Cover

考查内容:

欧拉回路

题目分析:

题目要求找到尽可能少的k条路径，覆盖整个无向图的每一条边，并输出任意一种方案。

这个无向图可能包含多个连通块，对于每一个连通块，若其中每个点的度都是偶数，那么该连通块可以构成欧拉回路，则一条路径即可覆盖整个连通块。若一个连通块有两个奇度节点，则我们为这两个点连接一条边之后，仍可以构成欧拉回路，去掉这条虚拟的边之后，欧拉回路的环形路径变成了一条链，仍可以用一条路径覆盖。若该连通块有2*n条奇度点，那么就相当于从环上拆掉不相邻的n条边，需要n条路径可覆盖该连通块。

我们为整个图上的奇度点两两连接一条边，并为这个边染色。求出加边之后每个连通块的欧拉回路（一个环），从任意一个节点开始遍历这个环，看到染色的边，就将前后断开，随后会形成n+1条链，将第一条链与最后一条链首尾相连，即可求得该连通块的一种最小路径覆盖方案。

我在写这道题的时候，由于没有注意孤立的点不需要额外路径来覆盖，而多输出了很多回车，导致格式错误qaq。

代码实现:

```
#define _CRT_SECURE_NO_DEPRECATE
/************************
*创建时间：2018 07 30
*文件类型：源代码文件
*题目来源：HDU
*当前状态：已通过
*备忘录：欧拉回路
*作者：HtBest
************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
using namespace std;
namespace IO
{
    #define SIZE 10000
    int T=0,END=0;
    char s[SIZE+1];
    int in()
    {
        T=0;
```

```
        return END=fread(s,1,SIZE,stdin);
    }
    int read(int &ans)
    {
        int i=ans=0,flag=1;
        while(1)
        {
            if(T==END&&!in())return i;
            if(s[T]==' '||s[T]=='\n')
                {if(i)return i;}
            else if(s[T]=='-')flag=-1;
            else
            {
                ans=ans*flag*10+s[T]-'0';
                ++i;
            }
            ++T;
        }
    }
}
using namespace IO;
#define MAXN 100001
int n,m,_edge,maxedge,head[MAXN],vis[3*MAXN],nodevis[MAXN],du[MAXN],num[MAXN];
int op[3*MAXN],_op,ls[MAXN],_ls;
deque <int> sta;
struct EDGE
{
    int a,b,next;
    EDGE(int a=0,int b=0,int next=0):a(a),b(b),next(next){}
}edge[3*MAXN];
/* Variable explain:
n:节点个数
m:边个数
edge:邻接表
_edge:邻接表元素个数
maxedge:真实边个数
head:每个节点的第一条出边
vis:每条边是否被访问
nodevis:每个点是否被访问
du:每个点的度
num:每个连通分量的大小
op:输出缓冲区
_op:输出缓冲区元素个数
ls:凑环缓冲区
_ls:凑环缓冲区元素个数
sta:存欧拉道路的栈
*/
void adde(int a,int b)
{
    edge[++_edge]=EDGE(a,b,head[a]);
    head[a]=_edge;
}
void read()
{
    int ls1,ls2;
    if(read(n)==0)exit(0);
    read(m);
    sta.clear();
    for(int i=1;i<=n;++i)head[i]=nodevis[i]=du[i]=0;
    for(int i=1;i<=_edge;++i)vis[i]=0;
    _edge=1;_op=_ls=0;
```

```
    for(int i=1;i<=m;++i)
        read(ls1),read(ls2),adde(ls1,ls2),adde(ls2,ls1),++du[ls1],++du[ls2];
    maxedge=_edge;
    ls1=0;
    for(int i=1;i<=n;++i)
        if(du[i]%2)
        {
            if(ls1)adde(ls1,i),adde(i,ls1),ls1=0;
            else ls1=i;
        }
    return;
}
int dfs(int a,int v)
{
    int num=0;
    nodevis[a]=v;
    for(int i=head[a];i;i=edge[i].next)
    {
        int b=edge[i].b;
        if(!vis[i])
        {
            vis[i]=vis[i^1]=v;
            num+=dfs(b,v);
            ++num;
            sta.push_back(i%2==0?i/2:-(i/2));
        }
    }
    return num;
}
int main()
{
    // freopen("in","r",stdin);
    // freopen("out","w",stdout);
    while(1)
    {
        read();
        int T=0,ans=0;
        for(int i=1;i<=n;++i)   if(!nodevis[i])++T,num[T]=dfs(i,T);
        for(int i=T;i;--i)
        {
            if(!num[i])continue;
            _ls=0;
            int head=++_op,cnt=0,seg=0;
            /*
            head:每个道路的第一个位置，放个数用的
            cnt:记录道路长度
            seg:记录是否为第一段，接到最后一段后面
            */
            for(int j=1;j<=num[i];++j)
            {
                int ls1=sta.back();
                sta.pop_back();
                if(abs(ls1)<<1>maxedge)
                {
                    if(seg==0)
                    {
                        for(int k=head+1;k<=_op;++k)
                            ls[++_ls]=op[k];
                        _op=head;
                        seg++;
                        cnt=0;
                        continue;
                    }
                    op[head]=cnt;
                    op[++_op]=0;//回车
```

```
                    ++ans;
                    cnt=0;
                    head=++_op;
                }
                else    op[++_op]=ls1,++cnt;
            }
            if(cnt||_ls)
            {
                for(int j=1;j<=_ls;++j)
                    op[++_op]=ls[j],++cnt;
                op[head]=cnt;
                op[++_op]=0;//回车
                ++ans;
            }
        }
        printf("%d\n",ans);
        for(int i=1;i<=_op;++i)
            op[i]?printf("%d ",op[i]):puts("");
    }

    return 0;
}
```

<题目跳转> <查看代码>