**图的应用实验报告**

姓名：陆明奇 学号：2050283 日期：2021年12月11日

1. **涉及数据结构和相关背景**

通过图（无向图或有向图）中所有边一次且仅一次行遍图中所有顶点的通路称为欧拉通路，通过图中所有边一次且仅一次行遍所有顶点的回路称为欧拉回路。具有欧拉回路的图称为欧拉图，具有欧拉通路而无欧拉回路的图称为半欧拉图。

无向图是否具有欧拉通路或回路的判定：

  （1）：G有欧拉通路的充分必要条件为：G 连通，G中只有两个奇度顶点(它们分别是欧拉通路的两个端点)。

（2）： G有欧拉回路(G为欧拉图)：G连通，G中均为偶度顶点。

有向图是否具有欧拉通路或回路的判定：

（1）： D有欧拉通路：D连通，除两个顶点外，其余顶点的入度均等于出度，这两个特殊的顶点中，一个顶点的入度比出度大1，另一个顶点的入度比出度小1。

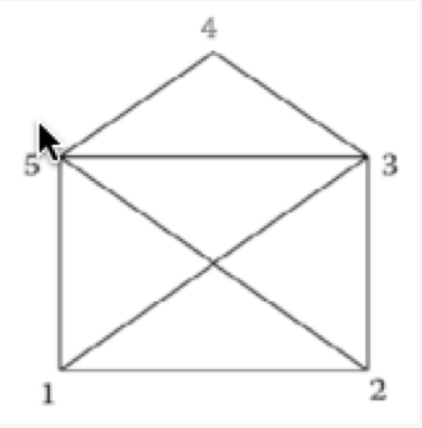
（2）： D有欧拉回路(D为欧拉图)：D连通，D中所有顶点的入度等于出度。 (注意：这里说有向图连通，说的是有向图是弱连通图。即把有向图中的边变成无向边，只要该图连通，那么原有向图即是弱连通图。)

**2. 实验内容**

**2.1 问题描述**

圣诞节马上到了，我们用一笔画画出圣诞老人的房子吧。现在的问题是，一共有多少种画法呢？

请你写一个程序，从下图所示房子的左下角（数字1）开始，按照节点递增顺序，输出所有可以一笔画完的顺序，要求一条边不能画两次。



**2.2 数据结构设计**

程序采用邻接矩阵保存图，一维数组存储一笔画的路径。

**2.3功能说明（函数、类）**

void DFS\_draw(MGraph G, int\* s, int i)

{

if (no == 1 && i != 1) {//递归终止条件

return;

}

s[no] = i;//保存路径

no++;//开始寻找下一段路径

for (int k = 1; k <= G.vexnum; k++) {

if (G.arcs[i][k] != 0) {

G.arcs[i][k] = 0;

G.arcs[k][i] = 0;

DFS\_draw(G, s, k);//搜索下一个点

G.arcs[i][k] = 1;//回退

G.arcs[k][i] = 1;

}

}

if (no == G.arcnum + 1) {//输出路径

cout << "第" << ++num << "种走法" << endl;

for (int k = 1; k <= G.arcnum; k++) {

cout << k << ':' << s[k] << "->" << s[k + 1] << endl;

}

cout << endl;

}

no--;//回退

return;

}

功能：输出所有画法的数量并输出所有一笔画完顺序

Para G: 以邻接矩阵为存储形式的图

Para s：存储路径的一维数组

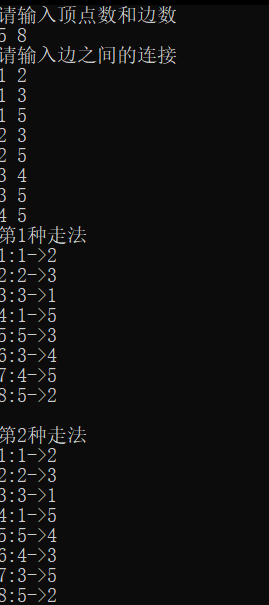
Para i：搜索的起点

返回值：无返回值

复杂度：判断是否存在欧拉路径或求解出一条欧拉路径的算法复杂度与无向图边数成线性关系 O(n + m)，n 为总点数，m 为总边数。但是 DFS 递归暴力搜索所有可能路径并输出的过程，算法复杂度难以分析，实质上是一个阶乘级别的NPC 问题，因为欧拉路径总数的理论上界趋近边数的全排列数 m！。调试过程中可以发现，输出 5个点以上的稠密图的所有欧拉路径就已经相当耗时。

**2.4 调试分析（遇到的问题和解决方法）**

在进行DFS搜索时，可能会出现找不到下一个顶点的情况，此时需要回退到上一步，回退的操作有将邻接矩阵arcs[i][j和arcs[j][i]重新置为1，记录路径的一维数组的下标no减1。若缺少了回退操作，则会导致程序出现错误。在DFS函数的开头还应该设置退出递归的条件。

运行结果

**3.实验总结**

一笔画问题的实质为无向图的欧拉路径问题。无向图欧拉路径有一个重要的 结论是：一个图存在欧拉路径，当且仅当其为连通图，且所有点度数均为偶 数，或者有且仅有两个点度数为奇数。若所有点度数均为偶数，则从每个点 出发，均可找到能返回自身的欧拉路径(欧拉回路)若有且仅有两个点度数为 奇数，则所有路径都是两个奇点一个作为起点，另一个作为终点。一条路径 的逆路径必然也是一条欧拉路径，因此以两点做起点的欧拉路径数量相同。 本题中，1 和 2 是奇点，因此只有 1,2 出发有一笔画路径。且各有 44 条。