题意解释

有一个 n 个点，$n+t$ 条边的连通图，构造出图的一个 $k \leq 3$ 染色方案，或者报告无解。

知识点提炼

二分图，广义串并联图，2-sat，随机化，搜索

核心解题思路

思路一：$k=1$

图中不能有边，可以通过 **subtask 1**。

思路二：$k=2$

bfs/dfs二分图判定，可以通过 **subtask 2**。

思路三：$n \leq 15$

爆搜，复杂度 $O(3^n \mathrm{poly}(n))$，可以通过 **subtask 3**。

思路四：$t=-1$ 以及 $t=0$

$t=-1$时，图是一棵树，树是二分图。

$t=0$ 时，图是一棵基环树，先给环跑三染色，然后剩余部分就是一个森林，bfs/dfs染色即可。

结合以上思路，可以通过 **subtask 1-5**。期望得分 55 分。

思路五：$t \leq 8$

注意到 t 很小，可以作为突破口。

找出原图的一棵生成树，那么额外边只有 $t+1$ 条。我们把额外边邻接的 $2(t+1)$ 个点拉出来，如果删除这些点，剩余的部分是容易2-染色的。也即我们可以把图缩小到只有 $O(2t)$ 个点的规模，之后暴力三染色。确定关键点的颜色之后，再依次确定剩余点的颜色即可。

实际上，对于这类 $m-n$ 很小的图，广义串并联图是一种常用的工具。如果你不了解，可以上网搜索相关资料学习一下。以上的思路可以用广义串并联图方法的基本操作来描述：

- 第一种是缩一度点，直接缩就行，点数-1，边数-1，额外记录一下比如 c_x 不能与 c_y 相同，从 y 向 x 连一条有向边。
- 第二种是缩二度点，同样也是可以直接缩，点数-1，边数-2，记录 c_x 不能同时与 c_y, c_z 相同，从 y, z 分别向 x 连一条有向边。这里要特别注意不能按照一般的广义串并联图方法在原图上连上边 y, z，原因很显然。
- 第三种是叠合重边，但在这题中任何时刻根本不存在重边。

你会发现这样进行完后，最终每个点的度数都会大于等于3，那么在最终的图中就有 $3n \leq 2m$。

观察上述操作，发现点数减少的量始终少于边数，所以 $m \leq n+t$ 始终成立。

于是可以推导出最终的图有 $n \leq 2t, m \leq 3t$，图的规模就变得非常小了。

这时候有 $n \leq 2t = 16$，所以直接做 $O(3^{2t}\mathrm{poly}(t))$ 的爆搜就可以了。

还原的时候只需要根据刚才的拓扑图做一个拓扑排序即可，容易发现拓扑图是个DAG。

所以复杂度是 $O(3^{2t}\mathrm{poly}(t)+n+m)$，实现还是比较容易的。期望得分 100 分。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef double dou;
typedef pair<int,int> pii;
#define fi first
#define se second
#define mapa make_pair
typedef long double ld;
typedef unsigned long long ull;
template <typename T>inline void read(T &x){
    x=0;char c=getchar();bool f=0;
    for(;c<'0'||c>'9';c=getchar()) f|=(c=='-');
    for(;c>='0'&&c<='9';c=getchar())
    x=(x<<1)+(x<<3)+(c^48);
    x=(f?-x:x);
}
const int N=1e5+5;
int taskid, n, m, k, t;
vector<int> e[N];
namespace task1{
    int col[N];
    bool flg=0;
    inline void dfs(int x, int c){                          // 二分图染色
        col[x]=c;
        for(auto y:e[x]){
            if(col[y]!=-1){
                if(col[x]==col[y]){flg=1; return ;}
            }
            else{
                dfs(y, c^1);
            }
        }
    }
    void solve(){
        memset(col, -1, sizeof col);
        for(int i=1; i<=n; ++i) if(col[i]==-1){
            dfs(i, 0);
            if(flg){
                printf("-1\n"); return ;
            }
        }
        printf("1\n");
        for(int i=1; i<=n; ++i) printf("%d ", col[i]+1);
    }
}
namespace task2{                                            // 广义串并联图的三染色
    bool ban[N][4];
    unordered_map<int, bool> h[N];
    int que[N], hh, tt;
    int deg[N];
```

```
    vector<int> g[N];
    int col[N];
    bool ins[N];
    int bin[N], cnt, bin2[N], cnt2;
    bool suc=0;
    vector<int> e2[N];
    inline void dfs(int x){
        if(suc) return ;
        if(x==cnt+1){
            bool flg=1;
            for(int i=1; i<=cnt; ++i){
                int u=bin[i];
                for(auto v:e2[u]){
                    if(col[u]==col[v]) {flg=0; break;}
                }
                if(flg==0) break;
            }
            if(flg) suc=1;
            return ;
        }
        for(int i=1; i<=3; ++i) {
            col[bin[x]]=i;
            dfs(x+1);
            if(suc) return ;
        }
    }
    int fa[N];
    inline int get(int x){
        if(x==fa[x]) return x;
        return fa[x]=get(fa[x]);
    }
    inline void merge(int x, int y){
        x=get(x); y=get(y);
        if(x==y) return ;
        fa[x]=y;
    }
    vector<int> bel[N];
    void solve(){
        hh=1; tt=0;
        for(int i=1; i<=n; ++i) {
            for(auto j:e[i]) h[i][j]=1;
            if(e[i].size()<=2) que[++tt]=i, ins[i]=1;
        }
        while(hh<=tt){
            int x=que[hh++];
            if(h[x].size()==0) continue;            // 孤立点
            if(h[x].size()==1){                     // 1度点
                for(auto t:h[x]){
                    int y=t.fi;                     // 原图 x--y，删去 x，变成 y
                    h[y].erase(x);                  // 加限制 y->x，确定y颜色后，再确
定x
                    g[y].push_back(x);
                    ++deg[x];
                    if(h[y].size()<=2&&!ins[y]) que[++tt]=y, ins[y]=1;
                }
```

```
                continue;
            }                                           // 2度点
            int y=0, z=0;
            for(auto t:h[x]){
                int v=t.fi;
                if(y==0) y=v;
                else z=v;
            }
            h[y].erase(x); h[z].erase(x);           // 原图 y--x--z，，删去 x，变成y
z (不加边)
            g[y].push_back(x); g[z].push_back(x);   // 加限制 y->x, z->x
            deg[x]+=2;
            if(h[y].size()<=2&&!ins[y]) que[++tt]=y, ins[y]=1;
            if(h[z].size()<=2&&!ins[z]) que[++tt]=z, ins[z]=1;
        }

        hh=1; tt=0;
        for(int i=1; i<=n; ++i) fa[i]=i;
        for(int i=1; i<=n; ++i) if(deg[i]==0) bin2[++cnt2]=i, que[++tt]=i;
        memset(ins, 0, sizeof ins);
        for(int i=1; i<=cnt2; ++i) ins[bin2[i]]=1;
        for(int i=1; i<=cnt2; ++i){
            int x=bin2[i];
            for(auto y:e[x]) if(ins[y]) e2[x].push_back(y), merge(x, y);
        }
        for(int i=1; i<=cnt2; ++i) bel[get(bin2[i])].push_back(bin2[i]);
        for(int i=1; i<=n; ++i) if(bel[i].size()){    // 把缩点之后的图建出来，每个连
通块爆搜三染色
            cnt=0;
            for(auto t:bel[i]) bin[++cnt]=t;
            dfs(1);
            if(!suc) {
                printf("-1\n");
                return ;
            }
        }
        // 如果找到成功方案，那么根据g[]中的图，对剩余的点2染色
        // g[]是一个DAG，可以拓扑排序的过程中染色
        while(hh<=tt){
            int x=que[hh++];
            if(col[x]==0){
                col[x]=1;
                while(ban[x][col[x]]) ++col[x];
            }
            for(auto y:g[x]){
                ban[y][col[x]]=1;
                --deg[y];
                if(deg[y]==0) que[++tt]=y;
            }
        }
        printf("1\n");
        for(int i=1; i<=n; ++i) printf("%d ", col[i]);
    }
}
int main(){
```

```
    read(taskid);
    read(n); read(m); read(k); read(t);
    for(int i=1, x, y; i<=m; ++i){
        read(x); read(y);
        e[x].push_back(y); e[y].push_back(x);
    }
    if(taskid==1){
        if(m!=0) printf("-1\n");
        else {
            printf("1\n");
            for(int i=1; i<=n; ++i) printf("1 ");
        }
        return 0;
    }
    if(taskid==2){
        task1::solve();
        return 0;
    }
    task2::solve();
    return 0;
}
```

思路六：$t \leq 15$

图的三染色是存在一种基于随机化的做法的，详见OIWIKI-随机化技巧-例：三部图的判定，需要用到随机函数分析和2-sat。

可以进一步优化思路五中的暴力三染色部分，在本题中不作要求。

本题易错点

- 注意缩点时，原图中颜色的限制和新图中颜色限制的对应关系