20221114测试总结

分数

题目	预计分数	实际分数	Delta	期望分数
最短母串(substr)	10	42	-32	42
要养花(flower)	20	20	0	100
折射(refract)	10	0	10	100
画作(paint)	?	40	?	40

题解

最短母串(substr)

To be Continued.

要养花(flower)

首先由于取模，无法使用树形结构进行快速维护并合并区间信息。这样一来，块状结构成为了我们的选择。

对于询问的区间，其答案就是在值域区间 $[ak,(a+1)k)$ 中的最大者。所以不妨对于每个块，事先预处理每个块对于每个模数的最大值，这样一来询问就变得非常简单。完整的块直接询问，不完整的块暴力获取即可。

最后在这里讲一下预处理部分。我们定义一个最大前缀数组 $\mathrm{maxa}[i]$，表示 1 到 i 中的最大元素。所以我们枚举每个块，然后求出每个块里面的 $\mathrm{maxa}[i]$，最后枚举模数 j 以及倍数 k，那么块 i 中在模 j 下的最大数 $\mathrm{ans}[i][j]$ 就等于：$\mathrm{ans}[i][j]=\max\{\mathrm{ans}[i][j],\mathrm{maxa}[k+j-1]-k\}$，其中每次都有 $k=k+j$，即 aj，而 $-k$ 起到取余的作用。

代码如下：

```
#pragma GCC optimize(3)
#include <cstdio>
#include <cstring>
using namespace std;
int const maxn=100000;
int n,m;
int a[100005];
int maxa[100005];
int ans[105][100005];
inline int max_(int a,int b){return a>b?a:b;}
inline int min_(int a,int b){return a<b?a:b;}

inline int re(){
    int k=0,f=1;
    char cre=getchar();
    while(!('0'<=cre&&cre<='9')){
        if(cre=='-') f=-1ll;
```

```
        cre=getchar();
    }
    while('0'<=cre&&cre<='9'){
        k=(k<<1ll)+(k<<3ll)+(cre^48ll);
        cre=getchar();
    }
    return 1ll*k*f;
}

void wr(int x){
    if(x<0){
        putchar('-');
        x=~x+1;
    }
    if(x>9) wr(x/10ll);
    putchar(x%10ll^48ll);
}

inline void solve(){
    for(int j=1;j<=maxn;++j)
        maxa[j]=max_(maxa[j],maxa[j-1]);        //最大前缀（类似于前缀和）
}

inline void solve1(int i){
    for(int j=1;j<=maxn;++j)
        for(int k=0;k<=maxn;k+=j)
            ans[i][j]=max_(ans[i][j],maxa[min_(k+j-1,100000)]-k);       //获取ans
数组，k+j-1意味着在ak到(a+1)k区间内，-k起到取余的作用
}

int const B=1000;
signed main(){
    n=re(),m=re();
    for(int i=0;i<n;++i) a[i]=re();
    //处理对于每个块，每个余数所对应的最大数
    int num=(n-1)/B+1;      //块的个数
    for(int i=0;i<num;++i){
        memset(maxa,0,sizeof(maxa));
        for(int j=i*B;j<min_(n,(i+1)*B);++j) maxa[a[j]]=a[j];   //预处理最大前缀
        solve();
        solve1(i);
    }
    while(m--){
        static int l,r,k;
        l=re(),r=re(),k=re();
        --l,--r;
        int x=l/B+1,y=r/B;
        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]);
            for(int i=l;i<x*B;++i) res=max_(res,a[i]%k);
            for(int i=r;i>=y*B;--i) res=max_(res,a[i]%k);
        }
        wr(res);
        putchar('\n');
    }
```

```
    return 0;
}
```

画作(paint)

首先给出结论：存在一种最优方案使得每次操作的区域是上一次的子集且颜色与上一次相反。

设 S 为之前包括当前操作区域的并，T 为下一个操作区域。接下来考虑证明上述结论：

- 如果 $S\cap T\neq\emptyset$，那么 T 一定可以转化为 S 的子集。
- $S\cap T=\emptyset$ 时，我们可以考虑一个区域 M，将 S 和 T 连接起来，这样我们直接操作 $S\cup M\cup T$，并再操作 $S\cup M$，这样一来答案并不会变劣。

剩下的我们只需要枚举每个最后操作的区域，然后同色区域边权设为 0，异色为 1。跑一个 01-bfs 之后答案就是对于枚举的点最远的黑色点的距离，最后对这些答案取最小值即可。

```
#include <deque>
#include <cstdio>
#include <cstring>
#include <iostream>
#define ll long long
using namespace std;
int n,m;
int ans=2e9;
char a[55][55];
int dx[]={0,1,0,-1};
int dy[]={1,0,-1,0};

inline ll re(){
    ll k=0,f=1;
    char cre=getchar();
    while(!('0'<=cre&&cre<='9')){
        if(cre=='-') f=-1ll;
        cre=getchar();
    }
    while('0'<=cre&&cre<='9'){
        k=(k<<1ll)+(k<<3ll)+(cre^48ll);
        cre=getchar();
    }
    return 1ll*k*f;
}

void wr(ll x){
    if(x<0){
        putchar('-');
        x=~x+1;
    }
    if(x>9) wr(x/10ll);
    putchar(x%10ll^48ll);
}

int dis[55][55];
inline int bfs(int sx,int sy){
    int res=0;
    deque< pair<int,int> > d;
    d.push_front({sx,sy});
    memset(dis,-1,sizeof(dis));
```

```
    dis[sx][sy]=0;
    while(d.size()){
        int x=d.front().first;
        int y=d.front().second;
        d.pop_front();
        if(a[x][y]=='1') res=max(res,dis[x][y]);
        for(int i=0;i<=3;++i){
            int ux=x+dx[i];
            int uy=y+dy[i];
            if(ux>=1&&ux<=n&&uy>=1&&uy<=m&&dis[ux][uy]==-1){
                if(a[x][y]==a[ux][uy]){
                    dis[ux][uy]=dis[x][y];
                    d.push_front({ux,uy});
                }
                else{
                    dis[ux][uy]=dis[x][y]+1;
                    d.push_back({ux,uy});
                }
            }
        }
    }
    return res;
}

signed main(){
    n=re(),m=re();
    for(int i=1;i<=n;++i)
        for(int j=1;j<=m;++j)
            cin>>a[i][j];
    for(int i=1;i<=n;++i)
        for(int j=1;j<=m;++j)
            ans=min(bfs(i,j),ans);
    wr(ans+1);
    return 0;
}
```