AtCoder

AtCoder Beginner Contest 048: *Upsolving*

Prof. Edson Alves - UnB/FGA

2020

Sumário

A - AtCoder *** Contest

Problema

Snuke is going to open a contest named "AtCoder s Contest". Here, s is a string of length 1 or greater, where the first character is an uppercase English letter, and the second and subsequent characters are lowercase English letters.

Snuke has decided to abbreviate the name of the contest as "AxC". Here, x is the uppercase English letter at the beginning of s.

Given the name of the contest, print the abbreviation of the name.

Entrada e saída

Constraints

- The length of s is between 1 and 100, inclusive.
- The first character in s is an uppercase English letter.
- The second and subsequent characters in s are lowercase English letters.

Input

Input is given from Standard Input in the following format:

```
AtCoder s Contest
```

Output

Print the abbreviation of the name of the contest.

Exemplo de entradas e saídas

Entrada	Saída
`AtCoder Beginner Contest`	`ABC`
`AtCoder Snuke Contest`	`ASC`
`AtCoder X Contest`	`AXC`

Solução

- Este é um problema bastante simples: basta extrair o primeiro caractere da segunda linha da entrada e inserí-lo na segunda posição da string que corresponde à solução
- O primeiro elemento de uma string pode ser acessado por meio da notação de colchetes (`s[0]`) ou através do método `front()`
- Ambas formas tem complexidade $O(1)$
- Com as garantias da entrada, não é necessário nenhum processamento deste caractere

Solução $O(1)$

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

using namespace std;

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

    string ans { "AxC" };
    ans[1] = b.front();

    cout << ans << '\n';

    return 0;
}
```

B - Between a and b...

Problema

You are given nonnegative integers a and b ($a \leq b$), and a positive integer x.
Among the integers between a and b, inclusive, how many are divisible by x?

Entrada e saída

Constraints

- $0 \leq a \leq b \leq 10^{18}$
- $1 \leq x \leq 10^{18}$

Input

Input is given from Standard Input in the following format:

a b x

Output

Print the number of the integers between a and b, inclusive, that are divisible by x.

Exemplo de entradas e saídas

Entrada	Saída
4 8 2	3
0 5 1	6
9 9 2	0
1 10000000000000000 3	3333333333333333

Solução

- O problema consiste em determinar o número de múltiplos de x no intervalo $[a, b]$
- Seja $m(x, b)$ o número de múltiplos de x em $[1, b]$
- É possível mostrar que

$$m(x, b) = \left\lfloor \frac{b}{x} \right\rfloor$$

- Assim, a solução do problema é $m(x, b) - m(x, a - 1)$
- Cuidado, entretanto, com o caso especial $a = 0$
- Neste caso, a solução é $m(x, b) + 1$, uma vez que zero é múltiplo de qualquer inteiro
- Esta solução tem complexidade $O(1)$

Solução $O(N)$

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

using namespace std;
using ll = long long;

int main()
{
    ll a, b, x;
    cin >> a >> b >> x;

    ll p = b / x, q = (a - 1)/x;

    auto ans = a > 0 ? p - q : p + 1;

    cout << ans << '\n';

    return 0;
}
```

C - Boxes and Candies

Problema

There are N boxes arranged in a row. Initially, the i-th box from the left contains a_i candies.

Snuke can perform the following operation any number of times:

- Choose a box containing at least one candy, and eat one of the candies in the chosen box.

His objective is as follows:

- Any two neighboring boxes contain at most x candies in total.

Find the minimum number of operations required to achieve the objective.

Entrada e saída

Constraints

- $2 \leq N \leq 10^5$
- $0 \leq a_i \leq 10^9$
- $0 \leq x \leq 10^9$

Input

Input is given from Standard Input in the following format:

N x
a_1 a_2 ... a_N

Output

Print the minimum number of operations required to achieve the objective.

Exemplo de entradas e saídas

Entrada	Saída
3 3 2 2 2	1
6 1 1 6 1 2 0 4	11
5 9 3 1 4 1 5	0
2 0 5 5	10

Solução

- Como o problema determina que a soma dos doces entre duas caixas adjancentes deve ser, no máximo, igual a x, é sempre ótimo diminur o máximo possível da caixa que é adjacente a outras duas
- Assim, considere que $s = a_{i-1} + a_i$, com $1 < i \leq N$
- Se $s \leq x$, não é preciso remover nenhum doce
- Se $s > x$, faça $d = s - x$
- Dos d doces a serem removidos, é ótimo remover $k = \min\{d, a_i\}$ doces de a_i
- Isto porque, além de realizar a devida correção, subtrair k doces de a_i pode auxiliar a correção do par de elementos vizinhos a_i e a_{i+1}
- Como todos os pares de elementos devem ser avaliados, esta solução tem complexidade $O(N)$

Solução $O(N)$

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

using namespace std;
using ll = long long;

ll solve(int N, ll x, vector<ll>& xs)
{
    ll ans = 0;

    for (int i = 1; i < N; ++i)
    {
        auto sum = xs[i] + xs[i - 1];
        auto diff = max(0LL, sum - x);

        ans += diff;
        xs[i] -= min(diff, xs[i]);
    }

    return ans;
}

```

Solução $O(N)$

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

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

- $1 \leq N \leq 10^5$
- $1 \leq A_i \leq 10^9$ $(1 \leq i \leq N)$
- A_i are distinct.
- $2 \leq T \leq 10^9$
- 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 2 100 50 200	1
5 8 50 30 40 10 20	2
10 100 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 $O(N \log N)$

```
22      }
23
24      return ans;
25  }
26
27  int main()
28  {
29      ios::sync_with_stdio(false);
30
31      int N, T;
32      cin >> N >> T;
33
34      vector<int> xs(N);
35
36      for (int i = 0; i < N; ++i)
37          cin >> xs[i];
38
39      cout << solve(N, T, xs) << '\n';
40
41      return 0;
42  }
```

Referências

1. AtCoder Beginner Contest 048 - Problem A: AtCoder *** Contest
2. AtCoder Beginner Contest 048 - Problem B: Between a and b...
3. AtCoder Beginner Contest 048 - Problem C: Boxes and Candies
4. AtCoder Beginner Contest 048 - Problem D: An Ordinary Game