Codechef COW207

Naruto and Rectangles

Prof. Edson Alves – UnB/FGA

Problema

Given n non-negative integers $a_1, a_2, \ldots, a_n$, where each represents a point at coordinate (i, a_i). n vertical lines are drawn such that the two endpoints of the line i is at (i, a_i) and $(i, 0)$.

Naruto wants to draw a rectangle using these lines. The base should be the x-axis, while the two sides should be any two lines from the above-given lines. The other lines will magically disappear. Note that he cannot move the lines. The only action done my Naruto is to draw the top line of the rectangle by connecting any two lines.

Now naruto wants to know the maximum possible area of rectangle he can get. Help him find it.

Entrada e saída

Input

- First line will contain T, number of testcases. Then the testcases follow.
- First line of each testcase contains n.
- Second line contains n space separated integers $a_1, a_2, \ldots, a_n$.

Output

For each testcase, output in a single line, the maximum area.

Constraints

- $1 \leq T \leq 10$
- $2 \leq N \leq 10^5$
- $2 \leq a_i \leq 10^3$

Exemplo de entradas e saídas

Sample Input

```
1
9
1 8 6 2 5 4 8 3 7
```

Sample Output

```
49
```

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

- Para cada par (i, j), com $j > i$, o retângulo de maior área que pode ser formado terá base $b = (j - i)$ e altura $h = \min(a_i, a_j)$
- Como há $O(N^2)$ pares deste tipo, uma solução que verifique todos eles tem complexidade $O(N^2)$ e, neste problema, tem veredito TLE
- Observe que, para um i fixo, basta verificar apenas dois índices: o elemento L mais à esquerda de i com $a_L \geq a_i$ e o elemento R mais à direita de i tal que $a_R \geq a_i$
- Estes dois elementos podem ser identificados, de forma eficiente, por meio de ordenação e dois ponteiros
- Primeiramente, monte um vetor de pares (a_i, i) e ordene este vetor em ordem não-crescente

Solução com complexidade $O(N)$

- Inicialize os ponteiros $L = \infty$ e $R = -\infty$, e a resposta com zero
- Para cada par do vetor ordenado, processe o pares (a, i) e (a_L, L), se $L < i$, e os pares (a, i) e (a_R, R), se $R > i$
- Após uma possível atualização da resposta, ajuste os ponteiros: $L = \min(L, i)$, $R = \max(R, i)$
- Observe que, com esta estratégia, a altura a_i do próximo elemento a ser processado será inferior à altura dos elementos já processados, de modo que ela pode ser combinada, de forma ótima, com qualquer um deles
- Por isso basta manter o registro, dentre eles, do que está mais à esquerda (L) e mais à direta (R) do vetor
- Esta solução tem complexidade $O(N \log N)$, por causa da ordenação

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

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

using namespace std;
using ii = pair<int, int>;
const int oo { 2'000'000'010 };

int solve(int N, const vector<int>& xs)
{
    vector<ii> ps(N);

    for (int i = 0; i < N; ++i)
        ps[i] = ii(xs[i], i);

    sort(ps.begin(), ps.end(), greater<ii>());

    auto L = oo, R = -oo, ans = 0;

    for (auto p : ps)
    {
        auto [x, i] = p;
```

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

```
22          if (L < i)
23              ans = max(ans, (i - L)*min(x, xs[L]));
24
25          if (R > i)
26              ans = max(ans, (R - i)*min(x, xs[R]));
27
28          L = min(L, i);
29          R = max(R, i);
30      }
31
32     return ans;
33 }
34
35 int main()
36 {
37      ios::sync_with_stdio(false);
38
39      int T;
40      cin >> T;
```

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

```
42      while (T--)
43      {
44          int N;
45          cin >> N;
46
47          vector<int> xs(N);
48
49          for (int i = 0; i < N; ++i)
50              cin >> xs[i];
51
52          auto ans = solve(N, xs);
53
54          cout << ans << '\n';
55      }
56
57      return 0;
58  }
```