Atcoder Rugular Contest 115

比赛链接

D - Odd Degree

题意

给定一个图，询问满足度数为奇数的点的个数恰好为 $k(1\sim n)$ 的生成图数量。

其中生成图的定义为原图的点集 $+$ 原图的边集的子集。

题解

首先考虑图是树的情况，有结论：如果 k 为奇数，显然生成图不存在；如果 k 为偶数，则生成图数量为 ${n\choose k}$。

下面用数学归纳发证明：对所有 $n\ge 1$，任意指定树上 $k(2\mid k)$ 个点作为度数为奇数的结点则满足条件的生成图唯一：

$n=1$ 时只有 $k=0$ 的情况，显然生成图唯一。

下面考虑从 $n+1$ 个点的树上任意指定 $k(2\mid k)$ 个点作为度数为奇数的结点。

任取一个度数为 1 的点，如果该点被指定为度数为奇数的点，则该点与树的连边一定保留。

考虑无视该点和该边，则等价于与该点相邻的另一个点在生成树中的奇偶性改变，于是有 $k'=k$ 或 $k'=k-2$。

如果该点被指定为度数为偶数的点，则该点与树的连边一定不保留，考虑无视该点和该边，于是有 $k'=k$。

然后等价于找 n 个点中有 k' 个点的生成图，根据数学归纳法，这是唯一的。

接下来考虑连通图的情况，假设图中有 n 个点和 m 条边，先确定一棵生成树。

对于非树边，有选于不选两种方案，同时非树边选与不选同时改变两个点的奇偶性。

当 k 为偶数时，任意指定 k 个点为度数为奇数的点，先考虑所有非树边选与不选对所有点奇偶性的影响，共 2^{m-n+1} 种情况。

最后等价于构造图有 k' 个点是度数为奇数的点的生成树的子图，此时方案数唯一。于是总方案数为 $2^{m-n+1}{n\choose k}$。

对于不连通图，考虑分治 NTT 处理背包的合并，时间复杂度 $O(n\log^2 n)$。

```
const int Mod=998244353,MAXN=5005;
int quick_pow(int a,int k){
```

```
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*a%Mod;
        a=1LL*a*a%Mod;
        k>>=1;
    }
    return ans;
}
namespace Poly{
    const int G=3;
    int rev[MAXN<<2];
    int build(int k){
        int n,pos=0;
        while((1<<pos)<=k)pos++;
        n=1<<pos;
        _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
        return n;
    }
    void NTT(int *f,int n,bool type){
        _for(i,0,n)if(i<rev[i])
        swap(f[i],f[rev[i]]);
        int t1,t2;
        for(int i=1;i<n;i<<=1){
            int w=quick_pow(G,(Mod-1)/(i<<1));
            for(int j=0;j<n;j+=(i<<1)){
                int cur=1;
                _for(k,j,j+i){
                    t1=f[k],t2=1LL*cur*f[k+i]%Mod;
                    f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                    cur=1LL*cur*w%Mod;
                }
            }
        }
        if(!type){
            reverse(f+1,f+n);
            int div=quick_pow(n,Mod-2);
            _for(i,0,n)f[i]=(1LL*f[i]*div%Mod+Mod)%Mod;
        }
    }
    void Mul(int *f,int _n,int *g,int _m){
        static int temp1[MAXN<<2],temp2[MAXN<<2];
        int n=build(_n+_m-2);
        _for(i,0,_n)temp1[i]=f[i];
        _for(i,_n,n)temp1[i]=0;
        _for(i,0,_m)temp2[i]=g[i];
        _for(i,_m,n)temp2[i]=0;
        NTT(temp1,n,true);NTT(temp2,n,true);
        _for(i,0,n)temp1[i]=1LL*temp1[i]*temp2[i]%Mod;
        NTT(temp1,n,false);
        _for(i,0,_n+_m-1)f[i]=temp1[i];
```

```
    }
}
int frac[MAXN],inv[MAXN],pow2[MAXN];
int C(int n,int m){
    return 1LL*frac[n]*inv[m]%Mod*inv[n-m]%Mod;
}
vector<int> g[MAXN];
int blk_id[MAXN],blk_cnt;
int pool[MAXN<<1],blk_sz[MAXN],blk_edge[MAXN],*blk_s[MAXN];
void dfs(int u){
    blk_id[u]=blk_cnt;
    _for(i,0,g[u].size()){
        int v=g[u][i];
        if(blk_id[v])continue;
        dfs(v);
    }
}
void solve(int lef,int rig){
    if(lef==rig)return;
    int mid=lef+rig>>1;
    solve(lef,mid);
    solve(mid+1,rig);
    int len1=1,len2=1;
    _rep(i,lef,mid)len1+=blk_sz[i];
    _rep(i,mid+1,rig)len2+=blk_sz[i];
    Poly::Mul(blk_s[lef],len1,blk_s[mid+1],len2);
}
int main()
{
    frac[0]=pow2[0]=1;
    _for(i,1,MAXN)frac[i]=1LL*i*frac[i-1]%Mod,pow2[i]=2LL*pow2[i-1]%Mod;
    inv[MAXN-1]=quick_pow(frac[MAXN-1],Mod-2);
    for(int i=MAXN-1;i;i--)inv[i-1]=1LL*inv[i]*i%Mod;
    int n=read_int(),m=read_int();
    _for(i,0,m){
        int u=read_int(),v=read_int();
        g[u].push_back(v);
        g[v].push_back(u);
    }
    _rep(i,1,n){
        if(!blk_id[i]){
            blk_cnt++;
            dfs(i);
        }
        blk_sz[blk_id[i]]++;
        blk_edge[blk_id[i]]+=g[i].size();
    }
    int *pos=pool;
    _rep(i,1,blk_cnt){
        blk_s[i]=pos;
        pos+=blk_sz[i]+1;
```

```
        int base=pow2[blk_edge[i]/2-blk_sz[i]+1];
        _rep(j,0,blk_sz[i]){
            if(j%2==0)
            blk_s[i][j]=1LL*C(blk_sz[i],j)*base%Mod;
            else
            blk_s[i][j]=0;
        }
    }
    solve(1,blk_cnt);
    _rep(i,0,n)
    enter(blk_s[1][i]);
    return 0;
}
```

E - LEQ and NEQ

题意

给定序列 A□求所有满足下列条件的序列 X 数目：

1. $1\le X_i\le A_i$
2. $X_i \neq X_{i+1}$

题解

设 $\text{dp}(i,j)$ 表示满足限制条件且 $X_i=j$ 的序列 $X_{1\sim i}$ 的数目□$S_i=\sum_{j=1}^{A_i}\text{dp}(i,j)$□

于是对于 $j\le \min(A_i,A_{i+1})$□有 $\text{dp}(i+1,j)=S_i-\text{dp}(i,j)$□

如果 $A_i\lt A_{i+1}$□对于 $A_i\lt j\le A_{i+1}$□有 $\text{dp}(i+1,j)=S_i$□

于是维护一棵支持区间取负，区间加，区间赋值的线段树即可，时间复杂度 $O(n\log n)$□

```
const int MAXN=5e5+5,Mod=998244353;
int
a[MAXN],b[MAXN],lef[MAXN<<2],rig[MAXN<<2],s[MAXN<<2],addTag[MAXN<<2],signTa
g[MAXN<<2],setTag[MAXN<<2];
bool setFlag[MAXN<<2];
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R,signTag[k]=1;
    if(L==R)
    return;
    int M=L+R>>1;
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
```

```
}
int getLen(int k){
    return b[rig[k]]-b[lef[k]-1];
}
void toSet(int k,int v){
    s[k]=1LL*getLen(k)*v%Mod;
    addTag[k]=0;
    signTag[k]=1;
    setTag[k]=v;
    setFlag[k]=true;
}
void toNeg(int k){
    s[k]=(Mod-s[k]);
    if(setFlag[k])
    setTag[k]=(Mod-setTag[k])%Mod;
    else{
        signTag[k]*=-1;
        addTag[k]=(Mod-addTag[k])%Mod;
    }
}
void toAdd(int k,int v){
    s[k]=(s[k]+1LL*getLen(k)*v)%Mod;
    if(setFlag[k])
    setTag[k]=(setTag[k]+v)%Mod;
    else
    addTag[k]=(addTag[k]+v)%Mod;
}
void push_down(int k){
    if(setFlag[k]){
        toSet(k<<1,setTag[k]);
        toSet(k<<1|1,setTag[k]);
        setFlag[k]=false;
    }
    else{
        if(signTag[k]==-1){
            toNeg(k<<1);
            toNeg(k<<1|1);
            signTag[k]=1;
        }
        if(addTag[k]){
            toAdd(k<<1,addTag[k]);
            toAdd(k<<1|1,addTag[k]);
            addTag[k]=0;
        }
    }
}
void push_up(int k){
    s[k]=(s[k<<1]+s[k<<1|1])%Mod;
}
void update1(int k,int L,int R,int v){
    if(L<=lef[k]&&rig[k]<=R){
```

```
        toSet(k,v);
        return;
    }
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L)
    update1(k<<1,L,R,v);
    if(mid<R)
    update1(k<<1|1,L,R,v);
    push_up(k);
}
void update2(int k,int L,int R,int v){
    if(L<=lef[k]&&rig[k]<=R){
        toNeg(k);
        toAdd(k,v);
        return;
    }
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L)
    update2(k<<1,L,R,v);
    if(mid<R)
    update2(k<<1|1,L,R,v);
    push_up(k);
}
int 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))%Mod;
}
int main()
{
    int n=read_int();
    _for(i,0,n)a[i]=b[i+1]=read_int();
    sort(b,b+n+1);
    int m=unique(b,b+n+1)-b;
    _for(i,0,n)a[i]=lower_bound(b,b+m,a[i])-b;
    build(1,1,m-1);
    update1(1,1,a[0],1);
    _for(i,1,n){
        int v=query(1,1,a[i-1]);
        if(a[i]<=a[i-1])
        update2(1,1,a[i],v);
```

```
        else{
            update1(1,a[i-1]+1,a[i],v);
            update2(1,1,a[i-1],v);
        }
    }
    enter(query(1,1,a[n-1]));
    return 0;
}
```