Codeforces Round #699 (Div. 2)

比赛链接

E. Sorting Books

题意

给定一个长度为 n 的序列 $\{a\}$，每次操作可以任取一个位置并将该位置的数移到序列末尾。

问最少需要多少次操作才能使序列中所有大小相同的数相邻。

题解

显然只需要在序列中选定一些位置，然后按特定顺序将他们移除就可以完成任务。定义 $\text{dp}(i)$ 表示不选择序列 $[i,n]$ 中的位置个数的最大值。

维护每个值 v 的最靠左出现的位置 l_v，最靠右出现的位置 r_v，动态维护 $[i,n]$ 中出现的次数 c_v。

如果保留位置 i，当 $i=l_{a_i}$ 时，需要移除 $[i,r_{a_i}]$ 中所有不等于 a_i 的位置，即 $\text{dp}(i)\gets c_{a_i}+\text{dp}(r_{a_i}+1)$。

当 $i\neq l_{a_i}$ 时，若不保留 l_{a_i} 的位置，则 l_{a_i} 的位置的数会被加到位置 n 后面。

为了使得 l_{a_i} 的位置的数与 i 位置的数所在段相邻，需要移除 $[i,n]$ 中所有不等于 a_i 的位置，于是 $\text{dp}(i)\gets c_{a_i}$。

若保留 l_{a_i} 的位置，则该情况会在计算 $\text{dp}(l_{a_i})$ 时考虑，此时不考虑。

如果不保留位置 i，则 $\text{dp}(i)\gets \text{dp}(i+1)$，最终答案为 $n-\text{dp}(1)$，时间复杂度 $O(n)$。

```
const int MAXN=5e5+5;
int a[MAXN],dp[MAXN],l[MAXN],r[MAXN],c[MAXN];
int main()
{
    int n=read_int();
    _rep(i,1,n)a[i]=read_int();
    for(int i=n;i;i--){
        l[a[i]]=i;
        if(!r[a[i]])r[a[i]]=i;
    }
    for(int i=n;i;i--){
        c[a[i]]++;
        dp[i]=dp[i+1];
```

```
            if(i==l[a[i]])dp[i]=max(dp[i],dp[r[a[i]]+1]+c[a[i]]);
            else dp[i]=max(dp[i],c[a[i]]);
        }
        enter(n-dp[1]);
        return 0;
}
```

F. AB Tree

题意

给定一棵以 1 为根节点的有根树以及 x 个字符 a 和 $n-x$ 个字符 b，要求用所给字符对树上每个结点进行标注。

给定每个节点代表一个字符串，该字符串由根节点到该节点路径上的所有字符依次拼接而成。

要求最小化所有节点代表的字符串中的互异的字符串个数，同时输出任意一种方案。

题解

设树的深度为 d，根节点深度为 0。于是答案一定不小于 $d+1$，因为一定有长度为 $1\sim d+1$ 的字符串。

先考虑如何判定答案是否为 $d+1$ 以及如何构造方案。为了满足条件，必须满足每个长度的字符串只有一种，于是同一深度的节点字符一定相同。

不妨设深度 i 共有 c_i 个节点，于是只需要从 $c_0,c_1\cdots c_d$ 中选出若干个数构成 x 即可。

这是一个背包 dp，考虑优化方案，首先由于 $\sum_{i=0}^d c_i=n$，所以 c_i 最多只有 $O(\sqrt n)$ 个互异的值。

考虑将所有值相同的 c_i 放在一起考虑，本题转化为多重背包问题，且物品权值只有 $O(\sqrt n)$ 种。

设 $\text{dp}(i,j)$ 表示只考虑前 i 种权值物品是否存在权值和为 j 的方案。

设当前物品权值为 val，数量为 cnt，于是对 $\text{dp}(i,j)$，只需要找到 $0\le k\le \text{cnt}$ 使得 $\text{dp}(i-1,j-k*\text{val})=1$ 即可。

不妨记 k_j 表示满足上述条件的最小的 k，且如果不存在 k 满足条件则 $k_j=-1$。

于是有若 $\text{dp}(i-1,j)=1$ 则 $k_j=0$，否则 $k_j=k_{j-\text{val}}+1$，注意 $k_{j-\text{val}}=-1$ 和 $k_{j-\text{val}}+1\gt \text{cnt}$ 的情况。

于是可以 $O(n)$ 维护每种权值，$O(n\sqrt n)$ 判定答案是否为 $d+1$。

如果答案等于 $d+1$，则逆序暴力并输出方案，时间复杂度 $O(n)$。

否则，答案一定为 $d+2$，下面给出构造。

首先假设当前层还剩下 x 个字符 a 和 y 个字符 b，当前节点数为 c_i。

若 c_i 满足 $c_i\le \max(x,y)$，则直接将该层节点用一种字符覆盖即可。

否则，考虑当前层的非叶子节点数。剩下的节点数总和为 $x+y$，而非叶子结点数总和一定不大于叶子结点数总和。

于是当前层的非叶子节点数 $\le$ 叶子结点数总和 $\le \frac {x+y}2\le \max(x,y)$。

不妨设 $\max(x,y)=x$，于是用全部 a 覆盖当前层所有非叶子节点与部分叶子节点，其他剩余节点全部用 b 覆盖，于是总答案仅 $+1$。

总时间复杂度 $O(n\sqrt n)$。

```
const int MAXN=1e5+5,MAXM=500;
struct Edge{
    int to,next;
}edge[MAXN];
int head[MAXN],edge_cnt;
void AddEdge(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
int sz[MAXN],last[MAXN],maxd;
vector<int> node[MAXN],c[MAXN],cv;
bool dp[MAXM][MAXN];
char ans[MAXN];
void dfs(int u,int d){
    node[d].push_back(u);
    sz[u]=1,maxd=max(maxd,d);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        dfs(v,d+1);
        sz[u]+=sz[v];
    }
}
bool cmp(int u,int v){
    return sz[u]>sz[v];
}
int main()
{
    int n=read_int(),x=read_int();
    _rep(i,2,n)
    AddEdge(read_int(),i);
    dfs(1,0);
    _for(i,0,n)
    c[node[i].size()].push_back(i);
    _rep(i,1,n){
        if(c[i].size()>0)
        cv.push_back(i);
    }
```

```
    dp[0][0]=true;
    _rep(i,1,cv.size()){
        int val=cv[i-1],cnt=c[val].size();
        mem(last,-1);
        _for(j,0,val){
            if(dp[i-1][j])
            dp[i][j]=true,last[j]=0;
            else
            dp[i][j]=false,last[j]=-1;
        }
        _rep(j,val,n){
            if(dp[i-1][j])
            dp[i][j]=true,last[j]=0;
            else if(last[j-val]!=-1&&last[j-val]+1<=cnt)
            dp[i][j]=true,last[j]=last[j-val]+1;
            else
            dp[i][j]=false,last[j]=-1;
        }
    }
    if(dp[cv.size()][x]){
        enter(maxd+1);
        _rep(i,1,n)ans[i]='b';
        for(int i=cv.size(),j=x;i;i--){
            int val=cv[i-1],pos=0;
            while(!dp[i-1][j]){
                j-=val;
                int d=c[val][pos++];
                _for(k,0,val)
                ans[node[d][k]]='a';
            }
        }
        puts(ans+1);
    }
    else{
        enter(maxd+2);
        pair<char,int> p1('a',x),p2('b',n-x);
        _for(i,0,n){
            if(p1.second<p2.second)swap(p1,p2);
            sort(node[i].begin(),node[i].end(),cmp);
            _for(j,0,node[i].size()){
                if(p1.second){
                    ans[node[i][j]]=p1.first;
                    p1.second--;
                }
                else{
                    ans[node[i][j]]=p2.first;
                    p2.second--;
                }
            }
        }
```

```
        puts(ans+1);
    }
    return 0;
}
```