# AtCoder

AtCoder Beginner Contest 047: *Upsolving*

Prof. Edson Alves - UnB/FGA

2020

## Sumário

## A – Fighting over Candies

## Problema

Two students of AtCoder Kindergarten are fighting over candy packs.

There are three candy packs, each of which contains $a$, $b$, and $c$ candies, respectively.

Teacher Evi is trying to distribute the packs between the two students so that each student gets the same number of candies. Determine whether it is possible.

Note that Evi cannot take candies out of the packs, and the whole contents of each pack must be given to one of the students.

## Entrada e saída

### Constraints

- $1 \leq a, b, c \leq 100$

### Input

Input is given from Standard Input in the following format:

```
a b c
```

### Output

If it is possible to distribute the packs so that each student gets the same number of candies, print ‘`Yes`’. Otherwise, print ‘`No`’.

## Exemplo de entradas e saídas

| **Entrada** | **Saída** |
|---|---|
| `10 30 20` | `Yes` |
| `30 30 100` | `No` |
| `56 25 31` | `Yes` |

## Solução

- Como os pacotes não podem ser fracionados, a única forma possível de se distribuir os doces igualmente é dar dois pacotes para uma criança e um para a outra
- Naturalmente, a criança que vai receber um único pacote deve receber o pacote com o maior número de doces
- Assim, se os valores $a, b, c$ estiverem em ordem crescente, o resposta será '`Yes`' se, e somente se, $c = a + b$
- Mesmo com a rotina de ordenação, a complexidade da solução é $O(1)$

## Solução $O(1)$

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

using namespace std;

string solve(int a, int b, int c)
{
    vector<int> xs { a, b, c };
    sort(xs.begin(), xs.end());

    return xs[2] == xs[0] + xs[1] ? "Yes" : "No";
}

int main()
{
    int a, b, c;
    cin >> a >> b >> c;

    cout << solve(a, b, c) << '\n';

    return 0;
}
```

## B – Snuke’s Coloring 2 (ABC Edit)

## Problema

There is a rectangle in the $xy$-plane, with its lower left corner at $(0, 0)$ and its upper right corner at $(W, H)$. Each of its sides is parallel to the $x$-axis or $y$-axis. Initially, the whole region within the rectangle is painted white.

Snuke plotted $N$ points into the rectangle. The coordinate of the $i$-th $(1 \leq i \leq N)$ point was $(x_i, y_i)$.

Then, he created an integer sequence $a$ of length $N$, and for each $1 \leq i \leq N$, he painted some region within the rectangle black, as follows:

- If $a_i = 1$, he painted the region satisfying $x < x_i$ within the rectangle.
- If $a_i = 2$, he painted the region satisfying $x > x_i$ within the rectangle.
- If $a_i = 3$, he painted the region satisfying $y < y_i$ within the rectangle.
- If $a_i = 4$, he painted the region satisfying $y > y_i$ within the rectangle.

Find the area of the white region within the rectangle after he finished painting.

## Entrada e saída

**Constraints**

- $1 \leq W, H \leq 100$
- $1 \leq N \leq 100$
- $0 \leq x_i \leq W (1 \leq i \leq N)$
- $0 \leq y_i \leq H (1 \leq i \leq N)$
- $W, H, x_i$ and $y_i$ are integers.
- $a_i$ $(1 \leq i \leq N)$ is $1, 2, 3$ or $4$.

## Entrada e saída

### Input

Input is given from Standard Input in the following format:

$$\begin{array}{lll} W & H & N \\ x_1 & y_1 & a_1 \\ x_2 & y_2 & a_2 \\ \ldots & & \\ x_N & y_N & a_N \end{array}$$

### Output

Print the area of the white region within the rectangle after Snuke finished painting.

## Exemplo de entradas e saídas

| **Entrada** | **Saída** |
|---|---|
| ```5 4 2```<br>```2 1 1```<br>```3 3 4``` | 9 |
| ```5 4 3```<br>```2 1 1```<br>```3 3 4```<br>```1 4 2``` | 0 |
| ```10 10 5```<br>```1 6 1```<br>```4 1 3```<br>```6 9 4```<br>```9 4 2```<br>```3 1 3``` | 64 |

## Solução

- Observe que, a cada ação, a região branca ou se mantém, ou se reduz em uma de suas dimensões
- Assuma que a região branca seja delimita no eixo-$x$ pelo intervalo $[x_L, x_R]$ e no eixo-$y$ pelo intervalo $[y_L, y_R]$
- Para $a_i = 1$, o valor de $x_L$ será dado por $x_L = \max(x_L, x_i)$
- De forma semelhante, para $a_i = 2$ vale que $x_R = \min(x_R, x_i)$
- O mesmo vale para $a_i = 3$ e $a_i = 4$, que afetam as coordenadas $y$ de maneira similar
- A solução tem complexidade $O(N)$, uma vez que cada pintura pode ser feita em $O(1)$

## Solução $O(N)$

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

using namespace std;

int main()
{
    int W, H, N;
    cin >> W >> H >> N;

    int xL = 0, xR = W, yL = 0, yR = H;

    while (N--)
    {
        int x, y, a;
        cin >> x >> y >> a;

        switch (a) {
        case 1:
            xL = max(xL, x);
            break;

```

## Solução $O(N)$

```
            case 2:
                xR = min(xR, x);
                break;

            case 3:
                yL = max(yL, y);
                break;

            case 4:
                yR = min(yR, y);
                break;
            }
        }

    auto w = max(xR - xL, 0), h = max(yR - yL, 0);
    int ans = w*h;

    cout << ans << '\n';

    return 0;
}
```

## C – 1D Reversi

## Problema

Two foxes Jiro and Saburo are playing a game called 1D Reversi. This game is played on a board, using black and white stones. On the board, stones are placed in a row, and each player places a new stone to either end of the row. Similarly to the original game of Reversi, when a white stone is placed, all black stones between the new white stone and another white stone, turn into white stones, and vice versa.

In the middle of a game, something came up and Saburo has to leave the game. The state of the board at this point is described by a string $S$. There are $|S|$ (the length of $S$) stones on the board, and each character in $S$ represents the color of the $i$-th $(1 \leq i \leq |S|)$ stone from the left. If the $i$-th character in $S$ is '`B`', it means that the color of the corresponding stone on the board is black. Similarly, if the $i$-th character in $S$ is '`W`', it means that the color of the corresponding stone is white.

Jiro wants all stones on the board to be of the same color. For this purpose, he will place new stones on the board according to the rules. Find the minimum number of new stones that he needs to place.

## Entrada e saída

**Constraints**

- $1 \leq |S| \leq 10^5$
- Each character in $S$ is ‘B’ or ‘W’.

**Input**

Input is given from Standard Input in the following format:

```
S
```

**Output**

Print the minimum number of new stones that Jiro needs to place for his purpose.

## Exemplo de entradas e saídas

| **Entrada** | **Saída** |
|---|---|
| `BBBWW` | 1 |
| `WWWWWW` | 0 |
| `WBWBWBWBWB` | 9 |

## Solução

- Embora não seja óbvio à primeira vista, transformar todas as peças em pretas ou em brancas exige o mesmo número de operações
- A estratégia de solução é gulosa: enquanto houver duas cores entre as peças, coloque no início da fila uma peça de cor oposta à da primeira peça
- Isto vai acontecer $K - 1$ vezes, onde $K$ é o número de blocos contíguos de peças da mesma cor
- Assim, a solução consiste em determinar $K$, o que pode ser feito por meio da técnica dos dois ponteiros
- Assim a solução tem complexidade $O(N)$

## Solução $O(N)$

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

using namespace std;

int solve(const string& s)
{
    int ans = 0, N = (int) s.size();

    int L = 0;

    while (L < N)
    {
        auto R = L;

        while (R < N and s[R] == s[L])
            ++R;

        if (R == N)
            break;

        ++ans;
```

## Solução $O(N)$

```
            L = R;
    }

    return ans;
}

int main()
{
    ios::sync_with_stdio(false);

    string s;
    cin >> s;

    auto ans = solve(s);

    cout << ans << '\n';

    return 0;
}
```

## D – An Invisible Hand

## Problema

There are $N$ towns located in a line, conveniently numbered 1 through $N$. Takahashi the merchant is going on a travel from town 1 to town $N$, buying and selling apples.

Takahashi will begin the travel at town 1, with no apple in his possession. The actions that can be performed during the travel are as follows:

- *Move*: When at town $i$ $(i < N)$, move to town $i + 1$.
- *Merchandise*: Buy or sell an arbitrary number of apples at the current town. Here, it is assumed that one apple can always be bought and sold for $A_i$ yen (the currency of Japan) at town $i$ $(1 \leq i \leq N)$), where $A_i$ are distinct integers. Also, you can assume that he has an infinite supply of money.

For some reason, there is a constraint on merchandising apple during the travel: the sum of the number of apples bought and the number of apples sold during the whole travel, must be at most $T$. (Note that a single apple can be counted in both.)

## Problema

During the travel, Takahashi will perform actions so that the profit of the travel is maximized. Here, the profit of the travel is the amount of money that is gained by selling apples, minus the amount of money that is spent on buying apples. Note that we are not interested in apples in his possession at the end of the travel.

Aoki, a business rival of Takahashi, wants to trouble Takahashi by manipulating the market price of apples. Prior to the beginning of Takahashi's travel, Aoki can change $A_i$ into another arbitrary non-negative integer $A_i'$ for any town $i$, any number of times. The cost of performing this operation is $|A_i - A_i'|$. After performing this operation, different towns may have equal values of $A_i$.

Aoki's objective is to decrease Takahashi's expected profit by at least 1 yen. Find the minimum total cost to achieve it. You may assume that Takahashi's expected profit is initially at least 1 yen.

## Entrada e saída

**Constraints**

- $1 \le N \le 10^5$
- $1 \le A_i \le 10^9$ $(1 \le i \le N)$
- $A_i$ are distinct.
- $2 \le T \le 10^9$
- In the initial state, Takahashi's expected profit is at least 1 yen.

## Entrada e saída

### Input

Input is given from Standard Input in the following format:

```
$N$  $T$
$A_1$  $A_2$  ...  $A_N$
```

### Output

Print the minimum total cost to decrease Takahashi's expected profit by at least 1 yen.

## Exemplo de entradas e saídas

| **Entrada** | **Saída** |
|---|---|
| 3 2<br>100 50 200 | 1 |
| 5 8<br>50 30 40 10 20 | 2 |
| 10 100<br>7 10 4 5 9 3 6 8 2 1 | 2 |

## Solução

- Dados os limites do problema, não é possível propôr soluções que avaliem todas as possíveis interações, mesmo que seja por meio de programação dinâmica
- Assim, para se chegar a solução é preciso simplificar ao máximo as opções a serem consideradas
- Primeiramente, observe que se maças forem compradas na $i$-ésima cidade, elas devem ser vendidas na cidade $j$ ($i < j \leq N$) com maior $A_j$
- Assim, para cada cidade é possível computar o lucro máximo $L_i$ de se adquirir $T/2$ maças em $i$ e vendê-las pelo maior lucro possível
- Outra importante observação é que é ótimo comprar $T/2$ maças em uma mesma cidade e vendê-las todas em ou única outra cidade

## Solução

- Seja $L = \max\{L_1, L_2, \ldots, L_N\}$
- Para reduzir o lucro em um mais ienes basta reduzir o valor do $A_j$ associado ao maior lucro
- Porém, caso exista mais do que um valor $L_i$ que seja igual a $L$, a redução de um $A_j$ apenas não é suficiente
- Logo para cada $L_i = L$ devemos reduzir 1 iene no valor $A_j$ associado
- Portanto a solução é a quantidade de tais $L_i$
- Usando uma fila com prioridades e computando cada $L_i$ em $O(1)$, a solução tem complexidade $O(N \log N)$

## Solução $O(N \log N)$

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

using namespace std;

int solve(int N, int T, const vector<int>& xs)
{
    priority_queue<int> pq;
    int best = xs[N - 1];

    for (int i = N - 2; i >= 0; --i)
    {
        pq.push(best - xs[i]);
        best = max(best, xs[i]);
    }

    auto m = pq.top(), ans = 0;

    while (not pq.empty() and pq.top() == m)
    {
        ++ans;
        pq.pop();
```

## Solução $O(N \log N)$

```
    }

    return ans;
}

int main()
{
    ios::sync_with_stdio(false);

    int N, T;
    cin >> N >> T;

    vector<int> xs(N);

    for (int i = 0; i < N; ++i)
        cin >> xs[i];

    cout << solve(N, T, xs) << '\n';

    return 0;
}
```

## Referências

1. AtCoder Beginner Contest 047 – Problem A: Fighting Over Candies
2. AtCoder Beginner Contest 047 – Problem B: Snuke's Coloring 2 (ABC Edit)
3. AtCoder Beginner Contest 047 – Problem C: 1D Reversi
4. AtCoder Beginner Contest 047 – Problem D: An Invisible Hand