Week7 homework

序列延申与后缀查询

题目描述

有一个初始为空的序列 A*A*，在题目描述中，我们用 ai*a****i* 表示 A*A* 中的第 i*i* 个数字。

你需要支持以下两种操作：

1. 延申：在序列 A*A* 的尾部添加一个数 x*x*（形式化来说，假设当前 A*A* 中有 m*m* 个数，添加之后则有 m+1*m*+1 个数，且 am+1=x*a****m*+1=*x*）。
2. 查询：询问当前序列 A*A* 最后的 k*k* 个数中，最大的数是多少（形式化来说，假设当前 A*A* 中有 m*m* 个数，即询问 max{ai∣m−k+1≤i≤m}max{*ai**m*−*k*+1≤*i*≤*m*} 的值）。

为了强制你在线处理询问，输入数据进行了加密，你需要维护一个值叫 lastans/***a****s****t****a****n****s*（**初始值为 00**），表示上一次询问操作的答案，并且需要使用 lastans/***a****s****t****a****n****s* 来解密出真正的操作内容。具体细节可以查看输入格式。

输入格式

第一行，一个整数 n*n*，表示操作的次数。

接下来 n*n* 行，每行输入两个加密后的整数 a,b*a*,*b*。

你需要用 lastans/***a****s****t****a****n****s* 解密得到 op,v*o****p*,*v*，规则为：

- op=lastans⊕a*o****p*=/***a****s****t****a****n****s*⊕*a*
- v=lastans⊕b*v*=/***a****s****t****a****n****s*⊕*b*

（其中 ⊕⊕ 表示按位异或运算，即 C++ 的 `^` 运算符）

如果 op=1*o****p*=1，那么执行延申操作，在序列 A*A* 尾部添加一个 v*v*。

如果 op=2*o****p*=2，那么执行查询操作，询问序列 A*A* 最后的 v*v* 个数中，最大的数是多少。并且你需要在求解出这个答案之后，用此答案更新 lastans/***a****s****t****a****n****s*，以确保之后的操作能够正确解密。

输入保证第一个操作一定是延申操作，且 op=2*o****p*=2 时，v*v* 不会超过此时序列 A*A* 的长度。

输出格式

为了减少输出文件的大小，你并不需要按顺序输出所有询问的答案，你只需要输出一个值，表示所有询问的答案的和即可（请注意数据范围大小）。

样例 #1

样例输入 #1

```
6
1 2
1 3
1 1
2 3
1 2
3 3
```

Copy

样例输出 #1

```
7
```

Copy

数据范围与提示

【样例解释】

首先进行了3次延申操作，使得序列 A={2,3,1}A={2,3,1} ，然后第四次操作询问了后3个数中的最大值，答案应当为 33 ，于是第五次操作解密后是询问后1个数中的最大值，答案为 11 ，然后第六次操作解密后是询问后2个数中的最大值，答案应当为 33 ，所以最后输出 3+1+3=73+1+3=7 .

【数据范围】

对于 100% 的测试数据满足 n≤106,1≤op≤2,1≤v≤109n≤106,1≤o**p≤2,1≤v≤109 .

输入保证第一个操作一定是延申操作，且 op=2o**p=2 时，vv 不会超过此时序列 AA 的长度。

请注意读入数据的效率。

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

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int n;
    cin >> n;

    long long sum = 0;
    int lastans = 0;
    int m = 0;
    deque<pair<int, int>> decrease;
    for (int i = 0; i < n; ++i) {
        int a, b;
        cin >> a >> b;
        int op = lastans ^ a;
        int v = lastans ^ b;
```

```
        if (op == 1) {

            m++;
            while (!decrease.empty() && decrease.back().first < v) {
                decrease.pop_back();
            }
            decrease.emplace_back(v, m);
        } else if (op == 2) {
            int idx = m - v + 1;
            auto it = lower_bound(decrease.begin(), decrease.end(), make_pair(0,
idx),
                                  [](const pair<int, int>& a, const pair<int,
int>& b)
                                      {
                                  return a.second < b.second;
                                  });
            lastans = it->first;
            sum += lastans;
        }
    }
    cout << sum ;
    return 0;
}
```

逆序对

Description

ZZR 有一个序列 a1,a2,⋯ ,an$a1,a2,\cdots,a**n$，他允许最多进行 kk 次操作，每次操作交换两个相邻元素。

求经过变换后的序列中最少还有多少逆序对。

逆序对指的是二元组 (i,j)(i,j)，其满足 i<j$i<j$ 且 ai>aj$a**i>a**j$。

Format

Input

第一行包含两个数 n,kn,k，表示序列长度和交换两个相邻元素的次数上限。

第二行有 nn 个数，表示序列 a1,a2,⋯ ,an$a1,a2,\cdots,a**n$。

Output

一个数表示最少的逆序对的数量。

Samples

输入样例 1

```
3 1
2 2 1
```

Copy

输出样例 1

```
1
```

Copy

输入样例 2

```
3 0
2 2 1
```

Copy

输出样例 2

```
2
```

Copy

Limitation

1≤n≤1051≤*n*≤105，0≤k≤1090≤*k*≤109，1≤ai≤1091≤*a****i*≤109。

```
#include<bits/stdc++.h>
using namespace std;
#define N 1000005
int a[N] ,b[N];//b为辅助数组
long long cnt;
void merge_sort(int l , int r)
{
    if(r>l)
    {
        int mid = (l+r) / 2 ;
        merge_sort(l , mid);
        merge_sort(mid+1 , r);

        int i = l;
        int p = l , q = mid+1;
        while(p<=mid || q<=r)
        {
            if(q>r || (p<=mid && a[p]<=a[q]))
                b[i++] = a[p++];
            else
```

```
            {
                b[i++] = a[q++];
                cnt += mid -p + 1;
            }
        }

        for(i = l ; i <= r; i++)
            a[i] = b[i];
    }
}
int main()
{
    int n,k;
    cin>>n>>k;
    for(int i = 1 ; i <= n; i ++)cin >> a[i];
    cnt = 0;

    merge_sort(1 , n);
        if(cnt-k<0)cout<<0<<endl;
        else cout << cnt - k <<endl;

    return 0;
}
```

乒乓球比赛

Description

Y 老师举办了学校的乒乓球赛，为了兼顾赛程和公平，她决定采用瑞士轮赛制。

瑞士轮赛制如下：2×n2×*n* 名编号为 1~2n1~2*n* 的选手共进行 *rr* 轮比赛。每轮比赛开始前，以及所有比赛结束后，都会按照总分从高到低对选手进行一次排名。选手的总分为第一轮开始前的初始分数加上已参加过的所有比赛的得分和。总分相同的，约定编号较小的选手排名靠前。

每轮比赛的对阵安排与该轮比赛开始前的排名有关：第 11 名和第 22 名、第 33 名和第 44 名、……、第 2k–12*k*–1 名和第 2k2*k* 名、…… 、第 2n–12*n*–1 名和第 2n2*n* 名，各进行一场比赛。每场比赛胜者得 1 分，负者得 0 分。也就是说除了首轮以外，其它轮比赛的安排均不能事先确定，而是要取决于选手在之前比赛中的表现。

现给定每个选手的初始分数及其实力值，试计算在 *rr* 轮比赛过后，排名第 q*q* 的选手编号是多少。我们假设选手的实力值两两不同，且每场比赛中实力值较高的总能获胜。

Format

Input

输入的第一行是三个正整数 n,r,q*n*,*r*,*q*，每两个数之间用一个空格隔开，表示有 2×n2×*n* 名选手、*rr* 轮比赛，以及我们关心的名次 q*q*。

第二行是 2×n2×*n* 个非负整数 s1,s2,⋯,s2ns1,*s*2,⋯,*s*2*n*，每两个数之间用一个空格隔开，其中 sis**i* 表示编号为 *ii* 的选手的初始分数。

第三行是 2×n2×*n* 个正整数 w1,w2,...,w2nw1,*w*2,...,*w*2*n*，每两个数之间用一个空格隔开，其中 wi*w***i* 表示编号为 i*i* 的选手的实力值。

Output

输出一个整数，即 r*r* 轮比赛结束后，排名第 q*q* 的选手的编号。

Samples

样例输入

```
2 4 2
7 6 6 7
10 5 20 15
```

Copy

样例输出

```
1
```

Copy

Limitation

1≤n≤1051≤*n*≤105，1≤r≤3001≤*r*≤300，1≤q≤2n1≤*q*≤2*n*，0≤s1,s2,...,s2n≤1080≤*s*1,*s*2,...,*s*2*n*≤108，1≤w1,w2,...,w2n≤1081≤*w*1,*w*2,...,*w*2*n*≤108。

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
const int MAXN = 250007;
struct Player {
    int score, power, id;
    bool operator<(const Player& other) const {
        if (score == other.score) {
            return id < other.id;
        }
        return score > other.score;
    }
};

int n, rounds, q;
Player players[MAXN], tempPlayers[MAXN];
int winners[MAXN], losers[MAXN];

bool cmp(const Player &a, const Player &b) {
    return a<b;
}

void mergePlayers() {
    int l = 1, r = 1, mergedCount = 0;
```

```
    while (l <= n/2 && r <= n/2) {
        if (players[winners[l]] < players[losers[r]]) {
            tempPlayers[++mergedCount] = players[winners[l++]];
        } else {
            tempPlayers[++mergedCount] = players[losers[r++]];
        }
    }
    while (l <= n/2) tempPlayers[++mergedCount] = players[winners[l++]];
    while (r <= n/2) tempPlayers[++mergedCount] = players[losers[r++]];


    for (int i = 1; i <= n; ++i) {
        players[i] = tempPlayers[i];
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cin >> n;
    n *= 2;
    cin >> rounds >> q;

    for (int i = 1; i <= n; ++i) {
        cin >> players[i].score;
        players[i].id = i;
    }

    for (int i = 1; i <= n; ++i) {
        cin >> players[i].power;
    }

    sort(players+1 , players + n +1 ,cmp);

    for (int i = 0; i < rounds; ++i) {
        int winCount = 0, loseCount = 0;

        for (int j = 1; j <= n; j += 2) {
            if (players[j].power > players[j + 1].power) {
                winners[++winCount] = j;
                losers[++loseCount] = j + 1;
            } else {
                winners[++winCount] = j + 1;
                losers[++loseCount] = j;
            }
            players[winners[winCount]].score++;
        }

        mergePlayers();
    }

    cout << players[q].id << endl;

    return 0;
}
```

快速矩阵乘法

题目描述

给定 n,a,b,cn,a,b,c 和 f0,f1$f0,f1$，已知递推式

fi=a·fi−1+b·fi−2+c$f_i=a\cdot f_{i-1}+b\cdot f_{i-2}+c$

求 fn mod (109+7)$f_n \bmod (10^9+7)$。

输入输出格式

输入格式

六个数分别表示 n,a,b,c,f0,f1$n,a,b,c,f0,f1$。

输出格式

一个数表示 fn mod (109+7)$f_n \bmod (10^9+7)$

样例

输入数据 1

```
7 1 1 0 1 1
```

Copy

输出数据 1

```
21
```

Copy

数据范围

1≤n≤10181≤$n\le 10^{18}$，1≤a,b,c,f0,f1<109+71≤$a,b,c,f0,f1<10^9+7$。

```
#include <iostream>
#include <vector>

using namespace std;
const int MOD = 1e9 + 7;

vector<vector<long long>> matrixMul(const vector<vector<long long>>& A, const
vector<vector<long long>>& B) {
    int n = A.size();
    vector<vector<long long>> C(n, vector<long long>(n, 0));
```

```
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            for (int k = 0; k < n; ++k) {
                C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % MOD;
            }
        }
    }
    return C;
}

vector<vector<long long>> matrixPow(vector<vector<long long>> base, long long
exp) {
    int n = base.size();
    vector<vector<long long>> res(n, vector<long long>(n, 0));

    for (int i = 0; i < n; ++i) res[i][i] = 1;

    while (exp > 0) {
        if (exp % 2 == 1) {
            res = matrixMul(res, base);
        }
        base = matrixMul(base, base);
        exp /= 2;
    }
    return res;
}

int main() {
    long long n, a, b, c, f0, f1;
    cin >> n >> a >> b >> c >> f0 >> f1;

    if (n == 0) {
        cout << f0 % MOD << endl;
        return 0;
    }
    if (n == 1) {
        cout << f1 % MOD << endl;
        return 0;
    }


    vector<vector<long long>> T = {
        {a, b, c},
        {1, 0, 0},
        {0, 0, 1}
    };

    vector<vector<long long>> Tn = matrixPow(T, n - 1);
    long long fn = (Tn[0][0] * f1 + Tn[0][1] * f0 + Tn[0][2]) % MOD;
    cout << fn << endl;
    return 0;
}
```

你美还是我美

Description

“🍒Σ！！！！”

Y 看了 B 站的抽象视频，感到非常开心。所以他决定举办美丽值对抗赛来调剂同学们枯燥的军训生活。

已知同学分成两组，美丽值分别为 a1,a2,⋯ ,an$a1,a2,\cdots,a**n$ 和 b1,b2,⋯ ,bm$b1,b2,\cdots,b**m$。

美丽值为 xx 和 yy 的两名同学进行对抗会产生 |x−y||x−y| 颗美丽爱心，求两组同学两两对抗能产生多少美丽爱心。即求

∑i=1n∑j=1m|ai−bj|$i=1\sum n**j=1\sum m|a**i-b**j|$

Format

Input

第一行两个整数 n,mn,m 分别表示两组同学的数量。

第二行 nn 个正整数表示 a1,a2,⋯ ,an$a1,a2,\cdots,a**n$。

第三行 mm 个正整数表示 b1,b2,⋯ ,bm$b1,b2,\cdots,b**m$。

Output

一行一个数表示美丽爱心数量。

Samples

输入样例

```
3 2
1 2 3
1 10
```

Copy

输出样例

```
27
```

Copy

Limitation

1≤n,m≤2×1051≤n,m≤2×105，1≤ai,bi≤1081≤$a**i,b**i$≤108。

```
#include <iostream>
#include <vector>
#include <algorithm>
```

```cpp
using namespace std;

typedef long long ll;


int main() {
    int n, m;
    cin >> n >> m;

    vector<ll> a(n), b(m);
    for (int i = 0; i < n; ++i) cin >> a[i];
    for (int i = 0; i < m; ++i) cin >> b[i];

    sort(a.begin(), a.end());
    sort(b.begin(), b.end());

    vector<ll> prefixA(n);
    prefixA[0] = a[0];
    for (int i = 1; i < n; ++i) {
        prefixA[i] = prefixA[i - 1] + a[i];
    }

    ll totalLove = 0;

    for (int j = 0; j < m; ++j) {
        ll bj = b[j];

        int pos = upper_bound(a.begin(), a.end(), bj) - a.begin();


        ll sumLeft;
        if(pos-1<0)sumLeft=0;
        else  sumLeft = prefixA[pos-1];
        ll countLeft = pos;
        totalLove += (bj * countLeft - sumLeft);

        ll sumRight;
        sumRight = prefixA[n-1] - sumLeft;
        ll countRight = n - countLeft;
        totalLove += (sumRight - bj * countRight) ;
    }

    cout << totalLove  << endl;

    return 0;
}
```