SPOJ INVCNT

Inversion Count

Prof. Edson Alves – UnB/FGA

Problema

Let $A[0 \ldots n-1]$ be an array of n distinct positive integers. If $i < j$ and $A[i] > A[j]$ then the pair (i, j) is called an inversion of A. Given n and an array A your task is to find the number of inversions of A.

Entrada e saída

Input

The first line contains t, the number of testcases followed by a blank space. Each of the t tests start with a number n ($n \leq 200000$). Then $n+1$ lines follow. In the ith line a number $A[i-1]$ is given ($A[i-1] \leq 10^7$). The $(n+1)$th line is a blank space.

Output

For every test output one line giving the number of inversions of A.

Exemplo de entradas e saídas

Sample Input

```
2

3
3
1
2

5
2
3
8
6
1
```

Sample Output

```
2
5
```

Solução

- Uma árvore de Fenwick pode ser utilizada para manter um histograma dos números já processados
- Assim, se os a_j elementos do vetor de entrada forem processados um a um, do fim para o início, o número de inversões onde j é o segundo elemento do par, corresponde a $RSQ(0, a_j - 1)$, isto é, ao total de números que são estritamente menores que a_j e que já apareceram no vetor
- Se os elementos a_i forem processados do início ao fim, o número de inversões onde i é o primeiro elemento do par correspondem a $RSQ(i + 1, M)$, onde $M = 10^7$ é o maior valor possível para um elemento do vetor
- Esta solução tem complexidade $O(TN \log M)$, onde T é o número de casos de teste

Solução AC com complexidade $O(TN \log M)$

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

using namespace std;
using ll = long long;

const int MAX { 10000010 };

class BITree {
private:
    vector<int> ts;
    size_t N;

public:
    BITree(size_t n) : ts(n + 1, 0), N(n) {}

    int RSQ(int i, int j)
    {
        return RSQ(j) - RSQ(i - 1);
    }
```

Solução AC com complexidade $O(TN \log M)$

```
21 private:
22     int LSB(int n) { return n & (-n); }
23
24     int RSQ(int i)
25     {
26         int sum = 0;
27
28         while (i >= 1) {
29             sum += ts[i];
30             i -= LSB(i);
31         }
32
33         return sum;
34     }
35
36 public:
37     void add(size_t i, const int& x)
38     {
39         if (i == 0)
40             return;
```

Solução AC com complexidade $O(TN \log M)$

```
42          while (i <= N)
43          {
44              ts[i] += x;
45              i += LSB(i);
46          }
47      }
48  };
49
50  ll solve(const vector<int>& as, int N)
51  {
52      BITree ft(MAX);
53      ll ans = 0;
54
55      for (int i = N; i > 0; --i) {
56          ans += ft.RSQ(0, as[i] - 1);
57          ft.add(as[i], 1);
58      }
59
60      return ans;
61  }
```

Solução AC com complexidade $O(TN \log M)$

```
63 int main() {
64     ios::sync_with_stdio(false);
65
66     int T; cin >> T;
67
68     while (T--) {
69         int N; cin >> N;
70
71         vector<int> as(N + 1);
72
73         for (int i = 1; i <= N; ++i)
74             cin >> as[i];
75
76         auto ans = solve(as, N);
77
78         cout << ans << '\n';
79     }
80
81     return 0;
82 }
```