PA4 实验报告

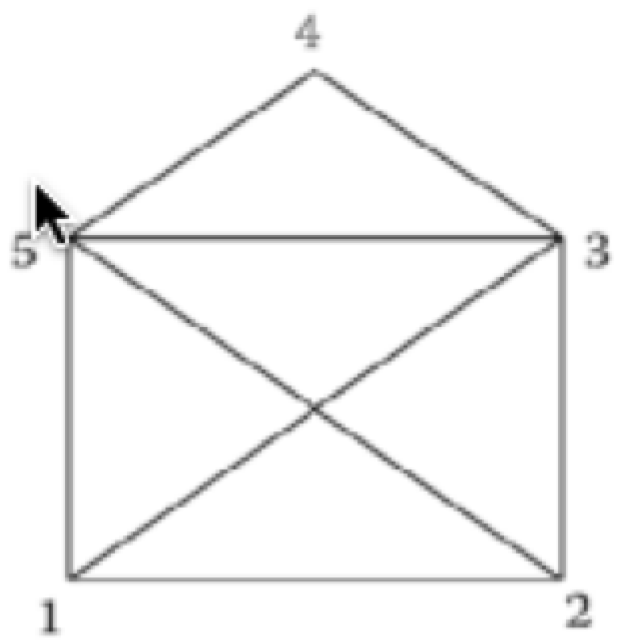
1. 涉及数据结构和相关背景

- 图的存储
- 图的遍历
    - `dfs`进行图的遍历
    - `dfs`递归过程
    - `dfs`深度控制

2. 实验内容

2.1 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。



2.2 基本要求

- 图的存储：邻接矩阵
- 进行`dfs`遍历方式进行图的遍历

2.3 数据结构设计

- 本图求解
- 邻接矩阵进行图的存储

```
int map[10][10];
```

- 一笔画方式计数

```
int cnt = 0;
```

- 控制深度进行`dfs`，达到相应深度进行一笔画顺序输出

2.4 功能说明

- `dfs`遍历

```
void dfs(int x, int k, string s)
{
    if (k > 8)
    {
        cnt++;
        cout << s << endl;
        return;
    }

    for (int y = 1; y <= 5; y++)
    {
        if (map[x][y] == 1)
        {
            map[x][y] = 0;
            map[y][x] = 0;

            /*输出格式控制*/
            if (k == 1)
                dfs(y, k + 1, s  + to_string(y));
            else
                dfs(y, k + 1, s + " -> " + to_string(y));

            map[x][y] = 1;
            map[y][x] = 1;
        }
    }
}
```

- 主函数调用

```
int main()
{
    for (int i = 0; i < 10; i++)
    {
        for (int j = 0; j < 10; j++)
```

```
        {
            if (i == j) map[i][j] = 0;
            else map[i][j] = 1;
        }
    }
    map[1][4] = 0;
    map[4][1] = 0;
    map[2][4] = 0;
    map[4][2] = 0;

    string s = "";
    dfs(1, 1, s);


    cout << endl << "一笔画总数为：" << cnt << endl;
    return 0;
}
```

2.7 调试分析

- 运行界面

Microsoft Visual Studio 调试控制台

```
1 -> 2 -> 3 -> 1 -> 5 -> 3 -> 4 -> 5 -> 2
1 -> 2 -> 3 -> 1 -> 5 -> 4 -> 3 -> 5 -> 2
1 -> 2 -> 3 -> 4 -> 5 -> 1 -> 3 -> 5 -> 2
1 -> 2 -> 3 -> 4 -> 5 -> 3 -> 1 -> 5 -> 2
1 -> 2 -> 3 -> 5 -> 1 -> 3 -> 4 -> 5 -> 2
1 -> 2 -> 3 -> 5 -> 4 -> 3 -> 1 -> 5 -> 2
1 -> 2 -> 5 -> 1 -> 3 -> 4 -> 5 -> 3 -> 2
1 -> 2 -> 5 -> 1 -> 3 -> 5 -> 4 -> 3 -> 2
1 -> 2 -> 5 -> 3 -> 1 -> 5 -> 4 -> 3 -> 2
1 -> 2 -> 5 -> 3 -> 4 -> 5 -> 1 -> 3 -> 2
1 -> 2 -> 5 -> 4 -> 3 -> 1 -> 5 -> 3 -> 2
1 -> 2 -> 5 -> 4 -> 3 -> 5 -> 1 -> 3 -> 2
1 -> 3 -> 2 -> 1 -> 5 -> 3 -> 4 -> 5 -> 2
1 -> 3 -> 2 -> 1 -> 5 -> 4 -> 3 -> 5 -> 2
1 -> 3 -> 2 -> 5 -> 3 -> 4 -> 5 -> 1 -> 2
1 -> 3 -> 2 -> 5 -> 4 -> 3 -> 5 -> 1 -> 2
1 -> 3 -> 4 -> 5 -> 1 -> 2 -> 3 -> 5 -> 2
1 -> 3 -> 4 -> 5 -> 1 -> 2 -> 5 -> 3 -> 2
1 -> 3 -> 4 -> 5 -> 2 -> 1 -> 5 -> 3 -> 2
1 -> 3 -> 4 -> 5 -> 2 -> 3 -> 5 -> 1 -> 2
1 -> 3 -> 4 -> 5 -> 3 -> 2 -> 1 -> 5 -> 2
1 -> 3 -> 4 -> 5 -> 3 -> 2 -> 5 -> 1 -> 2
1 -> 3 -> 5 -> 1 -> 2 -> 3 -> 4 -> 5 -> 2
1 -> 3 -> 5 -> 1 -> 2 -> 5 -> 4 -> 3 -> 2
1 -> 3 -> 5 -> 2 -> 1 -> 5 -> 4 -> 3 -> 2
1 -> 3 -> 5 -> 2 -> 3 -> 4 -> 5 -> 1 -> 2
1 -> 3 -> 5 -> 4 -> 3 -> 2 -> 1 -> 5 -> 2
1 -> 3 -> 5 -> 4 -> 3 -> 2 -> 5 -> 1 -> 2
1 -> 5 -> 2 -> 1 -> 3 -> 4 -> 5 -> 3 -> 2
1 -> 5 -> 2 -> 1 -> 3 -> 5 -> 4 -> 3 -> 2
1 -> 5 -> 2 -> 3 -> 4 -> 5 -> 3 -> 1 -> 2
1 -> 5 -> 2 -> 3 -> 5 -> 4 -> 3 -> 1 -> 2
1 -> 5 -> 3 -> 1 -> 2 -> 3 -> 4 -> 5 -> 2
1 -> 5 -> 3 -> 1 -> 2 -> 5 -> 4 -> 3 -> 2
1 -> 5 -> 3 -> 2 -> 1 -> 3 -> 4 -> 5 -> 2
1 -> 5 -> 3 -> 2 -> 5 -> 4 -> 3 -> 1 -> 2
1 -> 5 -> 3 -> 4 -> 5 -> 2 -> 1 -> 3 -> 2
1 -> 5 -> 3 -> 4 -> 5 -> 2 -> 3 -> 1 -> 2
1 -> 5 -> 4 -> 3 -> 1 -> 2 -> 3 -> 5 -> 2
1 -> 5 -> 4 -> 3 -> 1 -> 2 -> 5 -> 3 -> 2
1 -> 5 -> 4 -> 3 -> 2 -> 1 -> 3 -> 5 -> 2
1 -> 5 -> 4 -> 3 -> 2 -> 5 -> 3 -> 1 -> 2
1 -> 5 -> 4 -> 3 -> 5 -> 2 -> 1 -> 3 -> 2
1 -> 5 -> 4 -> 3 -> 5 -> 2 -> 3 -> 1 -> 2

一笔画总数为：44
```

  - 一笔画总数为44

2.6 欧拉图

- 一笔画问题的本质是能否形成欧拉回路
- 欧拉路径
  - 定义：如果图G中的一个路径包括每个边恰好一次，则该路径称为欧拉路径(Euler path)。
    如果一个回路是欧拉路径，则称为欧拉回路(Euler circuit)。
    具有欧拉回路的图称为欧拉图（简称E图）。具有欧拉路径但不具有欧拉回路的图称为半欧拉图
- 欧拉回路
  - 欧拉回路是指起点和终点相同的欧拉路
- 无向图
  - 存在欧拉路径的充分必要条件：度数为奇数的点只能是0个或者2个
  - 存在欧拉回路的充分必要条件：度数为奇数的只能是0个
- 有向图
  - 存在欧拉路径的充分必要条件：要么所有点的出度均等于入度， 要么除了两个点之外，其余所有点的出度等于入度，剩余的两个点：一个满足出度比入度多1（起点），另一个满足入度比出度多1（终点）
  - 存在欧拉回路的充分必要条件：所有点的出度均等于入度

2.6.1 `Hierholzier` 算法

- 对于一个欧拉回路，我们分析其组成部分，一个欧拉回路必是由多个环连接在一起，才能符合所有的点入度与出度相等。因此`Hierholzer`算法算法的核心就是依次寻找环，将所有的环拼合成一条欧拉回路

- 建两个栈，一个是节点栈，另一个路径栈
- 任取一点，加入节点栈
- 访问当前节点可访问出边，将另一端点入栈
- 当节点栈顶部元素已经没有可以访问的出边了，就把这个节点从节点栈出栈，压入路径栈中
- 直到所有边均被访问结束

2.6.2 `Fluery` 算法

- 设G为一无向欧拉图，求G中一条欧拉回路
  - 任取G中一顶点v_0, 令$P_0 = v_0$
  - 假设沿$P_i = v_0e_1v_1e_2v_2\dots e_iv_i$走到顶点$v_i$，按下面方法从$E(G)-\{e_1,e_2,\dots,e_i\}$中选$e_{i+1}$
    - e_{i+1}与v_i向关联
    - 除非没有其他边可以选择，否则e_{i+1}不应该是$G_i = G-\{e_1,e_2,\dots,e_i\}$中的**桥**
  - 当上一项的第二步不能再进行时算法停止。可以证明，当算法停止时，所得到的简单回路$P_m = v_0e_1v_1e_2v_2\dots e_mv_m \quad (v_m = v_0)$为其中一条欧拉回路

- 就是将已经遍历的边"删除"， 选择下一条边时，下一条边不能是**桥**, 否则会进入到另一个子图中无法再通过这条**桥**回到上一个子图中。但当没有其他边可以选择时，说明该子图已经遍历完，可通过桥进入到下一个子图了

  > 在图论中，一条边被称为“桥”代表这条边一旦被删除，这张图的连通块数量会增加。等价地说，一条边是一座桥当且仅当这条边不在任何环上。一张图可以有零或多座桥

3. 实验总结

- 本次实验进行了欧拉图的遍历，采用`dfs`的方式进行遍历，朴素方法进行寻找
- 讨论了两种欧拉回路的求解方法