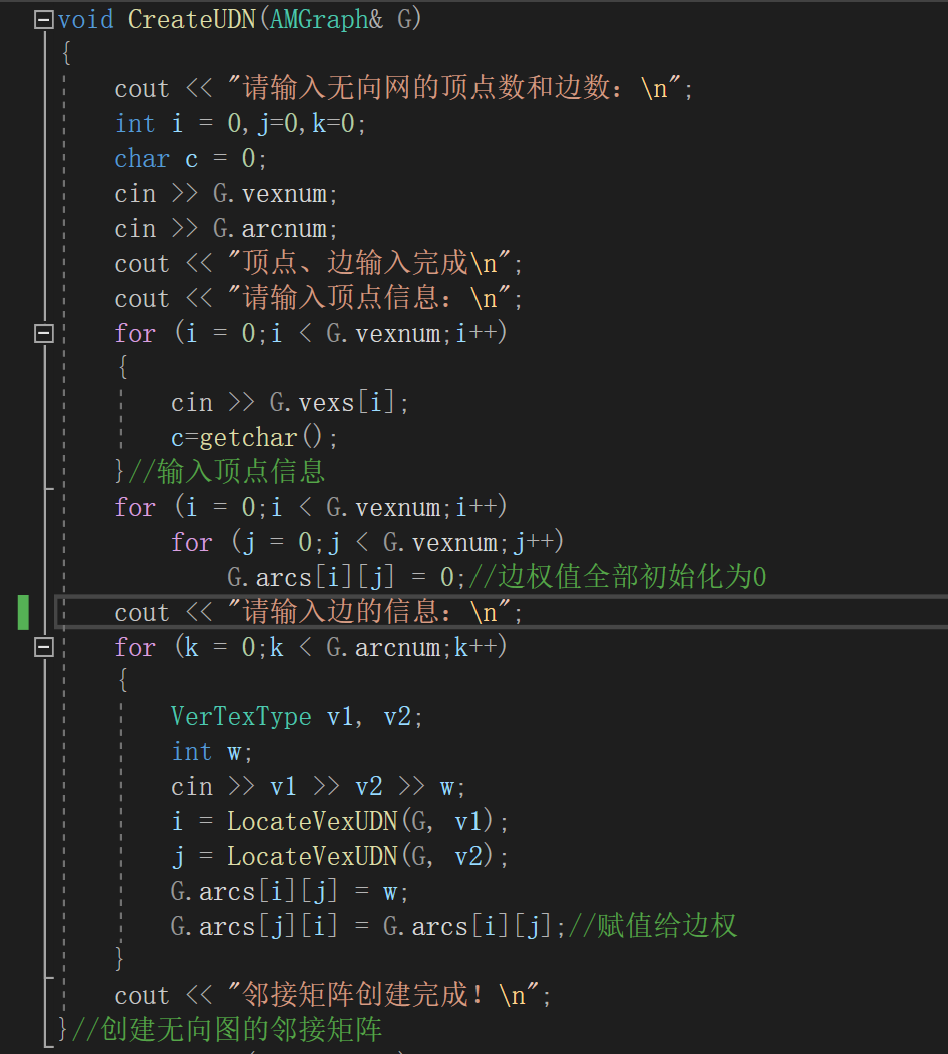
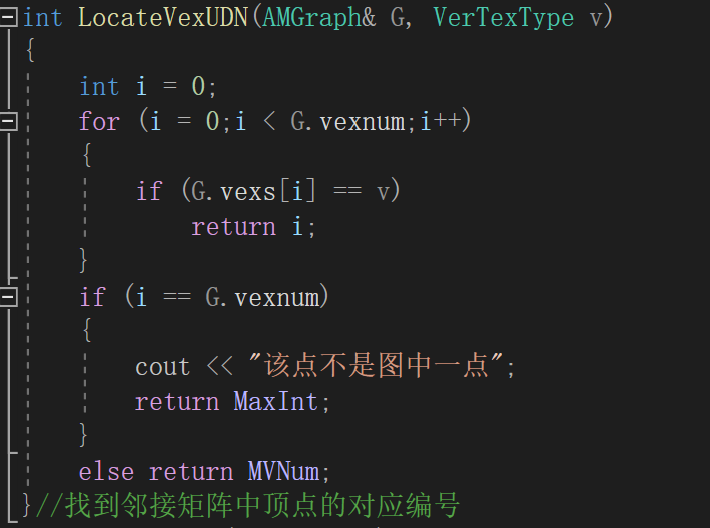
Lab3实验报告

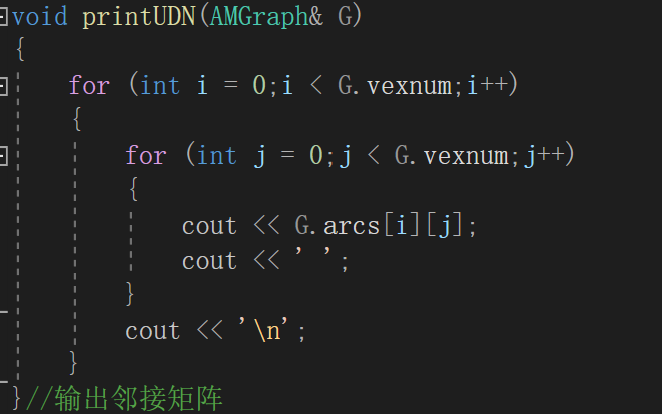
根据lab3评分细则中的样例输入，如图：



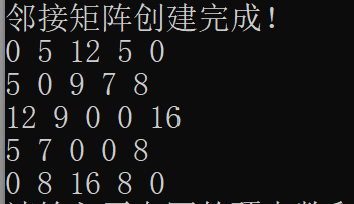
1. 图的创建
2. 邻接矩阵







对于输入的边的信息，找到对应端点在所有点中的序号，将对应n\*n的矩阵中的两个对应元素都赋值为边权值，记录完所有边以后再输出矩阵，结果如下：

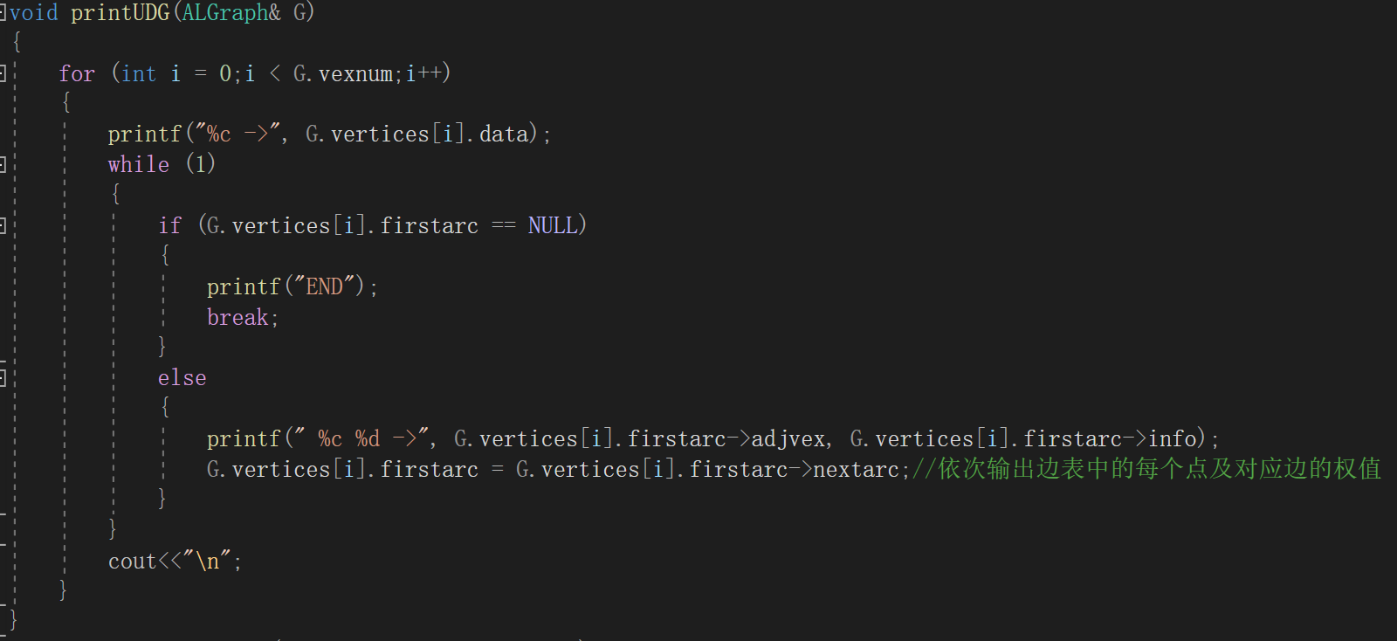


1. 邻接表

创建边节点和表头结点表的结构体，把输入的每条边的两个顶点A和B的第一个边结点分别修改为另一个结点B（A)。把对应的边权值也存入边结点的相关信息中。



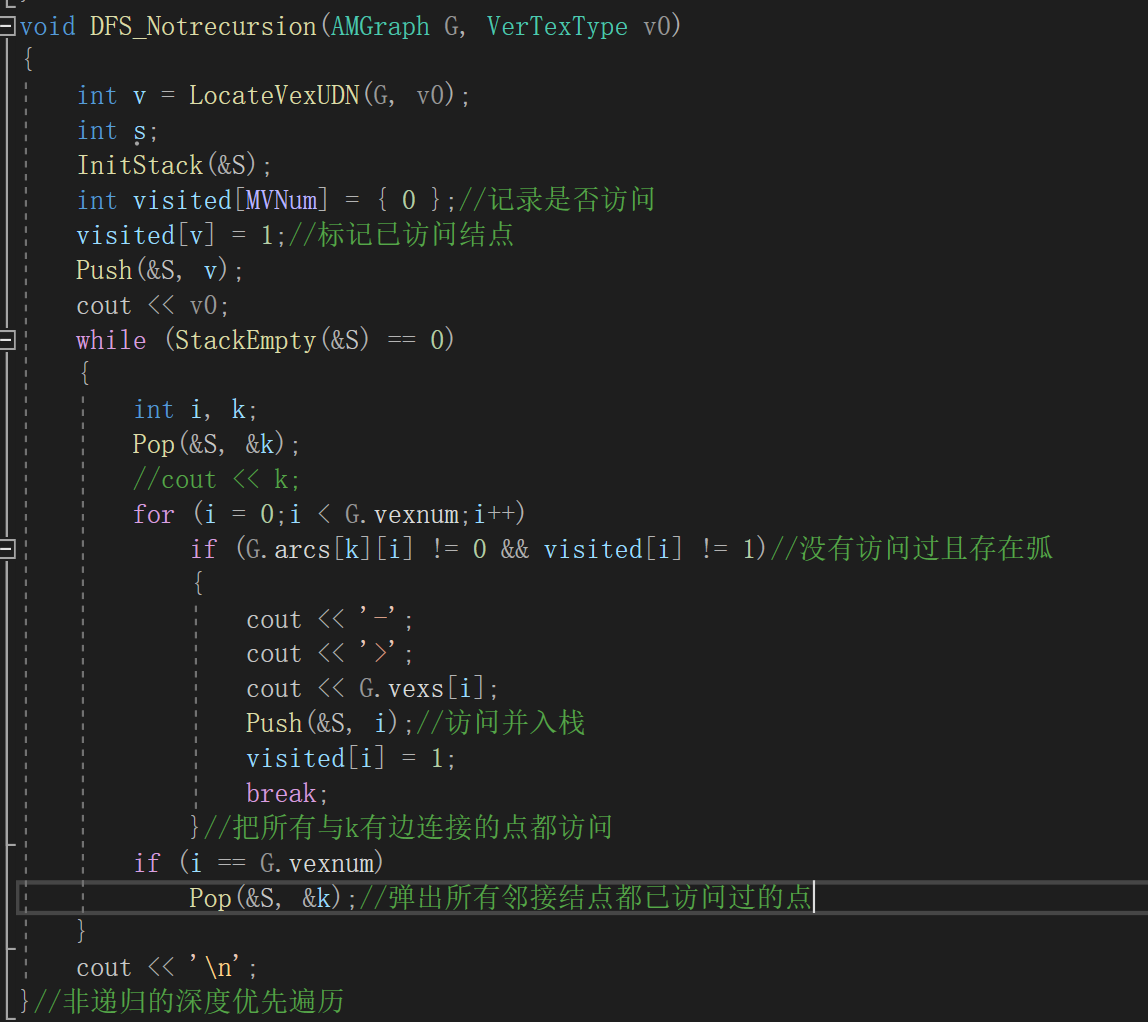


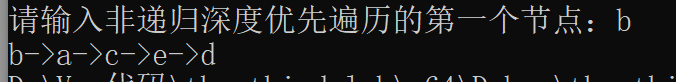


1. 图的遍历
2. 非递归的深度优先遍历

因为深度优先需要先进后出，所以用栈作为储存结构

具体过程如下:第一个顶点先入栈，然后进入while循环（循环条件为栈不空），循环中pop栈顶元素，遍历邻接矩阵，找到第一个与这个元素相邻且没有被访问到的节点，输出这个点、标记此点为已遍历并入栈，如果遍历了所有的点都没有找到满足条件的点，则把这个元素出栈并进入下一个while循环；否则，直接进入下一个while循环

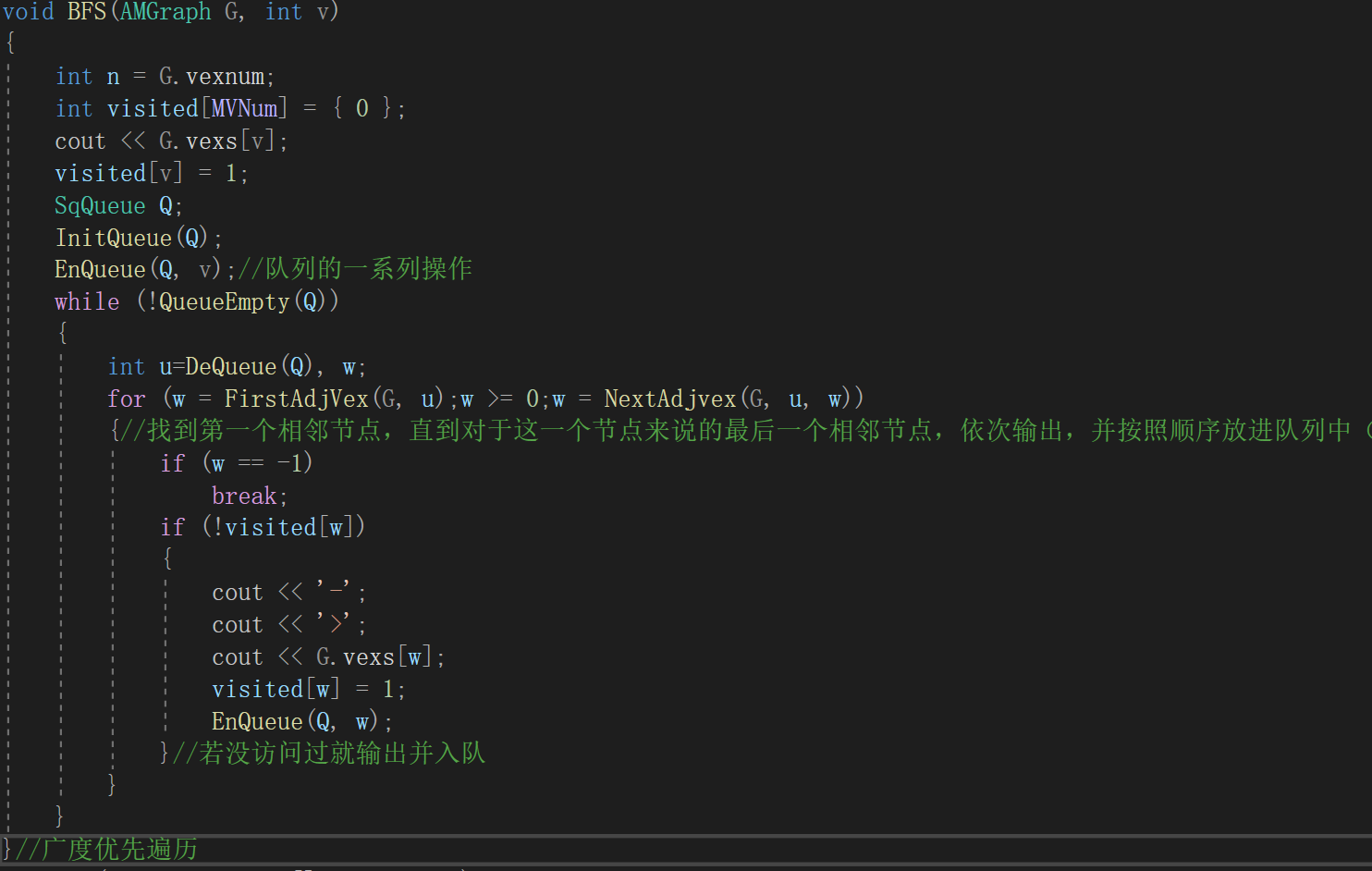


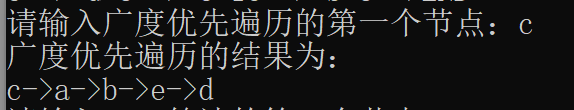


1. 广度优先遍历

因为广度优先需要先进先出，所以用队列作为储存结构

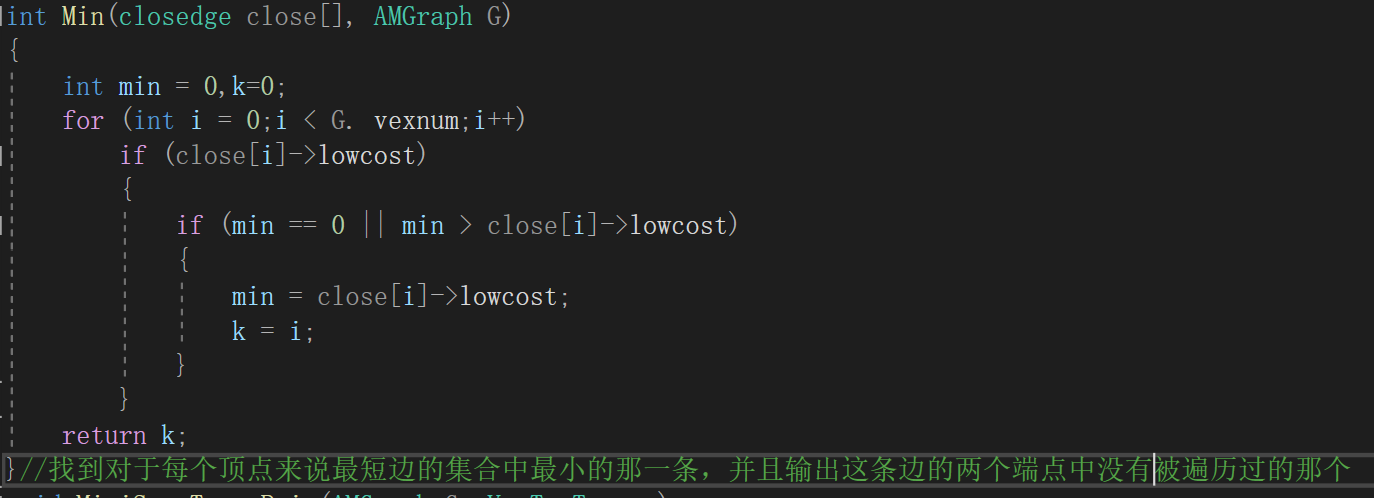
队列非空的时候，一个元素出队。对于这个元素，遍历，找到第一个相邻节点，直到对于这一个节点来说的最后一个相邻节点，依次判断是否被访问过，若没有被访问，则输出，并按照顺序放进队列中（因为先遍历到的节点的相邻节点要先输出，符合先进先出）。

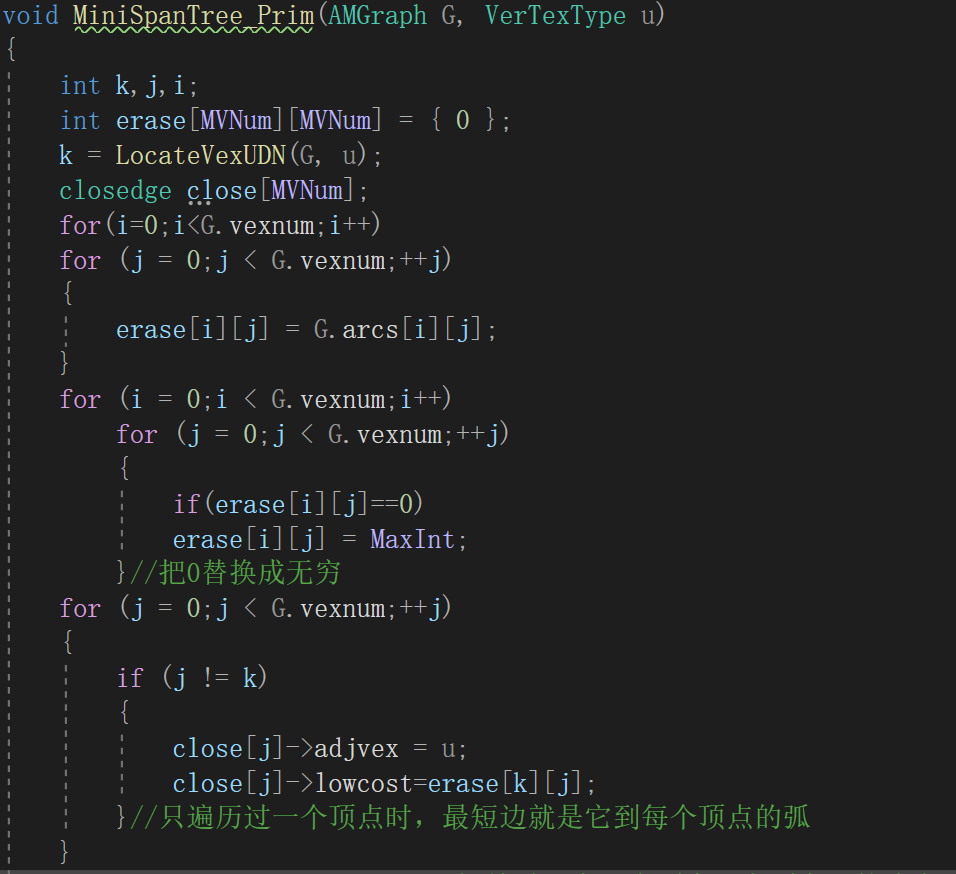


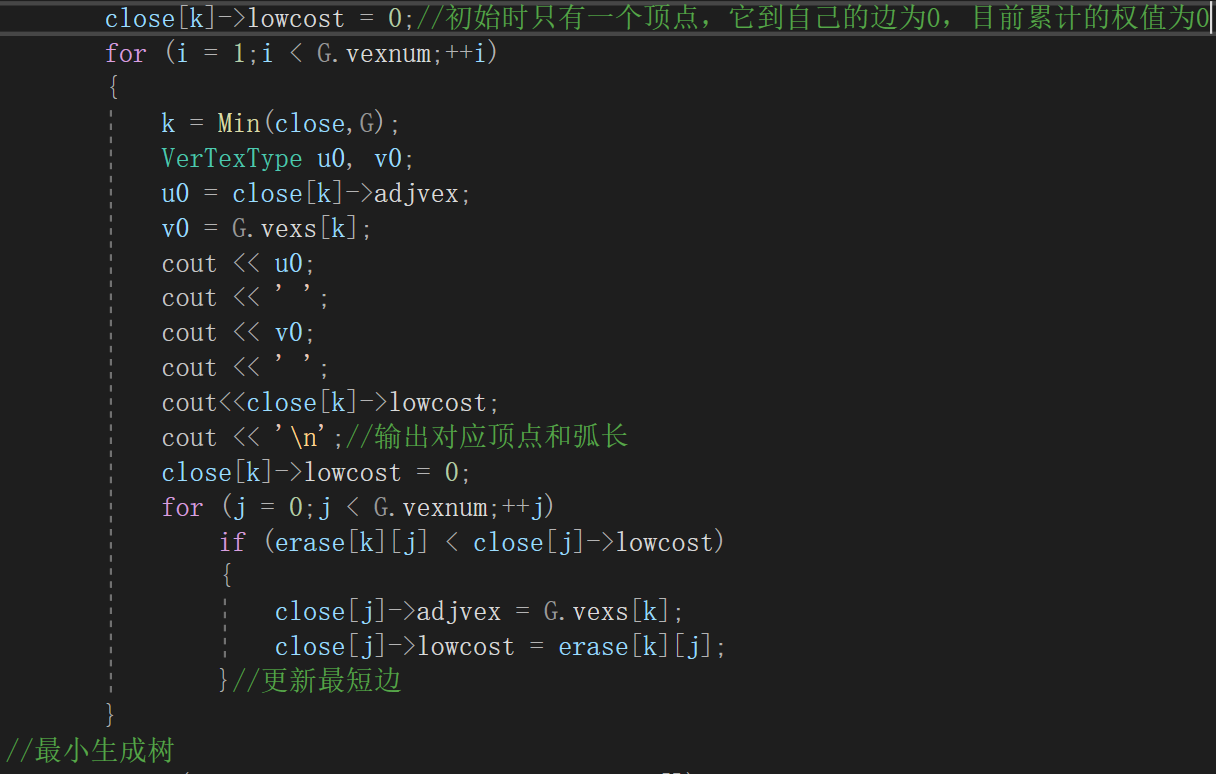


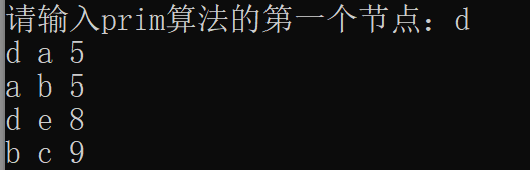
1. Prim算法求最小生成树

构造结构体close[]，用于对于目前没有加入到生成树中的顶点，储存它们各自与已经加入到生成树中的顶点之间的最小边以及这些边的另一个端点。每次生成树中新加入一个点，都需要更新对应的最短边，并把新加入点的最短边赋值为0。每次循环都找出现存的非最短边中最短的一个，输出所属的顶点，直至循环次数等于总的顶点数为止。









1. 图的最短路径

分别用数组D[ ]、S[ ]、Path[ ]存储当前已知的最短路径长度、标记最短路径是否已经被确定、最短路径的前驱（即前一个顶点）

对于与输入顶点直接相连的点，最短路径即是输入顶点和该点之间的边权值，前驱为输入顶点，对于的S都赋值为true；对于其他点，把当前的最短路径都记为MaxInt。

对于还没有确定最短路径的点A1、A2……，循环对比当前记录的最短路径和（已经确定最短路径的点的最短路径长加两点之间距离），如果后者更小，则把Ai的当前最短路径更新为这个更小的值，并更改前驱。

完成所有的最短路径记录后，用FindPath函数递归输出所有最短路径上的节点。（其实就是因为Path数组储存的是前驱，所以要倒过来使用Path，于是想到了递归输出。）

