错题集 4

1、小V的序列

[链接]

题意

给定一个长度为 n 的序列，保证序列中每个数取值随机。

接下来给定 m 个询问，每次给定一个数 b问序列中是否存在数与 b 异或后二进制表示中 1 的个数不超过 3。

题解

考虑将 b 的二进制表示分成四部分，每部分表示一个长度为 16 二进制数。

则如果要与 b 异或后二进制表示中 1 的个数不超过 3，则至少四部分要有一个部分与 b 相同。

考虑将序列中的每个数分成四个，每个数投入对应位置二进制数的桶中，然后暴力查询，时间复杂度 $O\left(4\cfrac {nm}{2^{16}}\log v\right)$

```
const int MAXN=1e6+5,MAXM=1<<16,Mod=998244353;
LL a[MAXN];
vector<LL> c[4][MAXM];
uint64_t G(uint64_t x){
    x^=x<<13;
    x^=x>>7;
    x^=x<<17;
    return x;
}
bool check(LL x){
    int cnt=0;
    _for(i,0,64){
        if((x>>i)&1)
        cnt++;
    }
    return cnt<=3;
}
int main()
{
    int n=read_int(),m=read_int();
    a[0]=read_LL();
    _for(i,1,n)a[i]=G(a[i-1]);
    _for(i,0,n){
        _for(j,0,4)
```

```
        c[j][(a[i]>>(16*j))&((1<<16)-1)].push_back(a[i]);
    }
    int ans=0;
    while(m--){
        LL b=read_LL();
        bool flag=false;
        _for(i,0,4){
            int t=(b>>(16*i))&((1<<16)-1);
            _for(j,0,c[i][t].size()){
                if(check(c[i][t][j]^b)){
                    flag=true;
                    break;
                }
            }
            if(flag)
            break;
        }
        ans=(ans<<1|flag)%Mod;
    }
    enter(ans);
    return 0;
}
```

2、Sasha and Array

链接

题意

给定一个长度为 n 的序列 A，接下来两种操作，操作 1 为区间加，操作 2 为区间斐波那契和查询。

其中，定义 $f(1)=f(2)=1,f(n)=f(n-1)+f(n-2)$，$[l,r]$ 区间斐波那契和为 $\sum_{i=l}^r f(a_i)$

题解

对每个结点，维护区间矩阵和 $\begin{pmatrix}a_n \\a_{n+1}\\ \end{pmatrix}$ 以及 2×2 的区间懒标记。

于是区间加 v 转化为对区间 $[l,r]$ 的每个矩阵乘上 $\begin{pmatrix}0 & 1 \\ 1 & 1\\ \end{pmatrix}^v$。时间复杂度 $O(m\log n\log v)$。

```
const int MAXN=1e5+5,MAX_size=2,MAXS=35,Mod=1e9+7;
struct Matrix{
    int r,c,ele[MAX_size][MAX_size];
    Matrix(int r=0,int c=0){
        this->r=r,this->c=c;
```

```
        mem(ele,0);
    }
    Matrix operator + (const Matrix &b)const{
        Matrix C;
        C.r=r,C.c=c;
        _for(i,0,r)_for(j,0,c)
        C.ele[i][j]=(ele[i][j]+b.ele[i][j])%Mod;
        return C;
    }
    Matrix operator * (const Matrix &b)const{
        Matrix C;
        C.r=r,C.c=b.c;
        _for(i,0,C.r)_for(j,0,C.c){
            C.ele[i][j]=0;
            _for(k,0,c)
            C.ele[i][j]=(C.ele[i][j]+1LL*ele[i][k]*b.ele[k][j])%Mod;
        }
        return C;
    }
    bool operator != (const Matrix &b)const{
        _for(i,0,r)_for(j,0,c)if(ele[i][j]!=b.ele[i][j])
        return true;
        return false;
    }
}A[MAXS],I,X1;
Matrix quick_pow(int k){
    Matrix ans=I;
    int pos=0;
    while(k){
        if(k&1)ans=ans*A[pos];
        pos++;
        k>>=1;
    }
    return ans;
}
int a[MAXN],lef[MAXN<<2],rig[MAXN<<2];
Matrix s[MAXN<<2],lazy[MAXN<<2];
void push_up(int k){
    s[k]=s[k<<1]+s[k<<1|1];
}
void push_tag(int k,Matrix lazy_tag){
    s[k]=lazy_tag*s[k];
    lazy[k]=lazy_tag*lazy[k];
}
void push_down(int k){
    if(lazy[k]!=I){
        push_tag(k<<1,lazy[k]);
        push_tag(k<<1|1,lazy[k]);
        lazy[k]=I;
    }
}
```

```
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R,lazy[k]=I;
    int M=L+R>>1;
    if(L==R)
    return s[k]=quick_pow(a[M]-1)*X1,void();
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
    push_up(k);
}
void update(int k,int L,int R,int v){
    if(L<=lef[k]&&rig[k]<=R)
    return push_tag(k,quick_pow(v));
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L)
    update(k<<1,L,R,v);
    if(mid<R)
    update(k<<1|1,L,R,v);
    push_up(k);
}
Matrix query(int k,int L,int R){
    if(L<=lef[k]&&rig[k]<=R)
    return s[k];
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=R)
    return query(k<<1,L,R);
    else if(mid<L)
    return query(k<<1|1,L,R);
    else
    return query(k<<1,L,R)+query(k<<1|1,L,R);
}
int main()
{
    A[0]=Matrix(2,2);I=Matrix(2,2);X1=Matrix(2,1);
A[0].ele[0][1]=A[0].ele[1][0]=A[0].ele[1][1]=I.ele[0][0]=I.ele[1][1]=X1.ele[0][0]=X1.ele[1][0]=1;
    _for(i,1,MAXS)
    A[i]=A[i-1]*A[i-1];
    int n=read_int(),m=read_int();
    _rep(i,1,n)a[i]=read_int();
    build(1,1,n);
    while(m--){
        int tp=read_int(),l=read_int(),r=read_int();
        if(tp==1)update(1,l,r,read_int());
        else
        enter(query(1,l,r).ele[0][0]);
    }
    return 0;
```

```
}
```

3、Count The Rectangles

链接

题意

给定若干水平线和竖直线，问可以构成多少矩形(保证水平线之间两两不相交，竖直线之间两两不相交)

题解

考虑扫描线，枚举每一条水平线，然后考虑与该水平线相交的竖直线和纵坐标大于该线的水平线。

扫描一遍 y 轴，用树状树状维护此时与扫描线相交的竖直线的 x 坐标。然后扫描到竖直线上端点则删去该 x 坐标贡献。

如果扫描到水平线则查询该水平线与当前枚举的水平线的 x 轴公共区间上的竖直线个数 t，于是答案增加 $\frac {t(t-1)}2$。

注意到对 y 坐标相同的对象，应该先处理水平线然后处理竖直线。时间复杂度 $O(n^2\log v)$。

```
const int Base=5e4+5,MAXV=Base<<1,MAXN=5005;
#define lowbit(x) ((x)&(-x))
int c[MAXV];
void add(int pos,int v){
    while(pos<MAXV){
        c[pos]+=v;
        pos+=lowbit(pos);
    }
}
int query_pre(int pos){
    int s=0;
    while(pos){
        s+=c[pos];
        pos-=lowbit(pos);
    }
    return s;
}
int query(int lef,int rig){return query_pre(rig)-query_pre(lef-1);}
struct seg{int lef,rig,y;};
struct Node{
    int type,lef,rig,y;
    bool operator < (const Node &b)const{
        if(y!=b.y)return y<b.y;
        else
        return type<b.type;
```

```
    }
    Node(){}
    Node(int type,seg s){
        this->type=type;
        if(type==0)
        lef=s.lef,rig=s.rig,y=s.y;
        else
        lef=rig=s.y,y=s.rig;
    }
};
vector<seg> seg_r,seg_c;
vector<Node> node;
int main()
{
    int n=read_int();
    _for(i,0,n){
        int
x1=read_int()+Base,y1=read_int()+Base,x2=read_int()+Base,y2=read_int()+Base
;
        if(x1>x2)swap(x1,x2);
        if(y1>y2)swap(y1,y2);
        if(y1==y2)
        seg_r.push_back(seg{x1,x2,y1});
        else
        seg_c.push_back(seg{y1,y2,x1});
    }
    LL ans=0;
    _for(i,0,seg_r.size()){
        node.clear();
        int lef=seg_r[i].lef,rig=seg_r[i].rig,y=seg_r[i].y;
        _for(j,0,seg_r.size()){
            if(seg_r[j].y>y)
            node.push_back(Node(0,seg_r[j]));
        }
        _for(j,0,seg_c.size()){
            if(lef>seg_c[j].y||rig<seg_c[j].y)continue;
            if(seg_c[j].lef>y||seg_c[j].rig<y)continue;
            add(seg_c[j].y,1);
            node.push_back(Node(1,seg_c[j]));
        }
        sort(node.begin(),node.end());
        _for(j,0,node.size()){
            if(node[j].type==0){
                int
ql=min(max(node[j].lef,lef),rig),qr=min(max(node[j].rig,lef),rig);
                int t=query(ql,qr);
                ans+=1LL*t*(t-1)/2;
            }
            else
            add(node[j].lef,-1);
```

```
        }
    }
    enter(ans);
    return 0;
}
```

4、Median Sum

链接

题意

给定 n 个数，定义集合的权值为该集合中所有数的和。求这 n 个数的集合的所有子集的权值构成的集合中的中位数。注意这里所有集合指可重集。

题解

设所有数的和为 S，一个子集 s_1 的权值为 w，则一定有一个该子集的补集 s_2 的权值 $S-w$ 与之对应。

于是中位数一定是不小于 $\frac S2$ 的第一个数。用 vis 表示子集的可能值，对新加入的数 a，有 $vis_{k+a}=vis_{k+a}\text{ | }vis_a$。

考虑 bitset 暴力位压一下，时间复杂度 $O\left(\frac {nS}{w}\right)=O\left(\frac {n^2v}{w}\right)$。

```
const int MAXN=2005;
bitset<MAXN*MAXN> vis;
int main()
{
    int n=read_int(),sum=0;
    vis[0]=1;
    _for(i,0,n){
        int a=read_int();
        sum+=a;
        vis|=(vis<<a);
    }
    int pos=(sum+1)/2;
    while(!vis[pos])pos++;
    enter(pos);
    return 0;
}
```

5、GCD or MIN

链接

题意

给定 n 个数，每次可以任选两个数进行 gcd 或 $\min$ 操作得到一个新数再删去原来两个数。不断进行操作直到只剩下一个数。

问剩下的数一共有多少种可能的取值。

题解

首先不难发现剩下的数一定不大于 $\min (a)$，再通过观察不难发现答案等于所有不超过 $\min (a)$ 的仅通过 gcd 操作可以得到的数。

首先 $\min (a)$ 一定可以得到，下面仅考虑小于 $\min (a)$ 的数 v。

记 a 中所有满足 $v\mid a_i$ 的数为 $b_1,b_2\cdots b_k$，不难发现 v 可以得到当且仅当 $\text{gcd}(b_1,b_2\cdots b_k)=v$。

于是可以遍历 a_i，对每个 a_i，遍历他们的所有因子 d，同时更新 $g(d)$。(d 从 1 开始)

如果 $g(d)$ 不存在则将 $g(d)$ 设为 a_i，否则将 $g(d)$ 设为 $\text{gcd}(g(d),a_i)$。

最后答案为 $1+$ 所有满足 $g(d)==d$ 的 d 的个数。时间复杂度 $O(n\sqrt v\log v)$。

```
const int MAXN=2005,inf=1e9;
map<int,int> g;
int gcd(int a,int b){
    while(b){
        int t=b;
        b=a%b;
        a=t;
    }
    return a;
}
void update(int idx,int v){
    if(g.find(idx)!=g.end())
    g[idx]=gcd(g[idx],v);
    else
    g[idx]=v;
}
int a[MAXN];
int main()
{
    int n=read_int(),minv=inf;
    _for(i,0,n)a[i]=read_int(),minv=min(a[i],minv);
```

```
    _for(i,0,n){
        for(int j=1;j*j<=a[i]&&j<minv;j++){
            if(a[i]%j==0){
                update(j,a[i]);
                if(a[i]/j<minv)
                update(a[i]/j,a[i]);
            }
        }
    }
    int ans=1;
    for(map<int,int>::iterator it=g.begin();it!=g.end();++it){
        if(it->first==it->second)
        ans++;
    }
    enter(ans);
    return 0;
}
```

6□Minimum Difference

链接

题意

给定 n 个位置，每个位置一个照明范围为 p_i 的灯，如果该灯朝左，则照明区域为 $[i-p_i,i-1]$□如果该灯朝右，则照明区域为 $[i+1,i+p_i]$□

判定是否可以给每个灯一个朝向，使得 $[1,n]$ 区域全被照明，同时输出合法方案。

题解

设 $\text{dp}(i)$ 表示前 i 个灯的最大照明前缀。

设 $\text{pre}(i)$ 如果为 0 则表示第 i 个灯没有约束，如果不为 0 则表示第 i 个灯强制朝左且第 $[\text{pre}(i)+1,i-1]$ 灯贪心朝右。

如果第 i 个灯强制朝左，则考虑与之前的照明前缀拼接，于是需要找到 j 满足 $\text{dp}(j)+1\ge i-p_i$□如果存在 j□则 $\text{dp}(i)$ 至少为 $i-1$□

然后 $j+1\sim i-1$ 的所有灯可以贪心考虑全部朝右，于是可以选取 $\max_{j+1\le k\le i-1} (i+p_i)$ 更新 $\text{dp}(i)$□可以 ST 表维护。

另外如果 $\text{pre}(j)=0$□即 j 没有强制向左的约束，还可以用 $j+p_j$ 更新 $\text{dp}(i)$□

不难发现满足条件的 j 越小 $\text{dp}(i)$ 越大，于是权值线段树维护满足条件的最小下标即可。

如果第 i 个灯朝右，首先 $\text{dp}(i)$ 至少为 $\text{dp}(i-1)$□如果 $\text{dp}(i-1)\ge i$□则 $\text{dp}(i)\gets i+p_i$□

最后，如果强制朝左的收益大于朝右的收益，则强制朝左且 $\text{pre}(i)=j$否则贪心朝右且 $\text{pre}(i)=0$

输出方案所有灯贪心朝右，遇到 $\text{pre}(pos)\neq 0$ 则把该灯改成朝左且跳转到 $\text{pre}(pos)$ 即可。

```
const int MAXN=3e5+5,MAXM=20,inf=1e9;
namespace ST{
    int d[MAXN][MAXM],lg2[MAXN];
    void build(int *a,int n){
        lg2[1]=0;
        _rep(i,2,n)lg2[i]=lg2[i>>1]+1;
        _rep(i,1,n)d[i][0]=a[i];
        for(int j=1;(1<<j)<=n;j++){
            _rep(i,1,n-(1<<j)+1)
            d[i][j]=max(d[i][j-1],d[i+(1<<(j-1))][j-1]);
        }
    }
    int query(int lef,int rig){
        int len=rig-lef+1;
        return max(d[lef][lg2[len]],d[rig-(1<<lg2[len])+1][lg2[len]]);
    }
};
int p[MAXN],a[MAXN],dp[MAXN],pre[MAXN];
char ans[MAXN];
int lef[MAXN<<2],rig[MAXN<<2],s[MAXN<<2];
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R,s[k]=inf;
    if(L==R)return;
    int M=L+R>>1;
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
}
void update(int k,int pos,int v){
    s[k]=min(s[k],v);
    if(lef[k]==rig[k])return;
    int mid=lef[k]+rig[k]>>1;
    if(mid>=pos)update(k<<1,pos,v);
    else update(k<<1|1,pos,v);
}
int query(int k,int L,int R){
    if(L<=lef[k]&&rig[k]<=R)return s[k];
    int mid=lef[k]+rig[k]>>1;
    if(mid>=R)return query(k<<1,L,R);
    else if(mid<L)return query(k<<1|1,L,R);
    return min(query(k<<1,L,R),query(k<<1|1,L,R));
}
int main()
{
```

```
    int T=read_int();
    while(T--){
        int n=read_int();
        _rep(i,1,n)p[i]=read_int(),a[i]=min(p[i]+i,n);
        ST::build(a,n);
        build(1,0,n);
        _rep(i,1,n){
            int last=query(1,max(0,i-p[i]-1),n);
            if(last==inf){
                dp[i]=dp[i-1];
                pre[i]=0;
            }
            else{
                int vl=i-1,vr=dp[i-1];
                if(!pre[last])vl=max(vl,ST::query(last,i-1));
                else if(last+1<=i-1)vl=max(vl,ST::query(last+1,i-1));
                if(dp[i-1]>=i)vr=max(vr,a[i]);
                if(vl<=vr){
                    dp[i]=vr;
                    pre[i]=0;
                }
                else{
                    dp[i]=vl;
                    pre[i]=last;
                }
            }
            update(1,dp[i],i);
        }
        if(dp[n]<n)puts("NO");
        else{
            puts("YES");
            _rep(i,1,n)ans[i]='R';
            ans[n+1]='\0';
            int pos=n;
            while(pos){
                if(pre[pos])ans[pos]='L',pos=pre[pos];
                else
                pos--;
            }
            puts(ans+1);
        }
    }
    return 0;
}
```