作业 PA4 实验报告

姓名：刘博洋 学号：2153538 日期：2022年12月2日

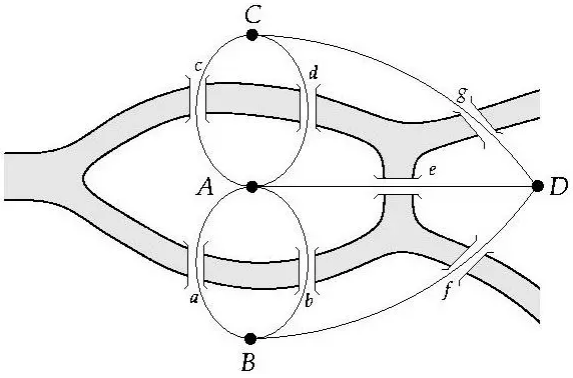
1.涉及数据结构和相关背景

图结构是一种非常重要的非线性结构，它是树的一般化和延伸，图在日常生活中有着广泛的应用场景，比如交通运输网，地铁网络，社交网络，计算机中的状态执行（自动机）等等都可以抽象成图结构。

图可以分为有向图和无向图，存储结构一般为邻接表、邻接矩阵等等，涉及到的主要算法有图的遍历（深度、广度优先搜索）、最小生成树问题、拓扑排序、关键路径、最短路径等等。

本次实验主要针对的是一种特殊的图，我们称为欧拉图，欧拉图起源于起源于18世纪瑞士数学家欧拉发表了图论的第一篇论文“哥尼斯堡七桥问题”。在当时的哥尼斯堡城有一条横贯全市的普雷格尔河，河中的两个岛与两岸用七座桥连结起来。当时那里的居民热衷于一个难题：有游人怎样不重复地走遍七桥，最后回到出发点。

为了解决这个问题，欧拉用 A,B,C,D 4个字母代替陆地，作为 4 个顶点，将联结两块陆地的桥用相应的线段表示，于是哥尼斯堡七桥问题就变成了在下列图中，是否存在从某一点除非经过每条边一次且仅一次，和经过所有的顶点的回路问题了。欧拉在论文中指出，这样的回路是不存在的。



下列给出一些简单的定义：

欧拉路径（欧拉通路）：通过图中所有边的简单路。（换句话说，每条边都通过且仅通过一次）也叫”一笔画”问题。
欧拉回路：闭合的欧拉路径。（即一个环，保证每条边都通过且仅通过一次）
欧拉图：包含欧拉回路的图。

判断有向连通图存在欧拉路径的条件

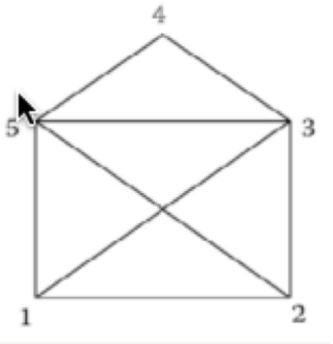
若所有顶点的入度等于出度，则能够找到从任意顶点出发的欧拉回路。反之也成立。
若有且仅有两个顶点入度不等于出度，其中一个顶点入度比出度大1，记为V 1 ，另一个顶点入度比出度小1，记为V 2 ，则只能够找到欧拉路径（路径从顶点V 2出发，到顶点V 1 V_1V 1 结束）。反之也成立。

2. 实验内容

2.1 欧拉路径（一笔画问题）

2.1.1 问题描述

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顶点顺序（欧拉路径），要求所有的边恰好都只画一次。例如，123153452就是其中的一条路径。



2.1.2 基本要求

图的存储可以使用邻接矩阵map[][]存储此图，其中map的对角线(map[1][1] map[2][2], map[3][3], map[4][4], map[5][5])和map[1][4], map[4][1], map[2][4], map[4][2]为0，其余元素为1

程序要添加适当的注释，程序的书写要采用 缩进格式 。
在实验报告中给出算法的复杂度分析。

2.1.3 数据结构设计

本题由于是一个确定的图，所以直接采用邻接矩阵作为存储结构，初始化map，由于后面的遍历操作需要删除边，所以我们需要建立一个数组cop来复制map并在每一次遍历后更新为map。同时用一个一维数组ans存放顺序遍历的顶点编号，用一个cnt记录递归遍历的次数便于存入ans。主要的存储结构如下

```
int map[6][6] =
{
    {0,0,0,0,0,0},
    {0,0,1,1,0,1},
    {0,1,0,1,0,1},
    {0,1,1,0,1,1},
    {0,0,0,1,0,1},
    {0,1,1,1,1,0},
};
int cop[6][6];
int ans[9] = { 0 };
int cnt = 0;
stack<int> T;
```

而本题的核心算法，即如何在图中找到每条欧拉路径，经过自己的摸索和一些资料的查阅，我采用的是Hierholzer算法。

Hierholzer 算法过程：

1. 选择任一顶点为起点，遍历所有相邻边。
2. 深度搜索，访问相邻顶点。将经过的边都删除。
3. 如果当前顶点没有相邻边，则将顶点入栈。
4. 栈中的顶点倒序输出，就是从起点出发的欧拉回路。

伪代码为

```
void dfs(int u)
{
   for {
            从u出发的所有边 如果存在且终点为i
             删除边ui
             dfs(i)
             push(i)
    }
}
//最后再出栈倒序得到ans
```

为了证明该算法的有效性，下说明两条性质。

性质一:

如果该图为欧拉图，则栈底的必定为起点。如果该图为半欧拉图，则栈底部存储的是与起点不同的另外一个奇度数顶点。

证明:

当顶点入栈时，说明当前所在顶点没有相邻边。

考虑到从起点出发到当前结点的路径中，除了起点和当前顶点外，其他的顶点都失去了偶数度数(入度与出度一一对应)。

如果起点和当前顶点不同，那么两者都失去了奇数度数。

如果图中包含欧拉回路，意味着所有顶点的初始度数都是偶数，而当前顶点的当前度数为0，表示当前顶点的初始度数必定是奇数，产生矛盾，因此假设不成立。当前顶点就是起点。

同样地，对于欧拉路径，当前顶点不可能是起点，否则起点的度数就是偶数，而欧拉路径中起点和终点的度数一定是奇数。因此，当前顶点不是起点，但是度数也是奇数，所以一定是终点。

性质二:

如果该图为欧拉图(/半欧拉图)，则栈中的自底到顶第 n 个顶点就是欧拉回路(/欧拉路径)上的第 n 个顶点。

证明:

在此只证明栈中相邻顶点在图中也为相邻顶点。因为模拟 Hierholzer 算法过程，可知该算法实际上就是在模拟“一笔画”过程，并且沿着画完的轨迹，从终点倒着逐一添加顶点到栈中。

并且主要以 n=2 的情况为例，后面的情况可以此类推。并且为了不用纠结于区分欧拉回路和欧拉路径，不妨以半欧拉图为例。

假设图中存在相邻的两顶点 V1,V2 ，并且深度搜索过程中，先访问 V2 随后访问了 V1 ，并且 V1 成为第一个入栈的顶点。由**性质一**可知， V1 就是欧拉路径上的起点(两个奇度数顶点任一可看作起点)。

根据 **Hierholzer 算法**，在遍历过程中，删除了途径的边，所以此时所有顶点的度数都为偶数。当然 deg(V2) 也是偶数，接下来就分类讨论。

如果 deg(V2)=0 ，也就是说当前顶点 V2 成为第二个入栈的顶点，那么 n=2 的情况就证毕了。

如果 deg(V2)>0，那么考虑当前包含 V2 的子图 G，显然 G 是一个欧拉图，那么当前以 V2 为起点继续实施 Hierholzer 算法遍历剩下的相邻边，根据**性质一**， V2 将会是 G 中第一个入栈的顶点。也就是，V2 是原图中第二个入栈的顶点。

综上所述，以此类推， Vn−1 入栈前最后接触过的 Vn 将会是第 n 个入栈的顶点，再结合直观理解， Vn 就是路径上的第 n 个顶点。

由于本题显然是一个欧拉图，所以可以直接对每一个顶点作为起点进行深度优先搜索输出ans，但要注意的是cop数组每一次深度优先搜索后都要更新为与map相同，ans也要全部置为0。

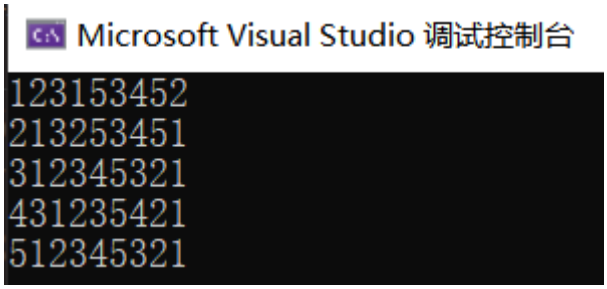
2.1.4功能说明（函数、类）

```
void Hierholzer(int i)//深度优先搜索同时删边、顶点入栈
{
    int j;
    for ( j = 1;j <= 5;j++)
    {
        if (cop[i][j] == 1)
        {
            cop[i][j] = 0;
            cop[j][i] = 0; //矩阵中两个元素都要置零
            Hierholzer(j);
            T.push(j);//存在边则入栈
        }
    }

}//该算法的时间复杂度为O（V+E），十分高效

for (int n = 1;n <= 5;n++)
    {
        Hierholzer(n);
        cout << n;//第一个点是起点，而入栈是从第二个点开始的
        while (T.empty() == 0)
        {
            cout << T.top();
            T.pop();
        }
        cout << endl;
        for (int i = 0;i < 6;i++)
        {
            for (int j = 0;j < 6;j++)
            {
                cop[i][j] = map[i][j];//把cop数组更新为与map一致
            }
        }
        memset(ans, 0, sizeof(ans));//重置ans数组
    }
```

运行结果



所以从每个顶点都可以得到欧拉路径，分别为：

123153452
213253451
312345321
431235421
512345321

2.1.5 调试分析（遇到的问题和解决方法）

本题遇到的最大问题就是如何采用深度优先搜索结合栈来处理边的访问，开始时没有利用栈，没有彻底理解算法的过程，导致输出的是倒序的，其实由于是深度优先搜索，最先输出的是最深的顶点，而在模拟一笔画的过程中，应该输出的顺序是从浅到深的，所以应该用栈来存储。而且在每一轮遍历中，要更新ans和cop数组重新进行遍历和输出。

2.1.6 其他方法

经过与同学的讨论和上网资料的搜索，发现对于欧拉路径的求解还可以借助以下几种方法，此处列出供参考

方法1：回溯法

遍历所有边，每遍历一个边则删除该边继续遍历，如果中间过程还没有遍历所有边就无法继续遍历了，则往前回溯继续遍历。该算法时间复杂度是指数级别。

方法2：弗罗莱(Fluery)算法

设G是一无向欧拉图，Fluery算法求解一条欧拉路径算法如下：

1. 任取G中一个点x
2. 选择一条从点x出发的边i（若i可以不为桥则不应将i选为桥），设i连向点y，删除i，然后：
   1）i不为桥，走到点y；
   2）i迫不得已必须为桥，走到点y并删去点x（因为删去i后，点x将成为孤立点）。
3. 步骤2无法执行时停止算法
   当算法停止时，依次经过的点所得到的简单回路为图G的一条欧拉路径。

3.实验总结

本次实验从一个图论的问题出发，引入欧拉图、欧拉路径等等问题，十分有趣并且引导我们去思考实现的算法。在学习过程中总会遇到不同的实际问题，也不是所有问题都能套用书上的模版，更多的问题需要靠我们自己去挖掘去思考出新的思想和思路，才能开拓我们的境界，提升我们的水平，本题代码并不难，但是给我们的启发和思考还是很多的，关于欧拉图和欧拉路径、欧拉回路等等还有更多有趣的算法和问题等待我们去思考和解决。