AtCoder

AtCoder Beginner Contest 043: *Upsolving*

Prof. Edson Alves - UnB/FGA

2020

Sumário

A - Children and Candies (ABC Edit)

Problema

There are N children in AtCoder Kindergarten. Mr. Evi will arrange the children in a line, then give 1 candy to the first child in the line, 2 candies to the second child, ..., N candies to the N-th child. How many candies will be necessary in total?

Entrada e saída

Constraints

- $1 \leq N \leq 100$

Input

Input is given from Standard Input in the following format:

N

Output

Print the necessary number of candies in total.

Exemplo de entradas e saídas

Entrada	Saída
3	6
10	55
1	1

Solução

- O problema consiste em determina a soma

$$S(N) = \sum_{i=1}^{N} = 1 + 2 + \ldots + N$$

- Como o máximo valor de N é relativamente pequeno ($N \leq 100$), é possível computar esta soma por meio de um laço, o que resulta em uma solução $O(N)$
- A soma $S(N)$, porém, pode ser computada por meio da expressão

$$S(N) = \frac{N(N+1)}{2}$$

- Esta expressão pode ser obtida por meio da soma dos termos de uma progressão aritmética de N termos, com $a_1 = 1$ e $a_N = N$
- Usando esta expressão, a complexidade da solução é reduzida para $O(1)$

Solução $O(1)$

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

using namespace std;

int main()
{
    int N;
    cin >> N;

    auto ans = N*(N + 1)/2;

    cout << ans << '\n';

    return 0;
}
```

B - Unhappy Hacking (ABC Edit)

Problema

Sig has built his own keyboard. Designed for ultimate simplicity, this keyboard only has 3 keys on it: the `'0'` key, the `'1'` key and the backspace key.

To begin with, he is using a plain text editor with this keyboard. This editor always displays one string (possibly empty). Just after the editor is launched, this string is empty. When each key on the keyboard is pressed, the following changes occur to the string:

Entrada e saída

- The `'0'` key: a letter `'0'` will be inserted to the right of the string.
- The `'1'` key: a letter `'1'` will be inserted to the right of the string.
- The backspace key: if the string is empty, nothing happens. Otherwise, the rightmost letter of the string is deleted.

Sig has launched the editor, and pressed these keys several times. You are given a string s, which is a record of his keystrokes in order. In this string, the letter `'0'` stands for the `'0'` key, the letter `'1'` stands for the `'1'` key and the letter `'B'` stands for the backspace key. What string is displayed in the editor now?

Entrada e saída

Constraints

- $1 \leq |s| \leq 10$ ($|s|$ denotes the length of s)
- s consists of the letters `'0'`, `'1'` e `'B'`.
- The correct answer is not and empty string.

Input

Input is given from Standard Input in the following format:

```
s
```

Output

Print the string displayed in the editor in the end.

Exemplo de entradas e saídas

Entrada	Saída
01B0	00
0BB1	1

Solução

- A solução do problema consiste em simular o comportamento descrito do editor
- Isto pode ser feito, em C++, por meio de uma variável do tipo string
- No caso dos caracteres `'0'` e `'1'`, o acréscimo à direita corresponde ao método `push_back()`, o qual tem complexidade $O(1)$
- Os caracteres `'B'` correspondem ao método `pop_back()`, que também tem complexidade $O(1)$
- É preciso atentar ao fato de que o método `push_back()` não deve ser invocado caso a string esteja vazia
- Assim, cada um dos caracteres é processado em $O(1)$, de modo que a solução tem complexidade $O(N)$

Solução $O(N)$

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

using namespace std;

string solve(const string& s)
{
    string res;

    for (auto c : s)
    {
        if (c == '0' or c == '1')
            res.push_back(c);
        else if (not res.empty())
            res.pop_back();
    }

    return res;
}

```

Solução $O(N)$

```
20 int main()
21 {
22     string s;
23     cin >> s;
24
25     auto ans = solve(s);
26
27     cout << ans << '\n';
28
29     return 0;
30 }
```

C - Be Together

Problema

Evi has N integers $a_1, a_2, \ldots, a_N$. His objective is to have N equal integers by transforming some of them.

He may transform each integer at most once. Transforming an integer x into another integer y costs him $(x-y)^2$ dollars. Even if $a_i = a_j$ $(i \neq j)$, he has to pay the cost separately for transforming each of them (See Sample 2).

Find the minimum total cost to achieve his objective.

Entrada e saída

Constraints

- $1 \leq N < 100$
- $-100 \leq a_i < 100$

Input

Input is given from Standard Input in the following format:

N

$a_1 \quad a_2 \quad \ldots \quad a_K$

Output

Print the minimum total cost to achieve Evi's objective.

Exemplo de entradas e saídas

Entrada	Saída
2 4 8	8
3 1 1 3	3
3 4 2 5	5

Solução

- Considere, sem perda de generalidade, que a_1 seja o menor dentre todos os números, e que a_N seja o maior dentre eles
- Seja $C(x)$ o custo para transformar todos os elementos $a_1, a_2, \ldots, a_N$
- Observe que $C(x) > C(a_1)$, se $x < a_1$
- Isto porque, estando x à esquerda de todos os números, se aproximar de a_1 reduz todos os custos individuais e, portanto, o custo total da transformação
- De forma análoga, $C(x) > C(a_N)$, se $x > a_N$

Solução

- Logo, o valor de x que minimiza o custo $C(x)$ deve pertencer ao intervalo $[a_1, a_N]$
- Como os limites do problema são relativamente pequenos, computar $C(x)$ para todos os x neste intervalo e registrar o menor valor de $C(x)$ teria complexidade $O(M)$ e veredito AC, onde

$$M = \max\{|a_1|, |a_2|, \ldots, |a_N|\}$$

- Contudo, este problema pode ser resolvido em $O(1)$
- Temos que

$$C(x) = (x - a_1)^2 + (x - a_2)^2 + \ldots + (x - a_N)^2$$

Solução

- Derivando em relação a x obtemos

$$C'(x) = 2(x - a_1) + 2(x - a_2) + \ldots + 2(x - a_N)$$

- A função $C(x)$ terá um ponto crítico em y se $C'(y) = 0$
- Esta igualdade nos dá

$$y = \frac{a_1 + a_2 + \ldots + a_N}{N}$$

- Assim, $C(x)$ será minimizada quando x for a média aritmética dos números $a_1, a_2, \ldots, a_N$
- Como os números devem ser todos inteiros após a transformação, o valor da média deve ser arredondado para o cálculo de $C(y)$, que será a resposta do problema

Solução $O(1)$

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

using namespace std;

int solve(int N, const vector<int>& xs)
{
    int m = (int) round(accumulate(xs.begin(), xs.end(), 0.0) / N);
    int res = 0;

    for (int i = 0; i < N; ++i)
        res += (xs[i] - m)*(xs[i] - m);

    return res;
}

```

Solução $O(1)$

```
16 int main()
17 {
18     int N;
19     cin >> N;
20
21     vector<int> xs(N);
22
23     for (int i = 0; i < N; ++i)
24         cin >> xs[i];
25
26     auto ans = solve(N, xs);
27
28     cout << ans << '\n';
29
30     return 0;
31 }
```

D – Iroha and a Grid

Problema

We have a large square grid with H rows and W columns. Iroha is now standing in the top-left cell. She will repeat going right or down to the adjacent cell, until she reaches the bottom-right cell.

However, she cannot enter the cells in the intersection of the bottom A rows and the leftmost B columns. (That is, there are $A \times B$ forbidden cells.) There is no restriction on entering the other cells.

Find the number of ways she can travel to the bottom-right cell.

Since this number can be extremely large, print the number modulo $10^9 + 7$.

Entrada e saída

Constraints

- $1 \leq H, W \leq 100,000$
- $1 \leq A < H$
- $1 \leq B < W$

Input

Input is given from Standard Input in the following format:

H W A B

Output

Print the number of ways she can travel to the bottom-right cell, modulo $10^9 + 7$.

Exemplo de entradas e saídas

Entrada	Saída
2 3 1 1	2
10 7 3 4	3570
100000 100000 99999 99999	1
100000 100000 44444 55555	738162020

Solução

- Este problema é uma variação de um problema combinatório conhecido
- O problema consiste em determinar o número de maneiras de, a partir do ponto $(1, 1)$, chegar ao ponto (m, n), podendo-se mover apenas uma unidade para à esquerda ou uma unidade para cima

Solução

- Todos os diferentes caminhos compartilham o fato que é necessário caminhar exatamente $m-1$ passos para a direita e $n-1$ passos para cima
- A diferença reside onde estes passos vão acontecer
- Uma vez determinado os pontos onde os passos para direita irão ocorrer, os para cima ficam determinados
- Assim, o número de maneiras distinas de se chegar a (m, n) a partir de $(1, 1)$ é dado por

$$ways(m, n) = \binom{(m-1)+(n-1)}{m-1}$$

Solução

- A diferença entre o problema combinatório conhecido e problema atual é a região bloqueada
- Outra diferença é Iroha está no canto superior esquerdo, então é útil posicionar o ponto $(1, 1)$ neste ponto de partida

Solução

- Assim, os caminhos são montado em duas etapas:
  1. Alcançar uma das posições à direita do bloqueio, na altura imediatamente anterior (marcados em azul na figura)
  2. Após se mover para a posição imediatamente abaixo (em verde, indicado pelas setas), se dirigir para o pontot (H, W)
- Como os caminhos de cada etapa são independentes, os produtos das quantidades de maneiras distintas de se chegar aos pontos desejados devem ser acumulados na resposta
- Para computar os coeficientes binomiais de maneira eficiente, deve-se precomputar os valores dos fatoriais e seus inversos multiplicativos

Solução

- Como $p = 10^9 + 7$ é primo, os inversos multiplicativos podem ser computado por meio do Teorema de Euler:

$$a^{p-2} \equiv a^{-1} \pmod{p}$$

- Este exponencial pode ser computada em $O(\log p)$, por meio de um algoritmo de divisão e conquista, denominado exponenciação rápida
- Considere o coeficiente $\binom{n}{m}$
- Neste problema, o maior valor possível para n é a soma $H + W$, de modo que só é preciso pré-computar os fatoriais até este valor
- Assim a complexidade da solução é $O((H + W) \log p)$

Solução $O((H+W)\log p)$

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

using namespace std;
using ll = long long;

ll fast_exp(ll a, ll n, ll m)
{
    ll res = 1, base = a;

    while (n)
    {
        if (n & 1)
            res = (res * base) % m;

        base = (base * base) % m;
        n >>= 1;
    }

    return res;
}

```

Solução $O((H+W)\log p)$

```
22 const ll MOD { 1000000007 };
23 const int MAX { 200010 };
24
25 ll fact[MAX], inv[MAX];
26
27 ll binom(int n, int m)
28 {
29     ll res = (fact[n] * inv[m]) % MOD;
30     return (res * inv[n - m]) % MOD;
31 }
32
33 void precomp()
34 {
35     fact[0] = inv[0] = 1;
36
37     for (int n = 1; n < MAX; ++n)
38         fact[n] = (n * fact[n - 1]) % MOD;
39
40     for (int n = 1; n < MAX; ++n)
41         inv[n] = fast_exp(fact[n], MOD - 2, MOD);
42 }
```

Solução $O((H+W)\log p)$

```
43
44 ll solve(int H, int W, int A, int B)
45 {
46     ll ans = 0;
47
48     for (int x = B + 1; x <= W; ++x)
49     {
50         auto m1 = x - 1, n1 = H - A - 1;
51         auto m2 = W - x + 1 - 1, n2 = A - 1;
52         auto ways = (binom(m1 + n1, m1) * binom(m2 + n2, m2)) % MOD;
53
54         ans = (ans + ways) % MOD;
55     }
56
57     return ans;
58 }
59
60 int main()
61 {
62     precomp();
63
```

Solução $O((H+W)\log p)$

```
64      int H, W, A, B;
65      cin >> H >> W >> A >> B;
66
67      auto ans = solve(H, W, A, B);
68
69      cout << ans << '\n';
70
71      return 0;
72  }
```

Referências

1. AtCoder Beginner Contest 043 - Problem A: Children and Candies (ABC Edit)
2. AtCoder Beginner Contest 043 - Problem B: Unhappy Hacking (ABC Edit)
3. AtCoder Beginner Contest 043 - Problem C: Be Together
4. AtCoder Beginner Contest 042 - Problem D: Iroha and a Grid