浅谈欧拉回路及其应用

Author:huan_yp

引入：

一位毒瘤群友经常在群里发一笔画红包，导致幻影彭相当火大，所以要尝试找出通解。

抽象：

将一笔画的过程抽象，即对于一张简单无向连通图 V, E ，需要找出一条路径，满足经过每条边各一次。

对于实际应用的限制，可以假设 $n \leq 25, m \leq 200$

结论及简要证明：

定义与一个点 u 相连的边数为其度数 d_u，显然满足以下两种情况：

- $C1$:所有点度数均为偶数。
- $C2$:存在且仅存在两个点度数为奇数，且这两个点必为起点和终点。

证明如下:

在图中任取一点，以该点作为起点，沿着欧拉回路走，当前顶点的出度为 1，然后经过其它的顶点，注意到如果欧拉路径经过一个顶点（包括起点），它必然离开这个点，这样出入度之和为偶数，直到所有的边逐一被走过，回路的终点在起点处结束，使得起点的入度加 1，这样经过起点的度数和变成偶数，欧拉回路结束。

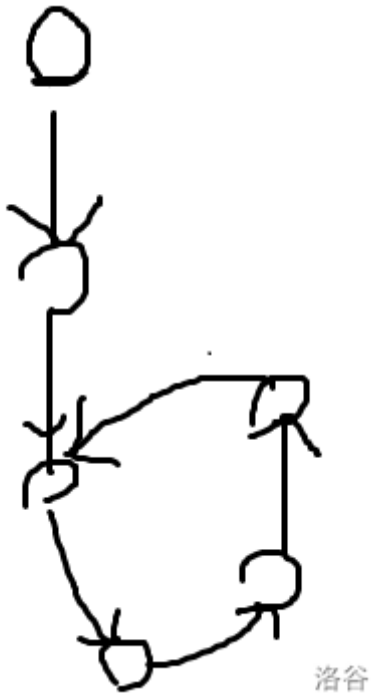
同理可证明 $C2$

任意图构造方案：

事实上，$C1, C2$ 的必要性，也是通过方案构造证明的。

对于 $C1$，我们从任意点出发，对图进行暴力遍历，实际上，无边可走的情况一定只存在于走回了起点，这个时候回溯即可，回溯到上一个岔路 x 并继续遍历，此时如果考虑已经走过的路径，不难发现新图是原图的一个子图，且删去的是一个环，并且仍然满足 $C1$ 性质，于是将 x 作为子图的起点，递归进行如上算法，求出子图上的欧拉回路后，合并环上，得到原图的欧拉回路。

对于 $C2$，从两个度数为奇数的点出发，任意走，如果走到无边的情况，我们断言图是一个 6 字图，即：



同 $C1$，进行回溯操作，即可找到唯一路径。

算法实现：

暴力 `DFS`，每次回溯时记录边，将所有边反向得到原图欧拉路径。

C++ 实现：

```
//Author:huan_yp
#include<bits/stdc++.h>
const int N=1e6+10;
struct edge{
    int v,next;
}a[N];
int top,cnt;
int st[N],head[N],d[N];
void addedge(int u,int v)
{
    a[cnt].next=head[u],head[u]=cnt++,a[cnt-1].v=v;
    a[cnt].next=head[v],head[v]=cnt++,a[cnt-1].v=u;
}
void FindPath(int u)
{
    for(int i=head[u];i!=-1;i=a[i].next)
    {
        if(!a[i].v)continue;
        int v=a[i].v;
        a[i].v=a[i^1].v=0;
        //标记已经使用过的边
        //由于存储同一条边的编号连续，所以可以直接异或
        FindPath(v);
        st[++top]=v;
    }
}
void PrintPath()
{
```

```
    int now=1;
    top--;
    while(top)
    {
        std::cout<<"Path"<<now++<<":"<<st[top+1]<<' '<<st[top]<<'\n';
        top--;
    }
}
int main()
{
    //以求解欧拉回路为例
    //欧拉路类似
    int n,m;
    std::cin>>n>>m;
    std::memset(head,-1,sizeof(head));
    for(int i=1;i<=m;i++)
    {
        int x,y;
        std::cin>>x>>y;
        d[x]++,d[y]++;
        addedge(x,y);
    }
    FindPath(1);
    st[++top]=1;
    PrintPath();
    return 0;
}
```

Python 实现：

```
#Author:huan_yp
class Edge():
    def __init__(self,v,rk,next):
        self.v,self.rk,self.next = v,rk,next

st,head,a,cnt= [],[],[],0
def PrintPath():
    now = 0
    while(len(st) >= 2):
        print("Path:"+str(now)+":"+str(st[-1])+"->"+str(st[-2]))
        now += 1;st.pop()
def Addedge(u,v):
    global cnt,a
    a[cnt] = Edge(v,cnt,head[u]);head[u] = cnt;cnt += 1
    a[cnt] = Edge(u,cnt,head[v]);head[v] = cnt;cnt += 1
def FindPath(u):
    global head,a,st
    i = head[u]
    while(i != -1):
        v = a[i].v
        if(a[i].v == None):
            i = a[i].next
            continue;
        a[i].v = a[i^1].v = None
        FindPath(v)
        i = a[i].next
        st.append(v)
```

```
n,m=list(map(int,input().split()))
head = [-1] * (n + 1)
a = [None] * (2 * (m + 1))
for i in range(m):
    x,y = list(map(int,input().split()))
    Addedge(x,y)
FindPath(1)
st.append(1)
PrintPath()
```

使用注意：

第一行两个正整数 n, m，表示图的点数和边数，接下来 m 行每行两个正整数 x, y ，表示存在一条 x, y 的无向边，可以重边和自环。

时间复杂度分析：

以上两份程序均为 $O(n * m)$，可以进一步优化到 $m * log_n$，使用数据结构维护边即可。

实际应用探讨：

解 QQ 一笔画红包的主要问题在于数边，由于图像构造相对复杂，所以边的统计比较困难，提供一种解决思路。

观察到走出第一步后，只需要点击点即可连边，将所有点一次标号为 $1->25$，尝试以每个点为起点，向所有点连边，即可判断是否存在这条边。

实际交互的一些问题：

可以使用 `Airtest` 框架交互，具体内容可以看这个

例题：

例题1

例题2

例题3：

请编写一个应用软件，支持自动领取 qq 一笔画红包。