Codeforces Round #251 (Div. 2)

Problema D: *Devu and his Brother*

Prof. Edson Alves – Faculdade UnB Gama

Problema

Devu and his brother love each other a lot. As they are super geeks, they only like to play with arrays. They are given two arrays a and b by their father. The array a is given to Devu and b to his brother.

As Devu is really a naughty kid, he wants the minimum value of his array a should be at least as much as the maximum value of his brother's array b.

Now you have to help Devu in achieving this condition. You can perform multiple operations on the arrays. In a single operation, you are allowed to decrease or increase any element of any of the arrays by 1. Note that you are allowed to apply the operation on any index of the array multiple times.

You need to find minimum number of operations required to satisfy Devu's condition so that the brothers can play peacefully without fighting.

Entrada e saída

Input

The first line contains two space-separated integers n, m $(1 \leq n, m \leq 10^5)$. The second line will contain n space-separated integers representing content of the array a $(1 \leq a_i \leq 10^9)$. The third line will contain m space-separated integers representing content of the array b $(1 \leq b_i \leq 10^9)$.

Output

You need to output a single integer representing the minimum number of operations needed to satisfy Devu's condition.

Exemplo de entradas e saídas

Sample Input	Sample Output
2 2 2 3 3 5	3
3 2 1 2 3 3 4	4
3 2 4 5 6 1 2	0

Solução com complexidade $O(N \log N)$

- Seja $f(x) = \sum_i \alpha(a_i, x)$, onde $\alpha(a_i, x) = x - a_i$, se $a_i < x$, e $\alpha(a_i, x) = 0$, caso contrário
- De forma semelhante, seja $g(x) = \sum_j \beta(b_j, x)$, onde $\beta(b_j, x) = b_j - x$, se $b_j > x$, e $\beta(b_j, x) = 0$, caso contrário
- Observe que as funções $f(x)$ e $g(x)$ representam os custos para tornar x uma cota inferior ou superior dos vetores a e b, respectivamente
- Assim, o problema consiste em determinar o mínimo da função $h(x) = f(x) + g(x)$
- Observe que $h'(x) = \mu(a, x) - \rho(b, x)$, onde $\mu(c, x)$ é o número de elementos do vetor a menores do que x e $\rho(b, x)$ é o número de elementos do vetor b maiores do que x
- Como a função $\mu(a, x)$ é não-decrescente e a função $\rho(c, x)$ é não-crescente, a $h'(x)$ é não-decrescente
- Isto significa que $h''(x) \geq 0$, ou seja, $h(x)$ é convexa (unimodal), de modo que seu mínimo pode ser obtido através de uma busca ternária

Solução AC com complexidade $O(N \log N)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

const ll oo { 1'000'000'010LL };

ll h(ll x, const vector<ll>& as, const vector<ll>& bs)
{
    ll y = 0;

    for (auto a : as)
        y += (a < x ? x - a : 0);

    for (auto b : bs)
        y += (b > x ? b - x : 0);

    return y;
}
```

Solução AC com complexidade $O(N \log N)$

```
21 ll solve(const vector<ll>& as, const vector<ll>& bs)
22 {
23     // f(x) é convexa: busca ternária
24     ll a = 0, b = oo, ans = 2'000'000'000'000'000'010LL;
25
26     while (a <= b)
27     {
28         auto m1 = a + (b - a)/3;
29         auto m2 = b - (b - a)/3;
30
31         auto y1 = h(m1, as, bs), y2 = h(m2, as, bs);
32
33         ans = min(ans, y1);
34         ans = min(ans, y2);
35
36         y1 > y2 ? a = m1 + 1 : b = m2 - 1;
37     }
38
39     return ans;
40 }
```

Solução AC com complexidade $O(N \log N)$

```
42 int main()
43 {
44     ios::sync_with_stdio(false);
45
46     int N, M;
47     cin >> N >> M;
48
49     vector<ll> as(N), bs(M);
50
51     for (int i = 0; i < N; ++i)
52         cin >> as[i];
53
54     for (int j = 0; j < M; ++j)
55         cin >> bs[j];
56
57     cout << solve(as, bs) << '\n';
58
59     return 0;
60 }
```