陈文泉 pure__Elysia 11.14,11.15 总结

现在是，私活时间！！！

前言

可能有一点点格式问题，因为这是两个总结直接粘贴到一起的，要不。。将就看看？

陈文泉 pure__Elysia 11.15

	预估	实际	差值
T1	100	94	-6
T2	20	20	0
T3	10	8	-2
T4	10	5	-5
总分	140	127	-13

T1

考场：

一眼建图，所以最短路。想过 `SPFA`，发现莫得负环，所以 `DJ` 更优。又想到所有边权都是 1，所以选择搜索。易知，`BFS` 显然比 `DFS` 好写多了。果断选择爆发式（`BFS`）。

写完，发现这个搜索可行点的过程太慢，写了个如果跳出边界就 break 的剪枝。

应付大样例已经够了，但是把 `m` 改成 0，哦豁，T 飞。

而后发现，前后都是有边界的，应该再把前边界的判断也给写进去。于是写了个二分。。。。

马上意识到，这玩意儿不是可以分类讨论 $O(1)$ 算出来的吗，写啥二分啊。

个人觉得这玩意儿剪枝巨弔无比，而且大样例，自己和 SD 里搓数据老哥造的大样例都能飞快过。芜湖，以为过了。于是放心了。

贴一下考场代码，个人觉得很 NB 了。属于是考场使用暴力剪枝得分的第三名（LQY 100pts, CXM 97pts,然后就是我的 94pts，大家相差就是 1 个点。）已经很满意了。

```
#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;

#define int long long
#define ri register int

inline int rd()
{
    int a=0,f=1;
    char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) a=(a<<1)+(a<<3)+c-'0';
    return a*f;
}

inline void wr(int x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10+'0');
}
```

```
int n,k,m,s,inf;
int del[100050],cnt;
int dis[100050];
bool ban[100050];

inline bool check1(int u,int i)
{
    int v=k/2+i/2+u;
    if(v-k+1<=0) return 0;
    else return 1;
}

inline bool check2(int u,int i)
{
    int v=u-(k/2+i/2);
    if(v+k-1>n) return 0;
    else return 1;
}

inline int bin1(int u)
{
    if(k/2<=u) return 1;
    return cnt-u-1;
}

inline int bin2(int u)
{
    return bin1(n-u+1);
}

queue<int>que;
inline void bfs()
{
    que.push(s);
    dis[s]=0;
    while(!que.empty())
    {
        ri u=que.front();
        que.pop();

        int lim=bin1(u);
//      cout<<lim<<endl;
        for(ri i=lim;i<=cnt;i++)
        {
            if(k/2+del[i]/2+u>n) break;
            ri v=u+del[i];
            if(check1(u,del[i]))
            {
                if(ban[v]==0&&dis[u]+1<dis[v])
                {
                    dis[v]=dis[u]+1;
                    que.push(v);
                }
            }
        }
//      cout<<lim<<endl;
        lim=bin2(u);
        for(ri i=lim;i<=cnt;i++)
```

```
        {
            ri v=u-del[i];
//          cout<<u<<" "<<i<<" "<<u-(k/2+i/2)<<endl;
            if(u-(k/2+del[i]/2)<=0) break;
            if(check2(u,del[i]))
            {
                if(ban[v]==0&&dis[u]+1<dis[v])
                {
                    dis[v]=dis[u]+1;
                    que.push(v);
                }
            }
        }
    }
}

signed main()
{
    freopen("reverse.in","r",stdin);
    freopen("reverse.out","w",stdout);

    memset(dis,127,sizeof(dis));
    inf=dis[0];
    n=rd(),k=rd(),m=rd(),s=rd();

    for(ri i=(k&1)+1;i<=k;i+=2)
        del[++cnt]=i;

    for(ri i=1;i<=m;i++)
        ban[rd()]=1;

    bfs();

    for(ri i=1;i<=n;i++)
        if(dis[i]!=inf) wr(dis[i]),putchar(' ');
        else putchar('-'),putchar('1'),putchar(' ');

    return 0;

}
```

正解

思考，我们暴力慢在了哪里。很明显，因为我们写的是 `BFS`,而且边权都是 1。所以我们遍历过了点一定不用遍历了。但是由于我们要遍历到这个点才知道这个点有没有被遍历，这好猪猪。

需要一个容器，记录哪些点还可以遍历，支持删除。

当当当当~！新学会的 `STL` **set**

因为我们刚才剪枝写的很 NB（蜜汁自信），而且想到的思路已经够完备了。所以只需要把遍历的过程变成用迭代器与 `lower_bound` 遍历就行了。改动其实很小。

```
#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;
```

```cpp
#define int long long
#define ri register int

inline int rd()
{
    int a=0,f=1;
    char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) a=(a<<1)+(a<<3)+c-'0';
    return a*f;
}

inline void wr(int x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10+'0');
}

inline int Max(int a,int b){if(a>b)return a;else return b;}
inline int Min(int a,int b){return -Max(-a,-b);}

int n,k,m,s;
int dis[100050];
bool ban[100050];

set<int>zhinshu,oushu;

queue<int>que;
inline void bfs()
{
    que.push(s);
    while(!que.empty())
    {
        ri u=que.front();
        que.pop();
        int l=Max(1,u-k+1),r=Min(n,u+k-1);
        l=l+(l+k-1)-u,r=r+(r-k+1)-u;

        if(((u&1)&&(k&1))||((u%2==0)&&(k%2==0)))
        {
            for(auto i=zhinshu.lower_bound(l);*i<=r&&i!=zhinshu.end();)
            {
                dis[*i]=dis[u]+1;
                que.push(*i);
                auto j=i++;
                zhinshu.erase(j);
            }
        }
        else
        {
            for(auto i=oushu.lower_bound(l);*i<=r&&i!=oushu.end();)
            {
                dis[*i]=dis[u]+1;
                que.push(*i);
                auto j=i++;
                oushu.erase(j);
            }
```

```
        }

    }
}

signed main()
{
    freopen("reverse.in","r",stdin);
    freopen("reverse.out","w",stdout);

    n=rd(),k=rd(),m=rd(),s=rd();
    memset(dis,-1,sizeof(dis));
    dis[s]=0;

    for(ri i=1;i<=m;i++)
        ban[rd()]=1;

    for(ri i=1;i<=n;i++)
        if(ban[i]==0&&i!=s)
            if(i&1) zhinshu.insert(i);
            else oushu.insert(i);

    bfs();

    for(ri i=1;i<=n;i++)
        wr(dis[i]),putchar(' ');

    return 0;

}
```

T2

考场

看起来用人脑做很简单，最后发现实现巨难。于是选择加入暴力神教。模拟每~~一~~种可能并判断。于是豪取~~（大嘘~~ 20pts。

```
#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;

#define int long long
#define ri register int

inline int rd()
{
    int a=0,f=1;
    char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) a=(a<<1)+(a<<3)+c-'0';
    return a*f;
}

inline void wr(int x)
```

```
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10+'0');
}

int n,ans;
int zs[10],ls[10];
int mp[10][10];

inline void check()
{
    bool b1=0;
    for(ri i=1;i<=n;i++)
    {
        b1=0;
        for(ri j=1;j<=n;j++)
        {
            if(mp[i][j]==ls[i]) b1=1;
        }
        if(b1==0) return ;
    }

    for(ri i=1;i<=n;i++)
    {
        b1=0;
        for(ri j=1;j<=n;j++)
        {
            if(mp[j][i]==zs[i]) b1=1;
        }
        if(b1==0) return ;
    }

    ans++;
    return ;
}

inline void test(int x,int y)
{
    for(ri i=0;i<=min(ls[x],zs[y]);i++)
    {
        mp[x][y]=i;
        if(x==n&&y==n) check();
        else
        {
            if(y<n)
                test(x,y+1);
            else
                test(x+1,1);
        }
    }
}

signed main()
{
    freopen("silhouette.in","r",stdin);
    freopen("silhouette.out","w",stdout);
```

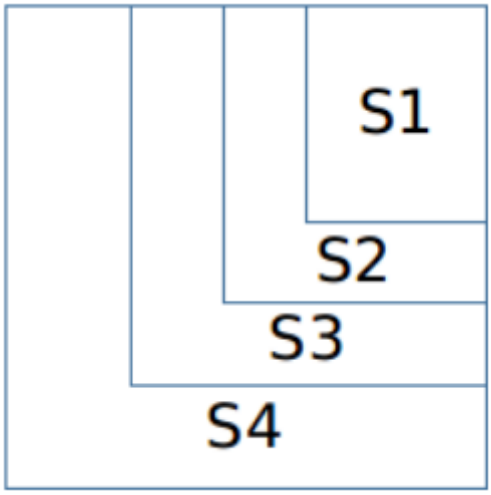
```
    n=rd();

    if(n==1)
    {
        puts("1");
        return 0;
    }

    if(n<=3)
    {
        for(ri i=1;i<=n;i++) zs[i]=rd();
        for(ri i=1;i<=n;i++) ls[i]=rd();

        test(1,1);

        wr(ans);

        return 0;
    }

    if(n==99995)
    {
        puts("401080963");
        return 0;
    }

    return 0;

}
```

正解

意识到，其实输入的大小顺序其实并不重要。的确，不一样的顺序会使得构造的图案不一样，但是我们统计的可能数是一样的。于是毫不留情，给他排序喽。

注意，如果正视图和左视图最大值不一，要写一个特判，输出 0。不然等下赋初始值会出问题。

我们排好了序，就会形成一个长成这样的"期望"图案。



这时，我们要清楚，去计算有多少合理，真的不好算。但其实去计算有多少不合法，倒是不难。因为我们不仅可以直接去计算，而且可以 dp ，方便容斥。

这里我太懒，直接把 dp 方程放上来了，就不推到了：

$$f[i]=C_a^i\times(S^i\times((S+1)^{a-i}-S^{a-i}))^b$$

由于我们处理完一个块后，下一个块更大，所以答案更新是一个“L”型。我们左上角叫做 a, 逆时针转依次是 b,c,d。那么转移如下：

$$f[i]=C_a^i\times(S^i\times((S+1)^{a+c-i}-S^{a+c-i}))^b\times(S^i\times(S+1)^{a-i})^d$$

统计答案，累加，为：

$$res=\sum_{i=0}^{a}(-1)^i\times f[i]$$

我的评价是：这是什么恐怖的阴间 dp 容斥题。

```
#pragma GCC optimize(2)
#include <bits/stdc++.h>
using namespace std;

#define int long long
#define ri register int

const int P=1e9+7;

inline int rd()
{
    int a=0,f=1;
    char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) a=(a<<1)+(a<<3)+c-'0';
    return a*f;
}

inline void wr(int x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10+'0');
}

int n,c;
int jc[500050],inv[500050];
int A[500050],B[500050],num[1000050];

inline int qp(int a,int b)
{
    int res=1;
    while(b)
    {
        if(b&1) res=(res*a)%P;
        a=(a*a)%P;
        b>>=1;
    }
    return res;
}
```

```
inline int C(int x,int y)
{
    return jc[x]*inv[y]%P*inv[x-y]%P;
}

inline int Lucas(int x,int y)
{
    if(!y) return 1;
    return C(x%P,y%P)*Lucas(x/P,y/P)%P;
}

inline int work(int a,int b,int c,int d,int s)
{
    int res=0;
    for(ri i=0;i<=a;i++)
    {
        int now=Lucas(a,i)*qp(qp(s,i)*((qp(s+1,a+c-i)-qp(s,a+c-
i)+P)%P)%P,b)%P*qp(qp(s,i)*qp(s+1,a-i)%P,d)%P;
        if(i&1) res=(res-now+P)%P;
        else res=(res+now)%P;
    }
    return res;
}

signed main()
{
    freopen("silhouette.in", "r", stdin);
    freopen("silhouette.out", "w", stdout);

    n=rd();
    for(ri i=1;i<=n;i++)
        A[i]=rd(),num[i]=A[i];
    for(ri i=1;i<=n;i++)
        B[i]=rd(),num[n+i]=B[i];

    jc[0]=1;
    for(ri i=1;i<=n;i++) jc[i]=jc[i-1]*i%P;
    inv[n]=qp(jc[n],P-2);
    for(ri i=n;i>=1;i--) inv[i-1]=inv[i]*i%P;

    sort(A+1,A+n+1);
    sort(B+1,B+n+1);
    sort(num+1,num+2*n+1);
    c=unique(num+1,num+2*n+1)-num-1;

    if(A[n]!=B[n])
    {
        puts("0");
        return 0;
    }

    int la=n+1,lb=n+1,ra=n,rb=n;
    int ans=1;
    for(ri i=c;i>=1;i--)
    {
        while(A[ra-1]==num[i]&&ra-1) ra--;
        while(B[rb-1]==num[i]&&rb-1) rb--;
        ans=(ans*work(la-ra,lb-rb,n-la+1,n-lb+1,num[i]))%P;
```

```
            la=ra,lb=rb;
        }

        wr(ans);

        return 0;

    }
```

T3

考场：

首先写了个大数据输出。然后想了很久，观察了一下数据范围，发现有 30 pts 可以状压。然后一想，诶，开 `long long` 或者 `__int128` 甚至有 40、60 pts。

那我为什么没拿 60 分捏？因为 10 分钟写不完了捏。

正解

施工中

T4

考场

哈哈， @hycc 学长讲过。哈哈，想起来是中国剩余定理。

哈哈，我忘辣！我不会！哈哈，哈哈哈哈哈哈，哈哈（笑不出来。。。哭辣

果断选择 puts 大样例骗分。

正解

题面很长，很有意思，也很好理解。可是 OI 是残酷的，我们把他的本质揭露出来：

给两个数， n, g 。你最终的答案是一个指数式， g 是底数。指数是什么？指数是一个和，每有一个 n 的因数 d ，这个和就加上一个组合数 C_n^d 。写出来就是

$$g^{\sum_{k|n} *C_n^k} \equiv ans(modP)$$

我们想办法求这个数，首先这个指数很明显长得就不对劲，而且老老实实算出来肯定太大了，要不得。在取模意义下的大指数运算，考虑欧拉函数。

欧拉函数，即使得指数部分可以缩减成在对（ P-1 ）取模的形式。于是目标就变成了求解如下式子。

$$\sum_{k|n} *C_n^k \mod (P-1)$$

观察到 n 虽然大，但是搜索因数只用根号级，去枚举因数的复杂度是可以接受的。但是这个组合数有点小烦啊。大组合数在取模意义下求解，需要用到 Lucas 定理。即使得在取模 p 时，以下成立：

$$C_b^a \equiv C_{b/p}^{a/p} * C_{b\%p}^{a\%p} \ (mod\ p)$$

But Lucas 定理成立的条件是 p 是 质 数。我们的 P=999911659， p=P-1=999911658。上过小学的都会跟你说这玩意儿不是质数。那咋办？~~（扩展卢卡斯）~~

发现，$999911658 = 2 * 3 * 4679 * 35617$ ，我们可以把这四个分别干碎，最后用中国剩余定理来合并答案。

因为刚才这几个 2，3，4679，35617 都只有一次，所以互质，而且都是质数嘛，就可以直接中国剩余定理硬钢，不用扩展。

计算组合数时，可以用预处理阶乘的方式，

算完了，重复一遍流程：

> 1.预处理出四个质因数的组合数阶乘。
>
> 2.在根号内，枚举出 n 的因数。
>
> 3.每枚举到一个（实际为两个或一个，毕竟因数成对出现嘛）因数，就用 Lucas 计算 4 次组合数，记录进对应的四个线性同余方程。
>
> 4.枚举完后，用 CRT 处理同于方程，合并答案。
>
> 5.以 g 为底数，刚才的答案为指数，用快速幂计算出最后答案。

做完力，上代码：

```
#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;

#define int long long
#define ri register int

const int P=999911658;

int n,g;
int a[5],b[5]={0,2,3,4679,35617};
int ans;
int jc[40005][5];

inline void init(int p,int f)
{
    jc[0][f]=jc[1][f]=1;
    for(ri i=2;i<=p;i++)
        jc[i][f]=(jc[i-1][f]*i)%p;
    return ;
}

inline int qp(int a,int b,int p)
{
    int res=1;
    while(b)
    {
        if(b&1) res=(res*a)%p;
        a=(a*a)%p;
        b>>=1;
    }
    return res;
}
```

```
inline int inv(int a,int p)
{
    return qp(a,p-2,p);
}

inline int C(int x,int y,int p,int f)
{
    if(x==y||y==0) return 1;
    if(y>x||x==0) return 0;
    int up=jc[x][f],down=jc[y][f]*jc[x-y][f]%p;
    return (up*inv(down,p))%p;
}

inline int Lucas(int x,int y,int p,int f)
{
    if(!x) return 1;
    return (Lucas(x/p,y/p,p,f)*C(x%p,y%p,p,f))%p;
}

inline void CRT()
{
    for(ri i=1;i<=4;i++)
    {
        ri m=P/b[i];
        ans+=((a[i]*m%P)*inv(m,b[i]))%P;
        ans%=P;
    }
    return ;
}

signed main()
{
    freopen("ancient.in", "r", stdin);
    freopen("ancient.out", "w", stdout);

    cin>>n>>g;

    g%=P+1;
    for(ri i=1;i<=4;i++) init(b[i],i);
    for(ri i=1;i*i<=n;i++)
        if(n%i==0)
        {
            for(ri j=1;j<=4;j++)
            {
                a[j]=(a[j]+Lucas(n,i,b[j],j))%b[j];
                if(i*i!=n)
                    a[j]=(a[j]+Lucas(n,n/i,b[j],j))%b[j];
            }
        }

    CRT();

    cout<<qp(g,ans,P+1)<<endl;

    return 0;

}
```

总结

1.暴力也并不是一定要靠减少搜索次数、特判、用那些 `break\continue` 去剪枝，也可以尝试使用数据结构维护。

2.多了解更多数据结构，尤其是 `STL` 里的。今天才刚刚学会 `set` ，大寄。

3.上过的知识点一定要写熟。比如之前的 Lucas ， CRT 。现在是基本都忘了，得额外花时间去复习。

4.打暴力、放弃、实现代码一定要干脆快速，给其他题预留空间。比如 T3 想到了高分暴力，但是完全莫得时间。这种情况尽量避免。

陈文泉 pure_Elysia 11.14

	预估	实际	差值
T1	10	10	0
T2	40	40	0
T3	10	10	0
T4	20	46	+26
总分	80	106	+26

T1

考场

别问，问就是最后写的 T1。因为一眼字符串，淦呐，我的弱项，润！

后面来看，瞪了 20min。意识到了 WOW！ 是 AC 自动机，字典树。

但是又怎样呢？我一样写不来~~（自卑子家大佬们）~~，况且只剩半个小时了。我的选择是特判大数据。

正解

还真是 AC 自动机，字典树。但是，，，没改完，又）施工中。。。。

T2

考场

首先，如果没有取模这个怪玩意儿，就是一个简简单单的 RMQ 问题。但是烦就烦在有一个取模。但是我们知道，这个取模的模数“一般来说”比较大。而我们的数“一般来说没那么大”。所以我们写一个 ST 表实现$O(1)$ 询问。如果这个区间的最大数小于模数，就可以直接输出，否则暴搜。

出人意料，这个思路比纯暴力高了 20pts。舒服了

是哪个小可爱忘记保存考场代码了啊？
哦，是我啊
那没事了

正解

首先感谢 @GMT`FFF 孙居（一声 的讲解。

我们发现，其实 ST 表不是那么的好去处理，其实分块就够了。

为什么呢？因为这次的“模数”也被我们记入成了块中的一个状态。观察 k 范围，发现 k 其实很小。这不仅否定了我们刚才剪枝的想法，也给我们提供了新思路。将模数也记入块中。

那么这题就很简单了，就是纯纯暴力维护就好。之所以选择分块，其实也很简单。因为 ST 表不好写预处理，线段树开的空间太大，也不好写，分块虽然慢一点点，但是够用就行。~~（8848钛合金手机：跑得快不一定赢，不跌跟头才算成功。~~

```
#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;

//#define int long long
#define ri register int

inline void rd(int &x)
{
    int a=0,f=1;
    char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) a=(a<<1)+(a<<3)+c-'0';
    x=a*f;
}

inline void wr(int x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10+'0');
}

inline int Max(int a,int b){if(a<b)return b;else return a;}
inline int Min(int a,int b){if(a<b)return a;else return b;}

const int N=1e5+5;
const int M=320;
int n,m,a[N],ans[M][N],vis[N],p[N],k;

signed main()
{
    freopen("flower.in","r",stdin);
    freopen("flower.out","w",stdout);

    rd(n),rd(m);

    k=1000;
    for(ri i=1;i<=n;i++)
        rd(a[i]),p[i]=(i-1)/k+1;
```

```
    for(ri i=1;i*(k-1)+1<=n;i++)
    {
        for(ri j=(i-1)*k+1;j<=Min(i*k,n);j++)
            vis[a[j]]=a[j];
        for(ri j=1;j<=N-5;j++)
            vis[j]=Max(vis[j],vis[j-1]);

        for(ri j=1;j<=N-5;j++)
            for(ri t=0;t<=N-5;t+=j)
                ans[i][j]=Max(ans[i][j],vis[Min(N-5,t+j-1)]-t);

        for(ri j=1;j<=N-5;j++)
            vis[j]=0;
    }

    while(m--)
    {
        ri l,r,t;
        rd(l),rd(r),rd(t);
        ri res=0;

        for(ri i=l;i<=Min(r,p[l]*k);i++)
            res=Max(res,a[i]%t);
        for(ri i=p[l]+1;i<p[r];i++)
            res=Max(res,ans[i][t]);
        for(ri i=Max(l,(p[r]-1)*k+1);i<=r;i++)
            res=Max(res,a[i]%t);

        wr(res),puts("");
    }

    return 0;

}
```

T3

考场

什么鬼玩意儿？太恐怖力，输出样例，开润。

正解

双）好多题，改不完力，施工中。。。

T4

考场：

每次染色都是染一个连通块，也就是说每个连通块都可以一起染色。这肯定是最节约的策略。但是为什么还会有更小的呢？因为通过多次染色，把不在一个连通块的多个连通块连一起了，于是乎更优。

但是这怎么写啊？写不来。

于是，我们选择我们刚才想到的“每次染完一个连通块”的伪算法，企图骗一点分。

这个算法也很好写，只需要用并查集连边统计黑色块数和白色块数就行。有可能白色块数很少，我们除了黑色块数外，还有可能是整个画布全部染成黑色后，再去染白色。即答案是 1+白色块 也是可能的。

```
#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;

//#define int long long
#define ri register int

inline void rd(int &x)
{
    int a=0,f=1;
    char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) a=(a<<1)+(a<<3)+c-'0';
    x=a*f;
}

inline void wr(int x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10+'0');
}

int n,m;

bool mp[60];
int fa[60];

inline int fi(int x)
{
    if(x==fa[x]) return x;
    else return fa[x]=fi(fa[x]);
}

signed main()
{
    freopen("paint.in","r",stdin);
    freopen("paint.out","w",stdout);

    rd(n),rd(m);

    for(ri i=1;i<=n;i++)
    {
        for(ri j=1;j<=m;j++)
            mp[(i-1)*m+j]=(getchar()=='1'),fa[(i-1)*m+j]=(i-1)*m+j;
        getchar();
    }

    for(ri i=1;i<=n*m;i++)
    {
        if(i%m)
        {
            int a=i,b=i+1;
```

```
            if(mp[a]==mp[b])
            {
                a=fi(a),b=fi(b);
                if(a!=b) fa[a]=b;
            }
        }

        if(i+m<=n*m)
        {
            int a=i,b=i+m;
            if(mp[a]==mp[b])
            {
                a=fi(a),b=fi(b);
                if(a!=b) fa[a]=b;
            }
        }
    }

    int bcnt=0,wcnt=0;
    for(ri i=1;i<=n*m;i++)
        if(fa[i]==i)
            if(mp[i]) bcnt++;
            else wcnt++;

    if(wcnt+1<=bcnt) wr(wcnt+1);
    else wr(bcnt);

    return 0;

}
```

正解

叒）施工中。。。

总结

1.遇到不熟的算法，要么狠心跳过，干脆一点；要么不惧艰难，使劲去推！

但是！这都比不上考试前认认真真补齐弱项来的划算！所以！快去复习 AC 自动机和字典树。

2.不要先入为主，先到一个离线快就去一味地想 ST 表。多想想现在学过的方法，会有更加优秀的新思路。

3.写不出正解，伪算法也是值得一试的，T4 他给了 46pts ，实在是太多了。大胆猜测，谨慎证明。

总总结

1.还是有很多薄弱项，太拖后腿了。甚至想到了方法写不出来。这主要归功于：①网课不认真，讲了题和知识点没有摸透。②居？然？能？忘？猪猪！

2.还是没有去想正解的“勇敢的心↑”，总是想着打暴力剪枝。虽然这不是错的，而且如果写不出正解这还是最好的归宿。但是怎么说呢？有的题我感觉还是有能力想出来正解的，还是要自信一点，尝试一下。

3.终究是资历少了一点，还有知识点没有学到。比如周六的最小生成树除了 P式，K式，居然还有 B式，还有就是昨天的 set。

4.总结？一下目前的薄弱项，下次自习时去洛谷过过板题：字符串：AC自动机、字典树。STL数据结构：set 。数论：CRT、Lucas、欧拉函数。其他算法：容斥原理。

现在是，私活时间！！！

十三英桀最美好的时光：