2022.11.14

题号	预计分数	实际分数	差值
substr	20	10	-10
flower	40	35	-5
refract	0	5	+5
paint	0	10	+10

~~状态有亿点差~~（~~分数差不多是这么多，记不清了~~）

T1

写了个花里胡哨的暴力，然后遗憾创鬼（指多加一个输出就会re）

而且也是错误的暴力（指大样例完全没有输出）

到现在没搞明白为什么会这样（提前跳出也会寄）

正解

状压dp加上AC自动机（话说这玩意真的会考吗），银牌佬说纯状压dp也能卡过去（但我不会）

施工中（~~keep out~~）

T2

打了个很纯粹的暴力，然后加了一点屁用没有的优化，成功掉了五分（虽然都差不多）

正解是分块

分析

考虑用值域进行分块，因为每次都要取模k，所以不能单纯的找最大值，首先在每次询问之前处理出一个块内的最优答案，块的大小我取的是$\sqrt{max(a_i)}$，用一个$last$数组存储一个值前面的比他大的最大值，然后用$ans[i][j]$来统计当模数为j时当前块内的最优答案

询问时处理出左右端点所属块，同一块直接枚举，不同块将两者之间的块检索一遍，在把两者到其所在块的端点检索一遍即可求出答案

```
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;
ll re(){
    ll s=0,w=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')w=-w;c=getchar();}
    while(isdigit(c)){s=(s<<1)+(s<<3)+(c^48);c=getchar();}
    return s*w;
}
const int D=1e5+114;
int BLOCK;
int Max(int x,int y){return x>y?x:y;}
```

```
int Min(int x,int y){return x<y?x:y;}
int n,m,maxx;
int a[D+1],lst[D+1];
int ans[130][D+1];

int main()
{
    n=re();m=re();
    for(int i=0;i<n;i++){a[i]=re();maxx=Max(maxx,a[i]);}
    BLOCK=(int)sqrt(maxx-1)+1;
    int fk=(n-1)/BLOCK+1;
    for(int i=0;i<fk;i++){
        memset(lst,0,sizeof lst);
        for(int j=i*BLOCK;j<(i+1)*BLOCK&&j<n;j++)lst[a[j]]=a[j];
        for(int j=1;j<=maxx;j++)lst[j]=Max(lst[j],lst[j-1]);
        for(int j=1;j<=maxx;j++)
            for(int k=0;k<=maxx;k+=j)
                ans[i][j]=Max(ans[i][j],lst[Min(k+j-1,maxx)]-k);
    }
    while(m--){
        int k,l,r;
        l=re();r=re();k=re();
        l--;r--;
        int x=(l/BLOCK)+1;
        int y=(r/BLOCK);
        int res=0;
        if(x==y+1){for(int i=l;i<=r;i++){res=Max(res,a[i]%k);}}
        else{
            for(int i=x;i<y;i++)res=Max(res,ans[i][k]%k);
            for(int i=l;i<x*BLOCK;i++)res=Max(res,a[i]%k);
            for(int i=r;i>=y*BLOCK;i--)res=Max(res,a[i]%k);
        }
        printf("%d\n",res);
    }
    return 0;
}
```

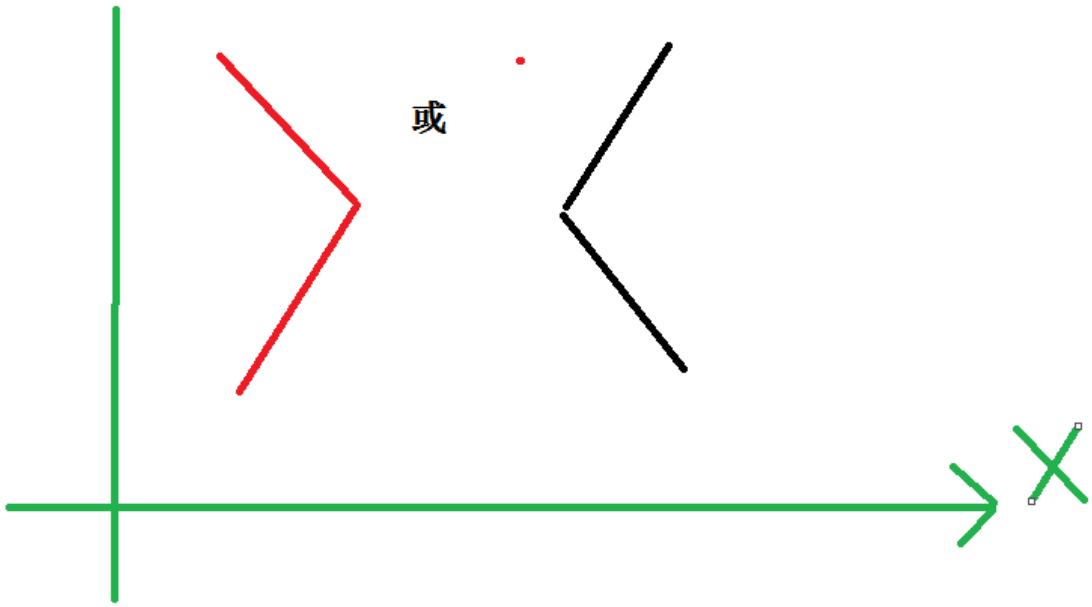
T3

很硬气的说：~~暴力都没打出来~~

分析

看到题面上所说的y坐标单调，很容易考虑用y进行排序之后去统计答案，但是错的（逃

因为排序之后没有了y的影响，还有x的，而且x的影响更难判断，所以我们选择用x去排序，然后枚举两个点，一个是当前点，一个是它前面的点（因为按x从小到大拍序，所以当前点的横坐标一定比它前面的点大），由$\forall j \in (2,k], x[j-2] < x[j] < x[j-1]$或 $x[j-1] < x[j] < x[j-2]$，我们可以得到不考虑y时的折线形状：



这时候我们考虑一个$dp[i][j]$，j这一维为1代表当前点将要向左折，为0代表向右折，那么转移过来时，上一个点到当前点向右的话，当前点下一次就必须向左，那么我们每次用y来限制走向，如果我们枚举的点比它前面的点矮，那么它就只能向它前面的点走，反之亦然，转移方程也就可以推出来了，每个点在一开始都要给它两种情况各一种方案数，（所以后面统计答案时还要减去n中方案），统计答案直接加上每个点接下来向右和向左两个情况数

```
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;
ll re(){
    ll s=0,w=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')w=-w;c=getchar();}
    while(isdigit(c)){s=(s<<1)+(s<<3)+(c^48);c=getchar();}
    return s*w;
}
const int mod=1e9+7;
const int D=6e3+114;
struct node{
    int x,y;
}dog[D];
bool cmp(node a,node b){return a.x!=b.x?a.x<b.x:a.y<b.y;}
int n;
ll f[2][D],sum[D];
ll ans;

int main()
{
    freopen("refract.in","r",stdin);
    freopen("refract.out","w",stdout);
    n=re();
    for(int i=1;i<=n;i++){dog[i].x=re();dog[i].y=re();}
    sort(dog+1,dog+1+n,cmp);

    for(int i=1;i<=n;i++){
        f[0][i]=f[1][i]=1;
        for(int j=i-1;j>=1;j--){
            if(i==j)continue;
```

```
            if(dog[i].y<dog[j].y){f[1][j]=(f[1][j]+f[0][i])%mod;}
            else if(dog[i].y>dog[j].y){f[0][i]=(f[0][i]+f[1][j])%mod;}
        }
    }
    for(int i=1;i<=n;i++){
        ans=(ans+f[0][i]+f[1][i])%mod;
    }
    ans=(ans-n)%mod;
    printf("%lld\n",ans);

    return 0;
}
```

T4

考场时同$T3$

分析

我们可以随意选择一个任意大小的连通块将其改变颜色，贪心的思考，那么第一次选择把所有要变黑的点包括的连通块，后面选择的一定是上一次选择的连通块的子集，考虑宽搜去找出最少下笔数

宽搜时，我们找离当前点最远的黑点，朝那边行动并且由此记录步数，同色不变，不同色步数加一，每个点都拿去遍历一遍，在每个点的最大答案中找到一个最小值就可以力！

```
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;
ll re(){
    ll s=0,w=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')w=-w;c=getchar();}
    while(isdigit(c)){s=(s<<1)+(s<<3)+(c^48);c=getchar();}
    return s*w;
}
int Max(int x,int y){return x>y?x:y;}
int Min(int x,int y){return x<y?x:y;}
const int D=60;
const int dx[]={0,1,0,-1};
const int dy[]={1,0,-1,0};
char cat[D][D];
char dog[D][D];
int x[D],y[D];
int n,m;
int dis[D][D];
struct node{int x,y;};
deque<node>q;
int bfs(int x,int y){
    memset(dis,-1,sizeof dis);
    dis[x][y]=0;
    q.push_back(node{x,y});
    int res=0;
    while(!q.empty()){
        int u=q.front().x;
        int v=q.front().y;
        q.pop_front();
        if(cat[u][v]=='1')
```

```
            res=Max(res,dis[u][v]);
        for(int i=0;i<4;i++){
            int ux=u+dx[i];
            int vy=v+dy[i];
            if(ux>0&&vy>0&&ux<=n&&vy<=m&&dis[ux][vy]==-1){
                if(cat[ux][vy]==cat[u][v]){
                    dis[ux][vy]=dis[u][v];
                    q.push_front(node{ux,vy});
                }
                else{
                    dis[ux][vy]=dis[u][v]+1;
                    q.push_back(node{ux,vy});
                }
            }
        }
    }


    return res;
}


int main()
{
    n=re();m=re();
    for(int i=1;i<=n;i++){
        scanf("%s",cat[i]+1);
    }
    int ans=0x3f3f3f3f;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            ans=Min(ans,bfs(i,j));
        }
    }
    printf("%d\n",ans+1);
    return 0;
}
```

总·总结

相较于另外两天，这天的题其实要简单一点（其实也不算简单），但是还是寄掉了（~~那确实是我的问题子罢~~），像类似于T2的分块，能想到，但是实现起来还是力有不逮，实力还是没到位，而类似于T4这种将问题转化的，也不是很熟练，需要在脑子里建立起相关的思维，不能看到题吊死在一个惯性思维上

差不多就这样（