Codeforces Round #198 (Div. 2)

Problema D: *Bubble Sort Graph*

Prof. Edson Alves – UnB/FGA

Problema

Iahub recently has learned Bubble Sort, an algorithm that is used to sort a permutation with n elements $a_1, a_2, \ldots, a_n$ in ascending order. He is bored of this so simple algorithm, so he invents his own graph. The graph (let's call it G) initially has n vertices and 0 edges. During Bubble Sort execution, edges appear as described in the following algorithm (pseudocode).

Problema

```
procedure bubbleSortGraph()
    build a graph G with n vertices and 0 edges
    repeat
        swapped = false
        for i = 1 to n - 1 inclusive do:
            if a[i] > a[i + 1] then
                add an undirected edge in G between a[i] and a[i + 1]
                swap( a[i], a[i + 1] )
                swapped = true
            end if
        end for
    until not swapped
    /* repeat the algorithm as long as swapped value is true. */
end procedure
```

Problema

For a graph, an independent set is a set of vertices in a graph, no two of which are adjacent (so there are no edges between vertices of an independent set). A maximum independent set is an independent set which has maximum cardinality. Given the permutation, find the size of the maximum independent set of graph G, if we use such permutation as the premutation a in procedure `bubbleSortGraph`.

Entrada e saída

Input

The first line of the input contains an integer n ($2 \leq n \leq 10^5$). The next line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$).

Output

Output a single integer – the answer to the problem.

Exemplo de entradas e saídas

Sample Input

```
3
3 1 2
```

Sample Output

```
2
```

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

- Gerar o grafo G por meio da execução do código do *bubblesort* apresentado leva a um veredito TLE, uma vez que, no pior caso, há $O(N^2)$ arestas (sequência em ordem decrescente)
- Mesmo que fosse possível construir o grafo G em tempo hábil, o maior conjunto independente é um problema *NP-Hard*, e como $N \leq 10^5$, novamente o veredito seria TLE
- O que deve ser observado é que não existirá uma aresta entre a_i e a_j se $a_i < a_j$, com $i < j$
- Observe que, pela transitividade, se não existe uma aresta entre a_i e a_j, e também não há aresta entre a_j e a_k, não haverá uma aresta entre a_i e a_k

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

- Deste modo, um conjunto independente em G será uma sequência crescente de $a = \{a_1, a_2, \ldots, a_N\}$
- A resposta do problema, portanto, será a maior subsequência crescente de a
- Dados os limites do problema, a implementação quadrática levaria ao TLE
- Portanto, o problema deve ser resolvido pela implementação linearítmica da LIS

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

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

using namespace std;

const int oo { 2000000010 };

int solve(int N, const vector<int>& as)
{
    vector<int> lis(N + 1, oo);
    lis[0] = 0;
    auto ans = 0;

    for (int i = 0; i < N; ++i)
    {
        auto it = lower_bound(lis.begin(), lis.end(), as[i]);
        auto pos = (int) (it - lis.begin());

        ans = max(ans, pos);
        lis[pos] = min(as[i], lis[pos]);
    }
```

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

```
22      return ans;
23  }
24
25  int main()
26  {
27      ios::sync_with_stdio(false);
28
29      int N;
30      cin >> N;
31
32      vector<int> as(N);
33
34      for (int i = 0; i < N; ++i)
35          cin >> as[i];
36
37      cout <<  solve(N, as) << '\n';
38
39      return 0;
40  }
```