Atcoder Rugular Contest 126

比赛链接

D - Pure Straight

题意

给定长度为 n 的序列，交换任意两个相邻元素的费用为 1。

每个元素的取值范围为 $[1\sim k]$，问使得序列的一个连续子序列恰好为 $1,2\cdots k$ 的最小费用。

题解

首先假设固定所选元素的初始下标为 $p_1\lt p_2\lt \cdots \lt p_k$，且最终这 k 个元素的下标 $[x,x+k-1]$。

不妨记权值等于 i 的元素的位置为 b_i，同时设 $c_i=x+i-1$。

先证明该情况下的最小费用为 $\sum_{i=1}^k |p_i-c_i|+\text{inv}(b_1,b_2\cdots,b_k)$，其中 $\text{inv}B$ 表示序列 B 中的逆序对。

首先证明答案的下界为 $\sum_{i=1}^k |p_i-c_i|+\text{inv}(b_1,b_2\cdots,b_k)$。

定义 $f(P)=\sum_{i=1}^k |p_i-c_i|+\text{inv}(b_1,b_2\cdots,b_k)$ 为序列 P 的势能(一定要保证 $p_1\lt p_2\lt \cdots \lt p_k$)。

如果某次交换的两个元素都属于所选元素，则 P 不变，$\text{inv}B$ 至多减少 1。

如果某次交换的至多有一个元素都属于所选元素，则 $\sum_{i=1}^k |p_i-c_i|$ 至多减 1，$\text{inv}B$ 不变。

接下来构造可以取到下界的方案，其实只要先在不交换相对位置的情况下将所有元素移到 $[x,x+k-1]$ 然后再调整次序消除逆序对即可。

接下来考虑固定所选元素的初始下标为 $p_1\lt p_2\lt \cdots \lt p_k$，如何确认最优的最终下标 $[x,x+k-1]$。

由于此时 $\text{inv}(b_1,b_2\cdots,b_k)$ 是定值，所以只需要考虑 $\sum_{i=1}^k |p_i-c_i|$，记 $m=\lceil \frac k2\rceil$，不难发现 $x_m=c_m$ 时最优。此时有

$$ \sum_{i=1}^k |p_i-c_i|=\sum_{i=1}^m (c_i-p_i)+\sum_{i=m+1}^k (p_i-c_i)=\sum_{i=1}^m (p_m-p_i-(m-i))+\sum_{i=m+1}^k (p_i-p_m-(i-m)) $$

上式中 $i\lt m$ 的 p_i 系数为 1，$i\gt m$ 的 p_i 系数为 -1，而 p_m 系数根据 k 的奇偶性为 0 或 -1。

其他项是关于 m 的多项式，为定值。因此在 dp 过程中维护上式式子的结果和逆序对即可。

时间复杂度 $O\left(n2^k\right)$

```
const int MAXN=205,MAXK=17,inf=1e9;
int a[MAXN],dp[1<<MAXK],bt[MAXK][1<<MAXK];
int main()
{
    int n=read_int(),k=read_int(),s=1<<k;
    for(int i=k-1;i>=0;i--){
        _for(j,0,s){
            if(j&(1<<i))
            bt[i][j]=bt[i+1][j]+1;
            else
            bt[i][j]=bt[i+1][j];
        }
    }
    int m=(k+1)/2,a;
    _for(i,0,s)dp[i]=inf;
    dp[0]=0;
    _for(i,0,n){
        a=read_int()-1;
        _for(j,0,s){
            if(!(j&(1<<a))){
                int v=dp[j]+bt[a][j];
                if(bt[0][j]>=m)
                v+=i;
                else if(bt[0][j]+1==m){
                    if(k%2==0)
                    v-=i;
                }
                else
                v-=i;
                dp[j|(1<<a)]=min(dp[j|(1<<a)],v);
            }
        }
    }
    int ans=dp[s-1];
    _rep(i,1,k)
    ans-=abs(m-i);
    enter(ans);
    return 0;
}
```

E - Infinite Operations

题意

给定一个序列 A，定义一次操作为选定两个 $a_i,a_j(a_i\gt a_j)$，将两个数赋值为 a_i-x,a_j+x，其中 $0\lt x\le \frac {a_i-a_j}2$，并得到 x 分。

定义 $F(A)$ 表示序列 A 经过无限次操作后能得到的最大得分。接下来若干次单点修改操作，每次操作后求 $F(A)$。

题解

定义势能函数 $G(A)=\sum_{i=1}^n\sum_{j=1}^{i}|a_i-a_j|$，不难发现一次操作得 x 分至少使得 $G(A)$ 减少 $2x$。

同时如果假定 $a_1\le a_2\le\cdots a_n$，则选定相邻两个值操作恰好可以得 x 分同时使得 $G(A)$ 减少 x。

任意取两个相邻数取平均，易知经过无数次操作后有 $G(A)=0$，因此答案为 $\frac {G(A)}2$，树状数组维护答案即可，时间复杂度 $O(n\log n)$。

```
const int MAXN=3e5+5,MAXV=6e5+5,mod=998244353,inv2=(mod+1)/2;
int a[MAXN],b[MAXV];
pair<int,int> opt[MAXN];
LL c1[MAXV],c2[MAXV];
#define lowbit(x) ((x)&(-x))
void update(int pos,int v1,int v2){
    while(pos<MAXV){
        c1[pos]+=v1;
        c2[pos]+=v2;
        pos+=lowbit(pos);
    }
}
pair<LL,LL> query(int pos){
    pair<LL,LL> ans=make_pair(0LL,0LL);
    while(pos){
        ans.first+=c1[pos];
        ans.second+=c2[pos];
        pos-=lowbit(pos);
    }
    return ans;
}
int main()
{
    int n=read_int(),q=read_int();
    _rep(i,1,n)
    a[i]=b[i]=read_int();
    _rep(i,1,q){
        opt[i].first=read_int();
        opt[i].second=read_int();
        b[n+i]=opt[i].second;
    }
    sort(b+1,b+n+q+1);
```

```
    int m=unique(b+1,b+n+q+1)-b;
    LL ans=0,sum=0;
    _rep(i,1,n){
        int v=lower_bound(b+1,b+m,a[i])-b;
        update(v,b[v],1);
        sum+=b[v];
        pair<LL,LL> t=query(v);
        ans+=1LL*b[v]*t.second-t.first+sum-t.first-1LL*b[v]*(i-t.second);
        ans%=mod;
    }
    _rep(i,1,q){
        int p=opt[i].first;
        int v=lower_bound(b+1,b+m,a[p])-b;
        pair<LL,LL> t=query(v);
        ans-=1LL*b[v]*t.second-t.first+sum-t.first-1LL*b[v]*(n-t.second);
        ans%=mod;
        update(v,-b[v],-1);
        sum-=b[v];
        a[p]=opt[i].second;
        v=lower_bound(b+1,b+m,a[p])-b;
        update(v,b[v],1);
        sum+=b[v];
        t=query(v);
        ans+=1LL*b[v]*t.second-t.first+sum-t.first-1LL*b[v]*(n-t.second);
        ans%=mod;
        enter(1LL*(ans+mod)*inv2%mod);
    }
    return 0;
}
```