# 基础算法B班 Contest7题解

周子嘉&贺一鸣&
骆天祺

# 目　录

# #A.图上数码游戏

## 题目大意

### 题目描述

$Mas$ 在研究八数码问题

$3 \times 3$ 的方格内完成把八数码,其规则为每次将空白数码与其上下左右相邻的方格上的数字进行交换,最终达到目标局面

这个问题对你来说太简单了,现在 $Mas$ 决定在一个无向图中完成

已知无向图有 $9$ 个节点 $m$ 条边,每个节点上有一个数字第 $i$ 个节点上的数为 $p_i$ ,其中 $0$ 表示空白数码

现在规定操作规则如下

若 $0$ 在顶点 $u$ 上,那么空白数码可以与 $v$ 上的数码进行交换,其中 $v$ 与 $u$ 有一条边连接

给出初始局面,请你计算最少需要多少步才能将这个数码问题还原成初始局面

> *初始局面为* $p_j = j(j \in [1,8])$ *同时* $p_9 = 0$

## 输入输出格式

### 输入格式

第一行输入一个正整数 $m$

接下来 $m$ 行每行两个整数 $u, v$ 表示 $u$ 到 $v$ 有一条边连接

最后一行输入 $8$ 个正整数,第 $i$ 个整数 $x$ 表示 $p_x = i$

### 输出格式

输出一个整数,表示最小步数,如果无法还原输出 `-1`

# #A.图上数码游戏

## 样例

**输入样例1**

```
5
1 2
1 3
1 9
2 9
3 9
3 9 2 4 5 6 7 8
```

**输出样例1**

```
5
```

**样例解释1**

初始局面为 $0,3,1,4,5,6,7,8,2$

第 $1$ 步,将 $p_1,p_9$ 交换,局面变为 $2,3,1,4,5,6,7,8,0$

第 $2$ 步,将 $p_2,p_9$ 交换,局面变为 $2,0,1,4,5,6,7,8,3$

第 $3$ 步,将 $p_1,p_2$ 交换,局面变为 $0,2,1,4,5,6,7,8,3$

第 $4$ 步,将 $p_1,p_3$ 交换,局面变为 $1,2,0,4,5,6,7,8,3$

第 $5$ 步,将 $p_3,p_9$ 交换,局面变为 $1,2,3,4,5,6,7,8,0$

**输入样例2**

```
12
6 5
5 4
4 1
4 7
8 5
2 1
2 5
6 9
3 6
9 8
8 7
3 2
2 3 4 6 1 9 7 8
```

**输出样例2**

```
16
```

## 限制

| | |
|---|---|
| 时间限制 | 2000ms |
| 内存限制 | 128MB |

# #A.图上数码游戏

这道题对于A了#569.八数码问题的同学来说还是很友好的（毕竟只要在原来代码上改一点点就A了），没A的同学可以看看下方八数码问题的代码，理解一下思路。

```cpp
#include<bits/stdc++.h>
using namespace std;
short dir[4][2]{{0,1},{0,-1},{1,0},{-1,0}};
string s,m="123804765";
unordered_map<string,bool> vis;
struct node{
    string c;
    long long t;
};
long long bfs(string s) {
    queue<node> q;
    q.push({s,0});
    vis[s]=true;
    while (q.size()) {
        auto cur=q.front();
        q.pop();
        if (cur.c==m) {
            return cur.t;
        }
        long long pos=cur.c.find('0');
        long long x=pos/3,y=pos%3;
        for (int i=0;i<4;i++) {
            long long tx=x+dir[i][0],ty=y+dir[i][1];
            if (tx<0||tx>2||ty<0||ty>2) {
                continue;
            }
            long long poss=tx*3+ty;
            swap(cur.c[pos],cur.c[poss]);
            if (!vis[cur.c]) {
                vis[cur.c]=true;
                q.push({cur.c,cur.t+1});
            }
            swap(cur.c[pos],cur.c[poss]);
        }
    }
    return -1;
}
int main()
{
    cin>>s;
    printf("%lld",bfs(s));
    return 0;
}
```

# #A.图上数码游戏

这道题与八数码问题的最大的不同之处就在于我们在八数码问题中，是把0向上下左右四个方向，靠swap移动；这道题中，我们只需改成与邻居swap。~~这道题还是非常水的~~出题人还是很良心的。

温馨提示：char数组与string的下标都是从0开始，而这一题中点的编号却是1~9，在下标处理方面得注意一下。

# #A.图上数码游戏

核心代码

```
#include<bits/stdc++.h>
using namespace std;
long long n,u,v,num;
string s="         ",m="123456780";
unordered_map<string,bool> vis;
vector<long long> g[10];
struct node{
    string c;
    long long t;
};
long long bfs(string s) {
    queue<node> q;
    q.push({s,0});
    vis[s]=true;
    while (q.size()) {
        auto cur=q.front();
        q.pop();
        if (cur.c==m) {
            return cur.t;
        }
        long long pos=cur.c.find('0');
        for (auto &&v:g[pos]) {
            swap(cur.c[pos],cur.c[v]);
            if (!vis[cur.c]) {
                vis[cur.c]=true;
                q.push({cur.c,cur.t+1});
            }
            swap(cur.c[pos],cur.c[v]);
        }
    }
    return -1;
}
int main()
{
    scanf("%lld",&n);
    for (long long i=1;i<=n;i++) {
        scanf("%lld%lld",&u,&v);
        u--;
        v--;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    for (long long i=1;i<=8;i++) {
        scanf("%lld",&num);
        num--;
        s[num]=i+'0';
    }
    for (long long i=0;i<9;i++) {
        if (s[i]==' ') {
            s[i]='0';
            break;
        }
    }
    printf("%lld",bfs(s));
    return 0;
}
```

# #B.连接牛棚

## 题目大意

## 样例及数据规模

### #B、连接牛棚

#### 题目描述

$FJ$ 的农场由 $N$ 块田地组成,编号为 $1 \sim N$.

在这些田地之间有 $M$ 条双向道路,每条道路连接两块田地

农场有两个牛棚,一个在田地 $1$ 中,另一个在田地 $N$ 中

$FJ$ 希望确保有一种方式可以沿着一组道路在两个牛棚之间行走

他愿意建造至多两条新道路来实现这一目标

由于田地的位置因素,在田地 $i$ 和 $j$ 之间建造新道路的花费是 $(i-j)^2$

请帮助 $FJ$ 求出使得牛棚 $1$ 和 $N$ 可以相互到达所需要的最小花费

#### 输入格式

每个测试用例的输入包含 $T$ 个子测试用例
所有子测试用例必须全部回答正确才能通过整个测试用例

输入的第一行包含 $T$,之后是 $T$ 个子测试用例

每个子测试用例的第一行包含两个整数 $N$ 和 $M$

以下 $M$ 行,每行包含两个整数 $i$ 和 $j$,表示一条连接两个不同田地 $i$ 和 $j$ 的道路

输入保证任何两个田地之间至多只有一条道路,并且所有子测试用例的 $N+M$ 之和不超过 $5 \times 10^5$

#### 输出格式

输出 $T$ 行
第 $i$ 行包含一个整数,为第 $i$ 个子测试用例的最小花费

#### 输出格式

输出 $T$ 行
第 $i$ 行包含一个整数,为第 $i$ 个子测试用例的最小花费

#### 输入样例

```
2
5 2
1 2
4 5
5 3
1 2
2 3
4 5
```

#### 输出样例

```
2
1
```

#### 样例解释

第一个子测试用例中,最优的方式是用一条道路连接田地 $2$ 和 $3$,用一条道路连接田地 $3$ 和 $4$

第二个子测试用例中,最优的方式是用一条道路连接田地 $3$ 和 $4$
不需要第二条道路

#### 数据规模

对于 $20\%$ 的数据 $N \leq 20$

对于 $50\%$ 的数据 $N \leq 10^3$

对于 $100\%$ 的数据 $\sum_{i=1}^{T}(N+M) \leq 5 \times 10^5$

# #B.连接牛棚

## 思路

这道题我们要预处理出1和n所在的联通分支。
然后由于最多只能添加两条道路，因此可以分成几种情况：

- 1与n连通，此时花费为0。
- 在1与n之间枚举一个i，分别在1和n所在的联通分支二分查找与i差最小的元素，计算花费并取min。
- 可以用并查集，这样就能轻松维护连通分支，虽然思路相同，但代码会更简单。

# #B.连接牛棚

## 核心代码

```
for(int i = 1;i <= n;i++)
{
    ll r1 = 1e18,r2 = 1e18,q = lower_bound(a.begin(),a.end(),i) - a.begin();
    if(q < a.size())
        r1 = min(r1,dis(a[q],i));
    if(q > 0)
        r1 = min(r1,dis(a[q - 1],i));
    q = lower_bound(b.begin(),b.end(),i) - b.begin();
    if(q < b.size())
        r2 = min(r2,dis(b[q],i));
    if(q > 0)
        r2 = min(r2,dis(b[q - 1],i));
    u = find(i);
    f[u] = min(f[u],r1);
    g[u] = min(g[u],r2);
}
```

# #C.二分图判定

## 题目大意

### 题目描述

给出一张 $n$ 个点 $m$ 条边的无向图,请你判定其是否是一张二分图

也即,是否存在一种对图中每个节点进行黑白染色的方案,使得任意两个相邻的节点拥有不同的颜色

# #C.二分图判定

## 数据范围

**数据范围**

对于 $30\%$ 的数据, $n \leq 20, m \leq 30$

对于 $50\%$ 的数据, $n \leq 2000, m \leq 3000$

对于 $100\%$ 的数据, $T \leq 15, n \leq 2 \times 10^5, m \leq 3 \times 10^5$

## 限制

时间限制 1800ms

内存限制 256MB

温馨提示：1.应该不会有人开2e5*2e5的邻接矩阵吧（即使是bitset）2.多测要清空

# #C.二分图判定

方法一

DFS直接染色

核心代码

```
void dfs(long long u) {
    vis[u]=true;
    for (long long i=0;i<g[u].size();i++) {
        if (!vis[g[u][i]]) {
            col[g[u][i]]=1-col[u];
            dfs(g[u][i]);
        }
        else {
            if (col[u]==col[g[u][i]]) {
                ans=false;
                return;
            }
        }
    }
}
```

# #C.二分图判定

## 方法二

我们可以用一种简洁而优雅的数据结构——并查集，开2*n的大小，前一半表示V1黑，V2黑……Vn黑,后一半表示V1白，V2白……Vn白。

对于边(u，v)，可能是u黑，v白，知道u黑可以推出v白，知道v白可以推出u黑，即知道其中一个可以推出另一个。因此，我们将u黑所在的集合与v白所在的集合合并。对于u白，v黑，同理，我们将u白所在的集合与v黑所在的集合合并。

# #C.二分图判定

## 方法二

这样，任意一个集合中任意两个元素都满足知道一个可以推出另一个的关系。

如果我们在某一时刻发现Vi黑与Vi白属于同一集合的元素（即Vi黑能推出Vi白，这显然不可能），那么这张图不是二分图；反之，这张图是二分图。

以上两种方法时间复杂度均为O(T(n+m))(一般方法二快大约300ms)，方法一空间复杂度O(n+m),方法二空间复杂度O(n)。

# #C.二分图判定

## 方法二核心代码

```
#include<bits/stdc++.h>
using namespace std;
long long t,n,m,u,v,f[400005];
bool ans;
void inik(long long n) {
    for (long long i=1;i<=n;i++) f[i]=i;
}
long long find(long long x) {
    if (f[x]==x) return x;
    return f[x]=find(f[x]);
}
void merge(long long a,long long b) {
    long long fa=find(a),fb=find(b);
    f[fa]=fb;
}
bool check(long long x,long long y) {
    return find(x)==find(y);
}
```

```
int main() {
    scanf("%lld",&t);
    for (long long i=1;i<=t;i++) {
        scanf("%lld%lld",&n,&m);
        inik(2*n);
        ans=true;
        for (long long j=1;j<=m;j++) {
            scanf("%lld%lld",&u,&v);
            merge(u,v+n);
            merge(v,u+n);
            if (check(u,u+n)||check(v,v+n)) {
                ans=false;
            }
        }
        if (ans) printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}
```

# #D.游戏通关

## 题目大意

### 题目描述

$Mas$ 是一位硬核游戏玩家

最近一款新游戏刚刚上市, $Mas$ 自然要快速攻略游戏,守护硬核游戏玩家的一切!

为简化模型,我们不妨假设游戏有 $N$ 个剧情点,通过游戏里不同的操作或选择可以从某个剧情点去往另外一个剧情点

此外,游戏还设置了一些**存档**,在某个剧情点可以将玩家的游戏进度保存在一个档位上,读取存档后可以回到剧情点,重新进行操作或者选择,到达不同的剧情点

为了追踪硬核游戏玩家 $Mas$ 的攻略进度,你打算写一个程序来完成这个工作

假设你已经知道了游戏的全部剧情点和流程,以及 $Mas$ 的游戏操作,请你输出 $Mas$ 的游戏进度

# #D.游戏通关

## 输入输出格式

### 输入格式

输入第一行是两个正整数 $N$ 和 $M$ ,表示总共有 $N$ 个剧情点, $Mas$ 有 $M$ 个游戏操作

接下来的 $N$ 行,每行对应一个剧情点的发展设定
第 $i$ 行的第一个数字是 $K_i$ ,表示剧情点 $i$ 通过一些操作或选择能去往下面 $K_i$ 个剧情点
接下来有 $K_i$ 个数字,第 $k$ 个数字表示做第 $k$ 个操作或选择可以去往的剧情点编号。

最后有 $M$ 行,每行第一个数字是 $0$ 、 $1$ 或 $2$ ,分别表示:

- $0$ 表示 $Mas$ 做出了某个操作或选择,后面紧接着一个数字 $j$ ,表示 $Mas$ 在当前剧情点做出了第 $j$ 个选择。我们保证 $Mas$ 的选择永远是合法的
- $1$ 表示 $Mas$ 进行了一次存档,后面紧接着是一个数字 $j$ ,表示存档放在了第 $j$ 个档位上
- $2$ 表示 $Mas$ 进行了一次读取存档的操作,后面紧接着是一个数字 $j$ ,表示读取了放在第 $j$ 个位置的存档

约定:所有操作或选择以及剧情点编号都从 $1$ 号开始。存档的档位不超过 $100$ 个,编号也从 $1$ 开始。游戏默认从 $1$ 号剧情点开始。总的选项数(即 $\sum K_i$ )不超过 $10^6$

### 输出格式

对于每个 $1$ (即存档) 操作,在一行中输出存档的剧情点编号

最后一行输出 $Mas$ 最后到达的剧情点编号

# #D.游戏通关

## 样例

### 输入样例

```
10 11
3 2 3 4
1 6
3 4 7 5
1 3
1 9
2 3 5
3 1 8 5
1 9
2 8 10
0
1 1
0 3
0 1
1 2
0 2
0 2
2 2
0 3
0 1
1 1
0 2
```

### 输出样例

```
1
3
9
10
```

### 样例解释

简单给出样例中经过的剧情点顺序：

`1 -> 4 -> 3 -> 7 -> 8 -> 3 -> 5 -> 9 -> 10` 。

档位 $1$ 开始存的是 $1$ 号剧情点；档位 $2$ 存的是 $3$ 号剧情点；档位 $1$ 后来又存了 $9$ 号剧情点

## 数据规模及限制

### 数据规模

对于全部的数据 $1 \leq N, M \leq 10^5$

| | |
|---|---|
| 时间限制 | 1000ms |
| 内存限制 | 512MB |

# #D.游戏通关

思路

模拟即可

~~很明显是来送分的~~

~~可对于某些人就是送不出去啊~~

~~还是提醒一下某些拿45分的人吧~~

唯一需要注意的一点就是K数组不要用二维数组存，会爆内存，我用的是vector数组代替二维数组

# The End