

2022/11/15

T1 reverse

题意

- 给出 0/1 串的长度 n , 和能操作的最长区间 K , 和 m 个 a_i , 和 1 的位置 S (其余都为 0)
- 每次操作可以选择一个长度为 K 的连续子串并翻转 , 翻转后 1 不能在 a_i 上
- 求对于每个位置 , 需要多少次操作使 1 在这个位置上 (不能输出 -1)
- $1 \leq n \leq 10^5, 1 \leq S, K \leq n, 0 \leq m \leq n$

分析

- 首先 , 对于每一个位置 i , 他可以经过一次操作到达的位置的左右界 l_i, r_i 为
 - 如果 $i > K$, $l_i = i - K + 1$, 否则 $l_i = K - i + 1$
 - 如果 $i \leq n - K$, $r_i = i + K - 1$, 否则 $r_i = n \times 2 - K - i - 1$
- 于是就可以直接跑 bfs , 时间复杂度 $O(nK)$
- 考虑优化 , 发现一个点经过一次操作可以到达的位置都是奇数点或都是偶数点 , 而且可到达的点是一段连续的区间 , 于是可以用 `set` 优化 bfs 的过程 , 搜到一个点就将其弹出 , 时间复杂度 $O(n \log n)$

代码

```
#include<bits/stdc++.h>
using namespace std;

const int N=1e5+10;
int n,k,m,s,que[N],head,tail,ans[N];
bool cant[N];
set<int>q[2];
set<int>::iterator it;

void bfs()
{
    memset(ans,0xff,sizeof(ans));
    que[++tail]=s;
    ans[s]=0;
    q[s&1].erase(s);
    int now;
    while(head!=tail)
    {
        now=que[++head];
        int l=now>K?now-K+1:K-now+1,r=now<=n-K?now+K-1:n*2-K-now+1,id=
(now&1)^(~K&1);
        it=q[id].lower_bound(l);
        while(it!=q[id].end()&&*it<=r)
        {
            if(!cant[*it])
            {
                ans[*it]=ans[now]+1;
```

```

        que[++tail]=*it;
    }
    q[id].erase(it++);
}
}
}

int main()
{
    scanf("%d%d%d%d",&n,&k,&m,&s);
    int tmp;
    for(int i=1;i<=m;++i)
        scanf("%d",&tmp),cant[tmp]=1;
    for(int i=1;i<=n;++i)
        q[i&1].insert(i);
    bfs();
    for(int i=1;i<=n;++i)
        printf("%d ",ans[i]);
    return 0;
}

```

T2 Silhouette

题意

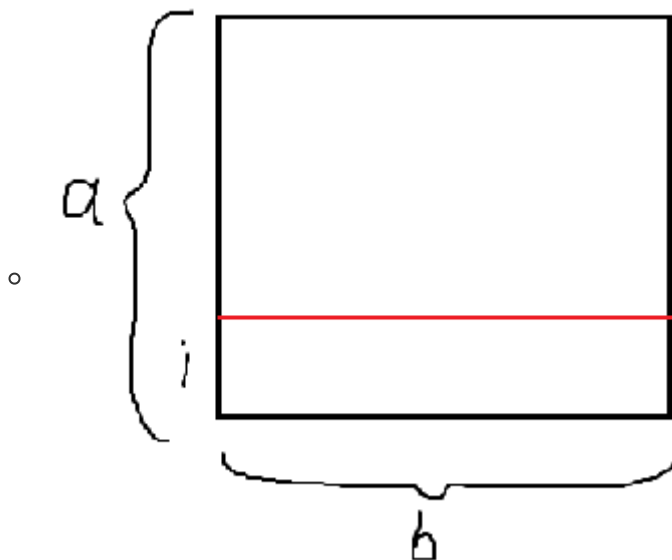
- 给定 2 个长度为 n 的序列 a_i, b_i
- 求 $\forall i \in [1, n], \max_{j=1}^n x_{i,j} = a_i, \max_{j=1}^n x_{j,i} = b_i$ 的非负整数解的个数
- $1 \leq n \leq 10^5, 1 \leq a_i, b_i \leq 10^9$

分析

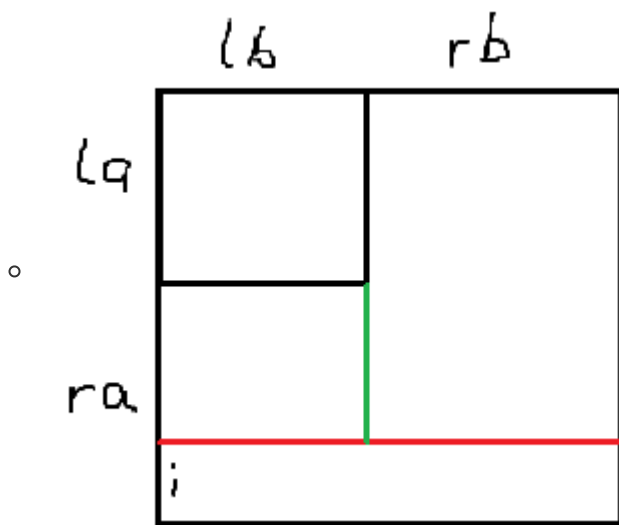
- 首先 a_i, b_i 的顺序并不会影响答案，于是现将 a_i, b_i 排序
- 设 $s_{i,j} = \min(a_i, b_j)$ ，即每个格子满足条件的最大值

	5	5	5	4	4	4	3	3	2	2
5	5	5	5	4	4	4	3	3	2	2
5	5	5	5	4	4	4	3	3	2	2
4	4	4	4	4	4	4	3	3	2	2
4	4	4	4	4	4	4	3	3	2	2
4	4	4	4	4	4	4	3	3	2	2
3	3	3	3	3	3	3	3	3	2	2
3	3	3	3	3	3	3	3	3	2	2
3	3	3	3	3	3	3	3	3	2	2
1	1	1	1	1	1	1	1	1	1	1

- 于是问题就变成了求这样一个矩形/L 形的满足条件的解的数量
- 我们设 F_i 表示矩形中至少有 i 行不满足条件的情况数， $s_{i,j}$ 简写为 s



- 对于不满足条件的行，每个格子可以取 $0 \sim s-1$ ，共有 s^{ib} 种情况
- 对于其它行，值可以任取，每个格子可以取 $0 \sim s$ ，共有 $(s+1)^{(a-i)b}$ 种情况，但我们还要保证每一列都满足条件，所以还要减去不满足条件的情况（即都小于 s ），共 $s^{(a-i)b}$ 种。所以共 $(s+1)^{(a-i)b} - s^{(a-i)b}$ 种
- 然后，我们是在 a 行种取 i 行不满足条件，所以还要乘上 $\binom{a}{i}$
- 所以共 $\binom{a}{i} \times s^{ib} \times ((s+1)^{(a-i)b} - s^{(a-i)b})$ 种
- 最后再容斥
- 对于 L 形的，我们思路相同，设 F_i 表示至少有 i 行不满足条件的情况数，将 L 分为 3 块（图中右下的 L 形是要求的）



- 首先，左上的矩形是已经确定合法的，所以枚举的 i 是 $1 \sim ra$
- 对于不满足条件的行，每个格子可以取 $0 \sim s-1$ ，共有 $s^{i(lb+rb)}$ 种情况
- 对于右上，值可以任取，每个格子可以取 $0 \sim s$ ，共有 $(s+1)^{(ra+la-i)rb}$ 种情况，但我们还要保证每一列都满足条件，所以还要减去不满足条件的情况（即都小于 s ），共 $s^{(ra+la-i)rb}$ 种。所以共 $(s+1)^{(ra+la-i)rb} - s^{(ra+la-i)rb}$ 种
- 对于左上（要求的），值可以任取，每个格子可以取 $0 \sim s$ ，共有 $(s+1)^{(ra-i)lb}$ 种情况，同时，因为我们在左上已经确定合法的矩形中，已经保证了每一列都是合法的，所以不需要再减去列不合法的情况
- 然后，我们是在 ra 行种取 i 行不满足条件，所以还要乘上 $\binom{ra}{i}$
- 所以共 $\binom{ra}{i} \times s^{i(lb+rb)} \times ((s+1)^{(ra+la-i)rb} - s^{(ra+la-i)rb}) \times (s+1)^{(ra-i)lb}$ 种
- 最后再容斥
- 关于容斥，之所以要容斥

- 我们要求的是只有 0 行不合法的情况
- 于是用 $F_0 - F_1$
- 但是在计算 F_1 时，因为算的是 **至少** 1 行不满足，所以在讨论至少第一行不满足时，会出现（举例）第一行和第二行都不满足的情况，而在讨论至少第二行不满足时，也会出现一次相同的情况，这种情况就减了 2 次
- 所以要再加上 F_2
- $F_3, F_4 \cdots$ 同理

代码

```
#include<bits/stdc++.h>
using namespace std;

const int N=1e5+10,P=1e9+7;
int n,a[N],b[N],num[N*2],tot,jc[N],jcinv[N],inv[N],ans=1;

void init_() //预处理阶乘和逆元，用于计算组合数
{
    inv[1]=1;
    for(int i=2;i<N;++i)
        inv[i]=1ll*(P-P/i)*inv[P/i]%P;
    jc[0]=jcinv[0]=1;
    for(int i=1;i<N;++i)
    {
        jc[i]=1ll*jc[i-1]*i%P;
        jcinv[i]=1ll*jcinv[i-1]*inv[i]%P;
    }
}

int mpow(int x,int y)
{
    int res=1;
    for(;y;y>>=1)
    {
        if(y&1) res=1ll*res*x%P;
        x=1ll*x*x%P;
    }
    return res;
}

int C(int x,int y)
{
    if(x==y) return 1;
    if(x<y) return 0;
    return 1ll*jc[x]*jcinv[y]%P*jcinv[x-y]%P;
}

int rongchi(int la,int ra,int lb,int rb,int s)
{
    int res=0,tmp;
    for(int i=0;i<=ra;++i)
    {
        tmp=mpow((P+mpow(s+1,la+ra-i)-mpow(s,la+ra-i))%P,rb);
```

```

        tmp=111*tmp*mpow(s,111*i*(1b+rb)%(P-1))%P*mpow(s+1,111*(ra-i)*1b%(P-1))%P;
        tmp=111*tmp*C(ra,i)%P;
        if(i&1) res+=P-tmp,res%=P;
        else res+=tmp,res%=P;
    }
    return res;
}

int main()
{
    init_();
    scanf("%d",&n);
    for(int i=1;i<=n;++i)
        scanf("%d",&a[i]),num[++tot]=a[i];
    for(int i=1;i<=n;++i)
        scanf("%d",&b[i]),num[++tot]=b[i];
    sort(a+1,a+n+1,greater<int>());
    sort(b+1,b+n+1,greater<int>());
    sort(num+1,num+tot+1,greater<int>()); //num存了不同的s，方便分出L和矩形
    tot=unique(num+1,num+tot+1)-num-1;
    int la=0,ra=0,lb=0,rb=0;
    for(int i=1;i<=tot;++i)
    {
        while(ra<n&&a[ra+1]==num[i]) ++ra;
        while(rb<n&&b[rb+1]==num[i]) ++rb;
        ans=111*ans*rongchi(la,ra-la,lb,rb-lb,num[i])%P;
        la=ra,lb=rb;
    }
    printf("%d",ans);
    return 0;
}

```

T4 ancient

实在没时间了，版题题解就翻书吧