

作业 PA5 实验报告

姓名：何正潇 学号：1950095 日期：2021 年 12 月 13 日

1. 涉及数据结构和相关背景

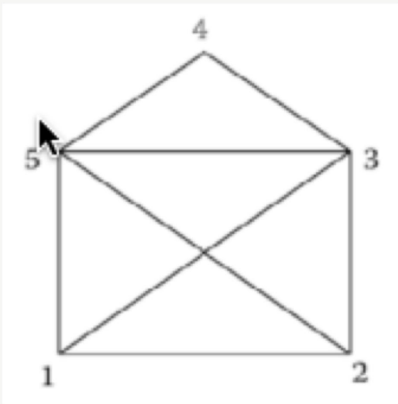
本题涉及欧拉通路的编程，即图的具体数学应用。这题在离散数学中也是一个比较重要的问题。

2. 实验内容

2.1.1 问题描述

实验内容：

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。



参考信息：

2.1.2 基本要求

基本要求就是将所有符合要求的一笔画路线都输出。

2.1.3 数据结构设计

一个邻接矩阵

2.1.4 功能说明（函数、类）

```
int times = 0;
void dfs(int i, bool mat[][7])
{
    for (int p = 1; p <= 5; p++)
    {
        if (mat[i][p] == true)
        {
            mat[i][p] = mat[p][i] = false;
```

```

        a.push_back(pair<int, int>(i, p));
        dfs(p, mat);
    }

}

if (a.size() == 8)
{
    for (int nIndex = 0; nIndex < (int)a.size(); nIndex++)
    {
        cout << a[nIndex].first << "->" << a[nIndex].second<<" ";
    }
    cout << endl;
    times++;
}
if (!a.empty())
{
    pair<int, int> temp;
    temp = a[a.size() - 1];
    mat[temp.first][temp.second] = mat[temp.second][temp.first] = 1;
    a.pop_back();
}

}

```

基本来说就是一个比较简单的 dfs 程序设计，用一个 vector 容器记录已经走过的路径。

2.1.5 调试分析（遇到的问题和解决方法）

```

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

```

总共有几条路子44

时间复杂度为 $O(n)n$ 为边的数量，即对每条边进行检索是否有通路，如果没有的话退回，并且恢复原邻接矩阵。当然这张图本身的数据量很小，因此算法的时间复杂度也影响不大。

2.1.6 总结和体会

本题总体来说是一个解决实际趣味问题的算法题，我们所需要利用所学的 dfs 方式进行图的遍历来找到所有符合要求的路径。是对本章所学内容的巩固和拓展。当然这部分知识也是离散数学中非常重要的知识，因此我们对这种类型的题目不光该能编出程序，同时也应该有数学方面的领悟。

程序源代码

```

#define _CRT_SECURE_NO_WARNINGS
#include<string.h>
#include<map>
#include<string>
#include<stack>
#include<iostream>
#include<iomanip>
#include<queue>
#include <climits>
#include <algorithm>
#include <iomanip>
#include <vector>
#include <set>
#include <queue>
#include <cmath>
using namespace std;
vector<pair<int, int>>a;
int times = 0;
void dfs(int i, bool mat[][7])
{
    for (int p = 1; p <= 5; p++)
    {
        if (mat[i][p] == true)
        {
            mat[i][p] = mat[p][i] = false;
            a.push_back(pair<int, int>(i, p));
            dfs(p, mat);
        }
    }

    if (a.size() == 8)
    {
        for (int nIndex = 0; nIndex < (int)a.size(); nIndex++)
        {
            cout << a[nIndex].first << "->" << a[nIndex].second<<" ";
        }
        cout << endl;
        times++;
    }
    if (!a.empty())
    {
        pair<int, int> temp;
        temp = a[a.size() - 1];
        mat[temp.first][temp.second] = mat[temp.second][temp.first] = 1;
    }
}

```

```
        a.pop_back();
    }

}

int main()
{
    bool mat[7][7];
    bool visited[6] = { false };
    memset(mat, 1, 7 * 7 );
    mat[1][1] = mat[2][2] = mat[3][3] = mat[4][4] = mat[5][5] = mat[1][4] = mat[4][1] =
mat[2][4] = (mat[4][2] = 0);
    dfs(1, mat);
    cout << "总共有几条路子" << times << endl;
    return 0;
}
```