# BUAA ICPC 2020-2021

比赛网址

## 训练结果

- 时间:2020.7.23 12:00~17:00
- rank:7/17
- 完成情况：5/6/11

## 题解

## F.Empty Vessels

#### 题意

#### 题解

```

```

## G.Maximum Product

#### 题意

#### 题解

```

```

## H.Biathlon 2.0

#### 题意

#### 题解

```

```

# I.Archaeological Research

### 题意

构造一个长度为$n$的数字序列(从数字1开始)，使得满足一定的条件下字典序最小。

条件：对于每个位置$i$给定之后的一些位置$p_j$该位置$p_j$上的数字是$i$位置之后第一次出现

### 题解

要字典序最小，当然是使用当前能使用最小的数字

每次走到一个位置，如果这个位置没有被之前任意一个位置限制，那么就放$1$

否则，只需要查找最早限制的位置到这个位置之间最小的没被用过的数字

这个可以用可持久化线段树维护

建立一个权值线段树，记录每个权值最后出现的位置，维护区间最小值，对于区间$[l,r]$只需要在$r$位置线段树中通过左右区间最小值判断区间内是否有合法权值，然后尽量往左走即可。

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 300005,maxm = 10000005,INF = 0x3f3f3f3f;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
int n,pre[maxn],ans[maxn];
```

```
int mn[maxm],ls[maxm],rs[maxm],rt[maxn],cnt,L,R;
void modify(int& u,int v,int l,int r,int V){
    if (!u) {u = ++cnt; ls[u] = ls[v]; rs[u] = rs[v]; mn[u] = mn[v];}
    if (l == r) {mn[u] = V; return;}
    int mid = (l + r) >> 1;
    if (mid >= L) modify(ls[u],ls[v],l,mid,V);
    else modify(rs[u],rs[v],mid + 1,r,V);
    mn[u] = min(mn[ls[u]],mn[rs[u]]);
}
int query(int u,int l,int r){
    if (l == r) return l;
    int mid = (l + r) >> 1;
    if (mn[ls[u]] < L) return query(ls[u],l,mid);
    return query(rs[u],mid + 1,r);
}
int main(){
    n = read();
    for (int i = 1; i <= n; i++){
        if (!pre[i] || pre[i] == i - 1) ans[i] = 1;
        else {L = pre[i] + 1; ans[i] = query(rt[i - 1],1,n);}
        L = ans[i];
        modify(rt[i],rt[i - 1],1,n,i);
        int m = read(),u;
        for (int j = 1; j <= m; j++){
            u = read();
            if (!pre[u]) pre[u] = i;
        }
    }
    printf("%d",ans[1]);
    for (int i = 2; i <= n; i++) printf(" %d",ans[i]); puts("");
    return 0;
}
```

# J.Sockets

### 题意

有$n$个插排$m$个设备和一个插口，每个插排有各自的插口个数$a_i$每个设备有各自的性能$b_j$$b_j$表示该设备正常运行最多能经过的插排个数，求最多能使多少设备正常运行

### 题解

容易发现，插排一定是越大越要先用，设备一定是$b_j$越大越先用

那么就二分使用的设备个数，贪心地分层插入插排。如果当前层有必须使用的设备，就使用之，否则尽量插插排。最后看能否合法插入完。

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 200005,maxm = 100005,INF = 0x3f3f3f3f;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
int n,m;
int A[maxn],B[maxn];
bool cmp(int a,int b){return a > b;}
bool check(int M){
    //cout << M << endl;
    LL left = 1; int pos = 1,dep = 0;
    while (true){
        while (M && left && B[M] == dep) M--,left--;
        if (!left && M) return false;
        LL tmp = 0;
        while (pos <= n && left) left--,tmp += A[pos++];
        while (M && left) M--,left--;
        if (!M) return true;
        left += tmp;
        dep++;
        if (pos > n) break;
    }
    while (M && left && B[M] >= dep) left--,M--;
    return M == 0;
}
int main(){
    n = read(); m = read();
    REP(i,n) A[i] = read();
    REP(j,m) B[j] = read();
```

```
    sort(A + 1,A + 1 + n,cmp);
    sort(B + 1,B + 1 + m,cmp);
    int l = 1,r = m,mid;
    while (l < r){
        mid = (l + r + 1) >> 1;
        if (check(mid)) l = mid;
        else r = mid - 1;
    }
    printf("%d\n",l);
    return 0;
}
```

# 训练实况

第五场：

开局hxm看A，wxg看G，wxg开写G

fyh和hxm讨论A无果，fyh继续想A，hxm想J

12:38 wxg过G

hxm和wxg讨论出J做法

13:07hxm过J

fyh wxg想H，fyh开写H，中间目测调题，wxg尝试乱搞A

14:28 fyh过H，hxm想I，想出做法

hxm用错数据结构，和wxg讨论后修改

wxg wa A 放弃A

16:06 hxm过I，wxg想出F，fyh尝试D

16:40 wxg过F

# 训练总结