欧拉路径试验报告

姓名: 范潇　学号: 2254298

2023年 12月 1日

1　涉及数据结构和相关背景

2　实验目的

1. 掌握图的存储结构和基本操作；
2. 灵活运用图的遍历方法。

3　实验内容

编写一个程序，其功能是输出图1中以结点1开始的欧拉路径。

4　程序实现

4.1　数据结构设计

```
int Map[N][N]={//下标为的元素为冗余，以确保有效数据的下标从开始01
   {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}
};
```

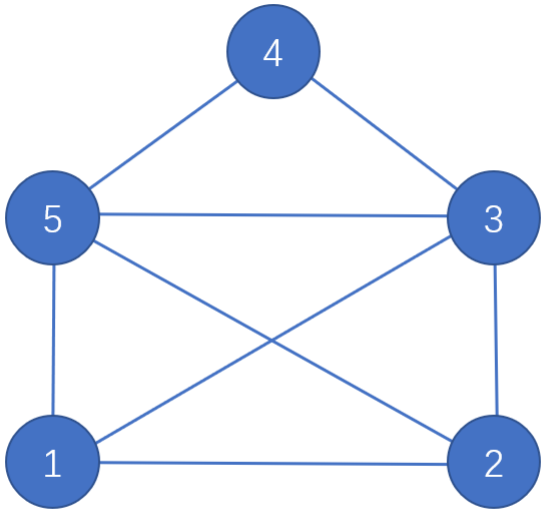

图 1: map对应的无向图

4.2 功能说明

```
void DFS(string path,int now)//分别存储当前的路径及其末端结点
{
    if(path.length()==9){//给定图的欧拉路径长度为9
        cout<<path<<endl;
        return;
    }
    for(int i =1;i<N;i++ )
      if(Map[now][i]){//对应边存在，且未出现在当前路径中
          path.push_back(i+'0');//添加进当前路径中
          Map[now][i]=0;//将对应边置零，表明已经出现在当前路径中
          Map[i][now]=0;
          DFS(path,i);//进一步递归
          path.pop_back();//恢复当前函数调用时的状态
          Map[now][i]=1;
          Map[i][now]=1;
      }
    return ;
}
```

```
19 int main()
20 {
21     DFS("1",1);//从指定结点开始
22     return 0;
23 }
```

4.3 调试分析

第一次调试时发现输出的序列明显不满足欧拉路径，根据调试模式下的自动变量窗口信息进行排查，发现是因为在第12行结束后只恢复了邻接矩阵，但是并没有恢复 path。

5 总结与体会

本题由于要求输出所有的欧拉路径，所以采用循环和递归相结合的形式。在编写程序时，要注意每次循环体开始前，都要保持邻接矩阵和用于存储欧拉路径的变量值不变。假设所给的无向图有 e条边，则其对应的欧拉路径也有 e条边，函数需要递归$e+1$层。而在第 i层中，至多再次调用$n-i$次自身。因此时间复杂度为 $O(n*e)$，如果使用邻接表实现，则时间复杂度可以降至$O(e^2)$。空间复杂度则是 $O(n^2)$，主要是由邻接矩阵贡献的。