OJ 12049

Just Prune The List

Prof. Edson Alves – UnB/FGA

Problema

You are given two list of integers. You can remove any number of elements from any of them. You have to ensure that after removing some elements both of the list will contain same elements, but not necessarily in same order. For example consider the following two lists

List # 1	1	2	3	2	1
List # 2	1	2	5	2	3

After removing 1 from first list and 5 from second list, both lists will contain same elements. We will find the following lists after removing two elements.

List # 1	1	2	3	2
List # 2	1	2	2	3

What is the minimum number of elements to be removed to obtain two list having same elements?

Entrada e saída

Input

The first line of the input file contains an integer T $(T \leq 100)$ which denotes the total number of test cases. The description of each test case is given below:

First line will contain two integers N $(1 \leq N \leq 10000)$, the number of element in the first list and M $(1 \leq M \leq 10000)$, the number of element in the second list. The next line will contain N integers representing the first list followed by another line having M elements representing the second list. Each integers in the input is 32 bit signed integer.

Output

For each test case output a single line containing the number of elements to be removed. See sample output for exact format.

Exemplo de entradas e saídas

Sample Input

```
1
5 5
1 2 3 2 1
1 2 5 2 3
```

Sample Output

```
2
```

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

- O problema consiste em determinar quantos elementos, em ambas listas, não pertencem a interseção de ambas
- Observe que as listas podem conter elementos repetidos: armazená-los em um `set` levaria a perda de informações importantes
- Uma solução é armazenar ambas listas nos `multisets` s e r e, para cada elemento de r, verificar se ele está ou não em s
- Se estiver, uma cópia dele deve ser removida de s
- Caso contrário, é um elemento a ser eliminado, acrescentando uma unidade à resposta
- Ao final do processo, cada elemento restante em s acrescenta uma unidade ao resultado final
- A STL do C++ pode ser utilizada: a função `set_intersection()` computa a interseção entre dois contêiners ordenados
- Neste caso a resposta passa a ser a soma $N + M$, subtraída do dobro da interseção

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

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

using namespace std;

int solve(multiset<int>& s, const multiset<int>& r)
{
    vector<int> xs;
    set_intersection(s.begin(), s.end(), r.begin(), r.end(), back_inserter(xs));

    return s.size() + r.size() - 2*xs.size();
}

int main()
{
    int T;
    cin >> T;

    while (T--) {
        int N, M, x;
        cin >> N >> M;
```

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

```
22          multiset<int> s, r;
23
24          while (N--)
25          {
26              cin >> x;
27              s.insert(x);
28          }
29
30          while (M--)
31          {
32              cin >> x;
33              r.insert(x);
34          }
35
36          cout << solve(s, r) << '\n';
37      }
38
39      return 0;
40  }
```