2020ICPC·小米 网络选拔赛第一场

比赛链接

E. Phone Network

题意

给出长度为 n 的序列 A，保证 $1\le a_i\le m$，对每个 $1\le i\le m$，询问保证 $1\sim i$ 的连续子序列的最小长度。

题解1

设 $R(i,l)$ 表示序列 A 中以 a_l 为左端点且包含 $1\sim i$ 的最短序列的右端点。

考虑 $R(i,1\sim n)$ 到 $R(i+1,1\sim n)$ 的转移。不妨设 $i+1$ 的位置分别为 $p_1,p_2\cdots p_k$，同时假设 $p_0=0$。

于是对 $p_j \lt l\le p_{j+1}$，有 $R(i+1,l)=\max\left(R(i,l),p_{j+1}\right)$，特别的对 $p_k \lt l\le n$，有 $R(i+1,l)=\inf$。

于是该题转化为需要维护一个支持区间 max 操作且能维护答案 $R(i,l)-l+1$ 的最小值的数据结构。但区间 max 操作难以维护答案。

不难发现 $R(i,1\sim n)$ 单调递增，于是对区间 $(p_j,p_{j+1}]$ 作 $\max$ 操作等价于找到区间左半 R 值不超过 p_{j+1} 的部分进行区间 set 操作。

最终可以 $O(n\log n)$ 维护答案。

```
const int MAXN=2e5+5,Inf=1e9;
int
lef[MAXN<<2],rig[MAXN<<2],maxv[MAXN<<2],minv[MAXN<<2],lazy[MAXN<<2],s[MAXN<
<2];
void push_up(int k){
    maxv[k]=max(maxv[k<<1],maxv[k<<1|1]);
    minv[k]=min(minv[k<<1],minv[k<<1|1]);
    s[k]=min(s[k<<1],s[k<<1|1]);
}
void push_tag(int k,int v){
    lazy[k]=maxv[k]=minv[k]=v;
    s[k]=v-rig[k]+1;
}
void push_down(int k){
    if(lazy[k]){
        push_tag(k<<1,lazy[k]);
        push_tag(k<<1|1,lazy[k]);
```

```
        lazy[k]=0;
    }
}
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R,s[k]=Inf;
    int M=L+R>>1;
    if(L==R)
    return;
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
}
void update(int k,int L,int R,int v){
    if(minv[k]>=v)return;
    if(L<=lef[k]&&rig[k]<=R&&maxv[k]<=v){
        push_tag(k,v);
        return;
    }
    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);
}
vector<int> p[MAXN];
int main()
{
    int n=read_int(),m=read_int();
    _rep(i,1,m)p[i].push_back(0);
    _rep(i,1,n)p[read_int()].push_back(i);
    build(1,1,n);
    _rep(i,1,m){
        _for(j,1,p[i].size())
        update(1,p[i][j-1]+1,p[i][j],p[i][j]);
        if(*(--p[i].end())!=n)
        update(1,*(--p[i].end())+1,n,Inf);
        space(s[1]);
    }
    return 0;
}
```

题解2

考虑吉司机线段树，每个结点维护未最小值的答案和总答案即可。

由于吉司机线段树修改时只修改最小值，而最小值的答案更新为最小值权值与最小值最右端点的差。

于是答案变为非最小值答案和最小值权值与最小值最右端点的差的最小值。

```
const int MAXN=2e5+5,Inf1=1e9,Inf2=2e9;
int
lef[MAXN<<2],rig[MAXN<<2],minv[MAXN<<2],secv[MAXN<<2],minp[MAXN<<2],s1[MAXN
<<2],s2[MAXN<<2],lazy[MAXN<<2];
void push_up(int k){
    s1[k]=min(s1[k<<1],s1[k<<1|1]);
    if(minv[k<<1]==minv[k<<1|1]){
        minv[k]=minv[k<<1];
        minp[k]=minp[k<<1|1];
        secv[k]=min(secv[k<<1],secv[k<<1|1]);
        s2[k]=min(s2[k<<1],s2[k<<1|1]);
    }
    else if(minv[k<<1]<minv[k<<1|1]){
        minv[k]=minv[k<<1];
        minp[k]=minp[k<<1];
        secv[k]=min(secv[k<<1],minv[k<<1|1]);
        s2[k]=min(s2[k<<1],s1[k<<1|1]);
    }
    else{
        minv[k]=minv[k<<1|1];
        minp[k]=minp[k<<1|1];
        secv[k]=min(minv[k<<1],secv[k<<1|1]);
        s2[k]=min(s1[k<<1],s2[k<<1|1]);
    }
}
void push_tag(int k,int v){
    if(v<=minv[k])return;
    s1[k]=min(s2[k],v-minp[k]+1);
    minv[k]=lazy[k]=v;
}
void push_down(int k){
    if(lazy[k]){
        push_tag(k<<1,lazy[k]);
        push_tag(k<<1|1,lazy[k]);
        lazy[k]=0;
    }
}
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R,lazy[k]=0;
    int M=L+R>>1;
    if(L==R){
        minv[k]=0,minp[k]=M;
        s1[k]=s2[k]=secv[k]=Inf2;
        return;
    }
    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(minv[k]>=v)return;
```

```
    if(L<=lef[k]&&rig[k]<=R&&secv[k]>v)
    return push_tag(k,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);
}
vector<int> p[MAXN];
int main()
{
    int n=read_int(),m=read_int();
    _rep(i,1,m)p[i].push_back(0);
    _rep(i,1,n)p[read_int()].push_back(i);
    build(1,1,n);
    _rep(i,1,m){
        _for(j,1,p[i].size())
        update(1,p[i][j-1]+1,p[i][j],p[i][j]);
        if(*(--p[i].end())!=n)
        update(1,*(--p[i].end())+1,n,Inf1);
        space(s1[1]);
    }
    return 0;
}
```

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:xiaomi_icpc_1

Last update: **2020/10/30 09:14**