

Olimpíada Brasileira de Informática 2017

Nível Sênior – Fase 2: *Upsolving*

Prof. Edson Alves – UnB/FGA

2020

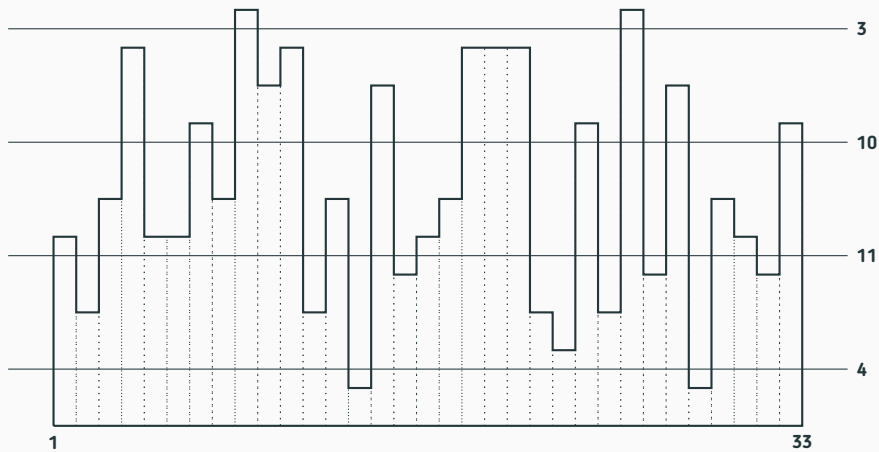
1. Cortando o Papel

Cortando o Papel

Problema

Uma folha de papel é composta de uma sequência de retângulos com diferentes alturas mas com larguras fixas, tal que as bases dos retângulos estão assentadas em uma linha horizontal. A figura ilustra uma folha exemplo com 33 retângulos. Nós gostaríamos de fazer um único corte horizontal, com a ajuda de um estilete e uma régua, que maximize o número resultante de pedaços separados pelo corte. A figura mostra quatro diferentes cortes que resultariam, respectivamente, em 4, 11, 10 e 3 pedaços.

Problema



Entrada

A primeira linha da entrada contém um inteiro N , representando o número de retângulos na folha de papel. A segunda linha contém N inteiros A_i , $1 \leq i \leq N$, representando a sequência de alturas dos retângulos.

Saída

Seu programa deve imprimir uma linha contendo um inteiro representando o número máximo de pedaços possível, com um único corte horizontal.

Restrições

- $2 \leq N \leq 10^5$
- $1 \leq A_i \leq 10^9$, para $1 \leq i \leq N$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 40 pontos, $N \leq 1000$

Exemplo de entradas e saídas

Entrada

5 6

1 2 4

1 3 3

4 3 6

4 5 2

2 4 1

3 5 5

Saída

7

Exemplo de entradas e saídas

Entrada

7 10

1 2 5

3 1 32

1 4 3

2 3 4

2 6 20

6 3 1

6 4 9

6 5 6

3 7 18

5 7 2

Saída

18

• a

Solução TLE $O(MN)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int pieces(double x, int N, const vector<int>& hs)
6 {
7     auto L = 0, res = 0;
8
9     while (L < N)
10    {
11        if (hs[L] <= x)
12        {
13            ++L;
14            continue;
15        }
16
17        auto R = L + 1;
18
19        while (R < N and hs[R] > x)
20            ++R;
21    }
```

Solução TLE $O(MN)$

```
22         ++res;
23         L = R;
24     }
25
26     return res + 1;
27 }
28
29 int solve(int N, const vector<int>& hs)
30 {
31     int M = *max_element(hs.begin(), hs.end()), ans = 2;
32
33     for (int i = 1; i <= M; ++i)
34         ans = max(ans, pieces(i, N, hs));
35
36     return ans;
37 }
38
39 int main()
40 {
41     ios::sync_with_stdio(false);
42 }
```

Solução TLE $O(MN)$

```
43     int N;  
44     cin >> N;  
45  
46     vector<int> hs(N);  
47  
48     for (int i = 0; i < N; ++i)  
49         cin >> hs[i];  
50  
51     auto ans = solve(N, hs);  
52  
53     cout << ans << '\n';  
54  
55     return 0;  
56 }
```

Solução $O(N \log N)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ii = pair<int, int>;
5
6 int solve(int N, vector<int>& hs)
7 {
8     // Compressão dos adjacentes iguais
9     auto it = unique(hs.begin(), hs.end());
10
11     N = (int) distance(hs.begin(), it);
12     hs.resize(N);
13
14     // Identificação dos pontos críticos
15     vector<int> cs;
16
17     for (int i = 1; i <= N - 2; ++i)
18         if ((hs[i - 1] < hs[i] and hs[i] > hs[i + 1]) ||
19             (hs[i - 1] > hs[i] and hs[i] < hs[i + 1]))
20             cs.push_back(hs[i]);
21 }
```

Solução $O(N \log N)$

```
22 // Finaliza com um mínimo
23 if (cs.size() % 2)
24     cs.push_back(0);
25
26 // Geração dos eventos para o sweep line
27 vector<ii> es;
28 int OPEN = 1, CLOSE = -1, ans = 1, open = 0;
29
30 for (size_t i = 0; i < cs.size(); i += 2)
31 {
32     es.push_back(ii(cs[i + 1], OPEN));
33     es.push_back(ii(cs[i], CLOSE));
34 }
35
36 sort(es.begin(), es.end());
37
38 for (auto e : es)
39 {
40     open += e.second;
41     ans = max(ans, open);
42 }
```

Solução $O(N \log N)$

```
43
44     return ans + 1;
45 }
46
47 int main()
48 {
49     ios::sync_with_stdio(false);
50
51     int N;
52     cin >> N;
53
54     vector<int> hs(N + 2, 0);
55
56     for (int i = 1; i <= N; ++i)
57         cin >> hs[i];
58
59     cout << solve(N, hs) << '\n';
60
61     return 0;
62 }
```


1. Dario e Xerxes
2. Frete
3. Mapa
4. Cortando o Papel
5. URI 1828 – Bazinga!