作业 PA5 实验报告

姓名：钱子贤 学号：2054170 日期：2021 年 12 月 19 日

实验报告格式按照模板来即可，对字体大小、缩进、颜色等不做要求

实验报告要求在文字简洁的同时将内容表示清楚

1. 涉及数据结构和相关背景

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。
特性
存在于一个连通的图（块）。图中无孤立的点
欧拉回路
对于无向图来说，度数（点上所连边的数量）为奇数的点为 0。
对于有向图来说，每个点的入度等于出度
欧拉路径
对于无向图来说，度数为奇数的点为 0 或者 2。当数量为 2 时，这两个点必为一个路径起点一个路径终点。
对于有向图来说，可容纳两个入度不等于出度的点，其中一个入度必出度大 1，为路径起点。另一个入度比出度小 1，为路径终点。

2. 实验内容

一笔画
实验目的：
1、掌握图的结构和基本操作；
2、灵活运用图的遍历方法。
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：
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
深度优先搜索
首先以一个未被访问过的顶点作为起始顶点，沿当前顶点的边走到未访问过的顶点； 当没有未访问过的顶点时，则回到上一个顶点，继续试探访问别的顶点， 直到所有的顶点都被访问。
实验要求 ：

(1) 程序要添加适当的注释，程序的书写要采用 缩进格式 。
(2) 程序要具在一定的 健壮性，即当输入数据非法时， 程序也能适当地做出反应，如 插入删除时指定的位置不对 等等。
(3) 程序要做到界面友好，在程序运行时用户可以根据相应的提示信息进行操作。
(4) 根据实验报告模板详细书写实验报告，在实验报告中给出主要算法的复杂度分析。

由于有向图有欧拉回路当且仅当图是弱连通的并且所有点的 入度-出度=0，可以发现关键问题在于如何给无向边定向使得入度-出度=0.
先给所有无向边任意指定一个方向，并令 d'[i]=i 的出度-入度（算上原图的有向边和定向了的无向边）。 如果某个 d'[i]是奇数（相当于说它连接了奇数条边，包括有向边和无向边），那么显然无解，否则令 d[i]=d'[i]/2（为了方便，暂时称其为“度”）。
考虑如果有一条边 u->v 在原图中是无向边，那么将它反向，变成 v->u 之后，u 的出度-1，入度+1，从而 d[u]-=1；同理，d[v]+=1. 可以发现这相当于说把 u 的一个度“流到”了 v 的一个度。

```
int a[10], c[10], du[10], n, x, y, k, t, tot = 0;
bool b[10][10];
char s[2];
void dfs(int u) {
    for (int i = 0; i < n i++)
        if (b[u][i]) {
            b[u][i] = b[i][u] = 0;//删边
            dfs(i);
        }
    c[++tot] = u;//递归退栈时存储，所以顺序是反的，也可以用栈
    for (int i = 1; i <= n; i++) {
        x = s[0] - 'A'; y = s[1] - 'A';//节省空间
        k = min(k, min(x, y));
        b[x][y] = b[y][x] = 1;//无向图标记路径
        du[x]++; du[y]++;//计算度
    }
    for (int i = 0; i < n; i++)
        if (du[i] & 1) a[++a[0]] = i;//计算度是奇数的点，并保存
    if (a[0] == 0) dfs(k);//题目要求输出字典序最小的方案
//没有度为奇数的点 ，这是欧拉环路的情况
    else if (a[0] == 2) dfs(a[1]);//第一个度为奇数的点是端点
//度为奇数的点为两个，这两个是两端的端点，这是欧拉路径的情况
```

算法（输出字典序最小的答案）
此法输出的是所有合法方案中字典序最小的答案
本算法求出的是倒序的欧拉回路，所以输出的时候要倒序输出才是字典序最小的答案。

（在网络流中）从 u 到 v 连一条容量为 1 的边，表示可以在 d[u] 中流出一个给 d[v]。另外，对每个点 u，如果 d[u]>0，那么要从 S 到 u 连一条容量为 d[u] 的边，表示 u 这里多出了 d[u] 的度； 否则 d[u]<=0，就从 u 到 T 连一条容量为 -d[u]的边，表示 u 这里缺少 -d[u] 的度。 执行一遍最大流之后如果 S 出发的所有弧都满流，就说明所有多出来的度都流出去了（同时所有少的度都补上了，因为少的度的总数 这时候只需要看网络流中哪些 u->v 的边流上了，就知道哪些边需要反向。把这些边反向之后找一遍有向图欧拉回路即可。

```
typedef pair<int, int> PII;
int n, m;
int d[N];
int g[N][N];
int ans[M], cnt;

void dfs(int x) {
    for (int i = 1; i <= n; ++i) {
        if (g[x][i]) {
            g[x][i] --, g[i][x] --;
            dfs(i);
        }
    }
    ans[++cnt] = x;
}



    int start = 1;
    while (!d[start])start++;

    for (int i = 1; i <= n; ++i) {
        if (d[i] % 2) {
            start = i;
            break;
        }
    }
    dfs(start);
```

输出结果：

```
5 8
0 1 1 0 1
1 0 1 0 1
1 1 0 1 1
0 0 1 0 1
1 1 1 1 0
最终一笔画的总数为:
44
```

3. 实验总结

一笔画问题

如果一个图存在一笔画，则一笔画的路径叫做欧拉路，如果最后又回到起点，那这个路径叫做欧拉回路。

我们定义奇点是指跟这个点相连的边数目有奇数个的点。对于能够一笔画的图，我们有以下两个定理。

定理 1：存在欧拉路的条件：图是连通的，有且只有 2 个奇点。

定理 2：存在欧拉回路的条件：图是连通的，有 0 个奇点。

两个定理的正确性是显而易见的，既然每条边都要经过一次，那么对于欧拉路，除了起点和终点外，每个点如果进入了一次，显然一定要出去一次，显然是偶点。对于欧拉回路，每个点进入和出去次数一定都是相等的，显然没有奇点。

一笔画性质:

1.凡是由偶点组成的连通图，一定可以一笔画成。画时可以把任一偶点为起点，最后一定能以这个点为终点画完此图。

2.凡是只有两个奇点的连通图（其余都为偶点），一定可以一笔画成。画时必须把一个奇点为起点，另一个奇点为终点。

3.其他情况的图都不能一笔画出。(奇点数除以二便可算出此图需几笔画成。)

求欧拉路的算法很简单，使用深度优先遍历即可。

根据一笔画的两个定理，如果寻找欧拉回路，对任意一个点执行深度优先遍历；找欧拉路，则对一个奇点执行 DFS，时间复杂度为 O(m+n)，m 为边数，n 是点数。