Atcoder Rugular Contest 125

比赛链接

D - Unique Subsequence

题意

给定一个长度为 n 的序列 A，求该序列含有的独特的子序列个数(不包括空序列)。

其中，定义独特的子序列为序列 A 的所有子序列构成的可重集中出现次数等于 1 的子序列。

题解

设 $\text{dp}(i)$ 为以位置 i 结尾的独特子序列个数。

考虑从 $1\sim n$ 动态维护每个 $\text{dp}(i)$，假设当前位置为 i，求出最大的 j 满足 $j\lt i\And a_i=a_j$。

对于所有终止位置小于 j 的独特子序列，加上 a_i 显然不构成独特子序列，因为有 a_j,a_i 两种选择。

对于所有终止位置 $\in [j,i)$ 的独特子序列，加上 a_i 还是独特子序列，因为只有 a_i 一种选择。于是有 $\text{dp}(i)=\sum_{k=j}^{i-1} \text{dp}(k)$。

同时，加入 a_i 后所有前面以 a_i 结尾的独特子序列都不再独特，因此有 $(j\lt i\And a_i=a_j)\to (\text{dp}(j)=0)$。

考虑树状数组加速上述操作，时间复杂度 $O\left(n\log n\right)$，最后答案即为所有 dp 值相加。

```
const int MAXN=2e5+5,mod=998244353;
#define lowbit(x) ((x)&(-x))
int c[MAXN];
void add(int pos,int v){
    while(pos<MAXN){
        c[pos]=(c[pos]+v)%mod;
        pos+=lowbit(pos);
    }
}
int query(int pos){
    int ans=0;
    while(pos){
        ans=(ans+c[pos])%mod;
        pos-=lowbit(pos);
    }
    return ans;
```

```
}
int query(int l,int r){
    return (query(r)+mod-query(l-1))%mod;
}
int a[MAXN],lst[MAXN],pre[MAXN],dp[MAXN];
int main()
{
    int n=read_int();
    _rep(i,1,n){
        a[i]=read_int();
        pre[i]=lst[a[i]];
        lst[a[i]]=i;
    }
    int ans=0;
    _rep(i,1,n){
        if(pre[i]){
            dp[i]=query(pre[i],i-1);
            add(pre[i],mod-dp[pre[i]]);
            dp[pre[i]]=0;
        }
        else
        dp[i]=(query(1,i-1)+1)%mod;
        add(i,dp[i]);
    }
    _rep(i,1,n)
    ans=(ans+dp[i])%mod;
    enter(ans);
    return 0;
}
```

E - Snack

题意

给定 n 种物品，第 i 种物品有 a_i 个。给定 m 个背包，第 i 个背包大小为 c_i且同种物品最多只能放 b_i 个。

问 m 个背包最多能装下的物品数。

题解

首先考虑网络流建图，将第 i 个物品作为点 L_i第 j 个背包作为点 R_j建边 $S\to L_i$容量为 $a_i$$L_i\to R_j$容量为 $b_j$$R_i\to T$容量为 c_i

于是答案为最大流，同时也等于最小割，考虑快速最小割计算。

假设已经确定保留 k 条 $S\to L_i$ 的连边。那么对每个 R_i显然他的割是 $\min(k\times b_i,c_i)$

且 R_i,R_j 的选项互不影响。

于是可以将所有 L_i 按 a_i 排序，所有 R_i 按 $\frac {c_i}{b_i}$ 排序。枚举 $k\in [0,n]$，用一个指针维护哪些 R_i 贡献为 $k\times b_i$，哪些 R_i 贡献为 c_i。

然后前缀和统计贡献，时间复杂度 $O\left(n\log n+m\log m\right)$。

```
const int MAXN=2e5+5;
const LL inf=1e18;
LL a[MAXN],preb[MAXN],prec[MAXN];
struct Node{
    unsigned long long b,c;
    bool operator < (const Node &t)const{
        return c*t.b>t.c*b;
    }
}node[MAXN];
int main()
{
    int n=read_int(),m=read_int();
    LL s=0,ans=inf;
    _for(i,0,n)
    a[i]=read_LL();
    sort(a,a+n);
    _rep(i,1,m)
    node[i].b=read_LL();
    _rep(i,1,m)
    node[i].c=read_LL();
    sort(node+1,node+m+1);
    _rep(i,1,m){
        preb[i]=preb[i-1]+node[i].b;
        prec[i]=prec[i-1]+node[i].c;
    }
    int pos=1;
    _rep(i,0,n){
        while(pos<=m&&node[pos].b*(n-i)<node[pos].c)
        pos++;
        ans=min(ans,s+preb[pos-1]*(n-i)+prec[m]-prec[pos-1]);
        s+=a[i];
    }
    enter(ans);
    return 0;
}
```