

<https://ac.nowcoder.com/acm/contest/7615/D>

给定一个排列，求有多少个上升序列 $\{a_{p_1}, a_{p_2} \dots a_{p_m}\}$, $s.t. \nexists j < p_1 \& a_j < a_{p_1}$ 且 $\nexists p_i < j < p_{i+1} \& a_{p_i} < a_j < a_{p_{i+1}}$ 且 $\nexists p_m < j \& a_{p_m} < a_j$

部分分做法是令 $dp[i]$ 表示以 a_i 为结尾的满足前两个条件的上升序列个数，令 $amax[j]$ 表示 j 到 i 之间 $> a_j$ 的最小的数，则 $dp[i] = [a_i == \min_{j=1}^i \{a_j\}] + \sum_{j < i \& a_j < a_i \& a_i < limit[j]} dp[j]$ ，要想满足最后一个条件，只需要倒着扫一遍看 a_i 是不是后缀最大值，复杂度 $O(n^2)$

Q:请给出有效转移次数最多的构造

于是我们可以hack一大票包括最优解在内的代码子呢

考虑每个数作为结尾时，只关注比它小的数，所以我们从小到大加入这些数，重新定义 $dp[i]$ 表示以数字 i 为结尾的满足前两个条件的上升序列个数， $pos[i]$ 表示数字 i 所在位置；如果 $j < i$ 且 $a_j > amax[j+1, i-1]$ ，那么 $dp[j]$ 就对 $dp[i]$ 有贡献；问题转化为快速算出一个区间的 $a_i, a_{i+1}, a_{i+2} \dots$ 跨过其右边某一定值 $limit$ 的贡献和

灵性建线段树，再灵性记录 $lans[rt]$ 表示以 rt 为根的左子树跨过右子树的最大值能贡献的值

如果右子树最大值 $< limit$ ，即无论如何也不可能贡献，则递归处理左子树；否则左子树一定要跨过右子树最大值才能贡献更右边的，即左子树的贡献就是 $lans[rt]$ ，递归处理右子树的贡献；线段树查询区间 $[1, pos[i]]$ 时先做右区间再做左区间可以维护当前区间要跨过的 $limit$ 值

单次询问区间跨过定值的贡献和复杂度为 $O(\log n)$ ，而求 $[1, pos[i]]$ 区间会拆成 $\log n$ 个区间，所以计算单个 $f[i]$ 的复杂度是 $O(\log^2 n)$ 的； $lans[rt]$ 的维护也是一个区间跨过定值的子问题，单个复杂度 $O(\log n)$ ，单次更新 $pos[i]$ 位置 $f[i]$ 的值会有 $\log n$ 个区间的 $lans$ 被更改，所以单次更新的复杂度是 $O(\log^2)$ 的；总时间复杂度 $O(n \log^2 n)$

```
#include <iostream>
#include <cstdio>
#define int long long
using namespace std;
const int MAX=2e5+5;
const int mod=998244353;
int n,ans;
int a[MAX],pos[MAX],f[MAX];
int amax[MAX<<2],lans[MAX<<2];    //lans表示左子树在>有子树最大值时的贡献和
bool leaf[MAX<<2];
void build(int rt,int l,int r){
    if(l==r){leaf[rt]=true;return ;}
```

```

    int mid=(l+r)>>1;
    build(rt<<1,l,mid);
    build(rt<<1|1,mid+1,r);
}
int getans(int rt,int limit){
    if(leaf[rt])return amax[rt]>limit?f[amax[rt]]:0;
    return amax[rt<<1|1]<limit?getans(rt<<1,limit):
(lans[rt]+getans(rt<<1|1,limit))%mod;
}
inline void pushup(int rt){
    amax[rt]=max(amax[rt<<1],amax[rt<<1|1]);
    lans[rt]=getans(rt<<1,amax[rt<<1|1]);
}
void modify(int rt,int l,int r,int aim,int c){
    if(l==r){
        amax[rt]=c;
        return ;
    }
    int mid=(l+r)>>1;
    if(aim<=mid)modify(rt<<1,l,mid,aim,c);
    else modify(rt<<1|1,mid+1,r,aim,c);
    pushup(rt);
}
int recmax;
int query(int rt,int l,int r,int ql,int qr){
    if(ql<=l && r<=qr){
        int v=recmax;
        recmax=max(recmax,amax[rt]);
        return getans(rt,v);
    }
    int mid=(l+r)>>1,ret=0;
    if(qr>mid)ret=query(rt<<1|1,mid+1,r,ql,qr);    //先做大数
    if(ql<=mid)ret=(ret+query(rt<<1,l,mid,ql,qr))%mod;
    return ret;
}
int tree[MAX];
inline int lowbit(int x){return x&(-x);}
inline void add(int x,int y){for(;x<=n;x+=lowbit(x))tree[x]+=y;}
inline int sum(int x){
    int ret=0;
    for(;x;x-=lowbit(x))ret+=tree[x];
    return ret;
}

```

```

signed main(){
    scanf("%lld",&n);
    for(int i=1;i<=n;i++)scanf("%lld",&a[i]),pos[a[i]]=i;
    build(1,1,n);
    for(int i=1;i<=n;i++){
        recmax=0;
        f[i]=max(1ll,query(1,1,n,1,pos[i]));
        modify(1,1,n,pos[i],i);
    }
    //for(int i=1;i<=n;i++)cout<<f[i]<<" ";
    //cout<<endl;
    for(int i=n;i;i--){
        if(sum(n)-sum(pos[i])==0)ans=(ans+f[i])%mod;
        add(pos[i],1);
    }
    printf("%lld",ans);
    return 0;
}

```