

2019 Multi-University Training Contest 2

[比赛链接](#)

B. Beauty Of Unimodal Sequence

题意

给定一个序列，求最长的单峰序列中字典序最小和最大的序列。

其中单峰序列定义为序列 $p_1, p_2 \dots p_t$ 满足约束 $p_1 < p_2 < \dots < p_t$ 且 $a_{p_1} < a_{p_2} < \dots < a_{p_k} > \dots > a_{p_t}$

题解

首先不难求出 $f(0, i)$ 表示以 a_i 结尾的最长单增序列， $f(1, i)$ 表示以 a_i 开头的最长单减序列。定义 $g(i)$ 表示以 a_i 开头的最长单峰序列。

于是有状态转移

$$g(i) = \max\left(f(1, i), \max_{j > i, a_j > a_i} g(j) + 1\right)$$

不难用线段树维护求解 $g(i)$ ，接下来考虑贪心逐位求最小和最大字典序单峰序列。

最小字典序只需要从前往后考虑找到第一个满足相应条件的位置。

考虑当前是否已经强制降序，通过 $f(1, i), g(i)$ 判断选取当前位置的数后是否可以构造最长单峰序列即可，时间复杂度 $O(n)$

最大字典序需要从后往前考虑找到第一个满足相应条件的位置，但需要保证每个数只被扫到 $O(1)$ 次，用桶维护即可。

总时间复杂度 $O(n \log n)$

```
const int MAXN=3e5+5;
int a[MAXN], b[MAXN], f[MAXN], dp[2][MAXN], g[MAXN], ans[MAXN];
void solve(int *a, int *dp, int n){
    int m=0;
    _for(i, 0, n){
        int pos=lower_bound(f, f+m, a[i]) - f;
        dp[i]=pos;
        f[pos]=a[i];
        if(m==pos)m++;
    }
}
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]=0;
    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){
    if(lef[k]==rig[k]){
        s[k]=max(s[k], v);
        return;
    }
    int mid=lef[k]+rig[k]>>1;
    if(pos<=mid)
        update(k<<1, pos, v);
    else
        update(k<<1|1, pos, v);
    s[k]=max(s[k<<1], s[k<<1|1]);
}
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);
    else
        return max(query(k<<1, L, R), query(k<<1|1, L, R));
}
vector<int> c1[MAXN], c2[MAXN];
int main()
{
    int n;
    while(~scanf("%d", &n)){
        _for(i, 0, n) a[i]=b[i]=read_int();
        sort(b, b+n);
        int m=unique(b, b+n)-b;
        _for(i, 0, n)
            a[i]=lower_bound(b, b+m, a[i])-b;
        solve(a, dp[0], n);
        reverse(a, a+n);
        solve(a, dp[1], n);
        reverse(a, a+n);
        reverse(dp[1], dp[1]+n);
        build(1, 0, m);
        int maxs=0;
        for(int i=n-1; i>=0; i--){
            g[i]=max(query(1, a[i]+1, m), dp[1][i])+1;
            update(1, a[i], g[i]);
            maxs=max(maxs, g[i]+dp[0][i]);
        }
    }
}
```

```

}
int pos=0,flag=1,curv=-1;
_for(i,0,n){
    if(flag){
        if(a[i]>curv&&pos+g[i]==maxs){
            ans[pos++]=i;
            curv=a[i];
        }
        else if(a[i]<curv&&pos+dp[1][i]+1==maxs){
            ans[pos++]=i;
            curv=a[i];
            flag=0;
        }
    }
    else{
        if(a[i]<curv&&pos+dp[1][i]+1==maxs){
            ans[pos++]=i;
            curv=a[i];
        }
    }
}
_for(i,0,maxs)
printf("%d%c",ans[i]+1,(i==maxs-1)?'\n':' ');
_for(i,0,maxs)c1[i].clear(),c2[i].clear();
for(int i=n-1;i>=0;i--){
    c1[maxs-g[i]].push_back(i);
    c2[maxs-dp[1][i]-1].push_back(i);
}
pos=0,flag=1,curv=-1;
_for(i,0,maxs){
    if(flag){
        int pos1=0,pos2=0;
        while(pos1<=c1[i].size()&&pos2<=c2[i].size()){
if(pos2==c2[i].size()||(pos1!=c1[i].size()&&c1[i][pos1]>=c2[i][pos2])){
            int p=c1[i][pos1++];
            if(a[p]>curv){
                ans[i]=p;
                curv=a[p];
                break;
            }
        }
        else{
            int p=c2[i][pos2++];
            if(a[p]<curv){
                ans[i]=p;
                curv=a[p];
                flag=0;
                break;
            }
        }
    }
}
}
}

```

```
    }
    else{
        _for(j,0,c2[i].size()){
            int p=c2[i][j];
            if(a[p]<curv){
                ans[i]=p;
                curv=a[p];
                break;
            }
        }
    }
    _for(i,0,maxs)
    printf("%d%c",ans[i]+1,(i==maxs-1)?'\n':' ');
}
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:2021_buaa_spring_training9&rev=1624195921

Last update: 2021/06/20 21:32