AtCoder

AtCoder Beginner Contest 042: *Upsolving*

Prof. Edson Alves - UnB/FGA

2020

Sumário

1. A – Iroha and Haiku
2. B – Iroha Loves Strings
3. C – Iroha's Obsession
4. D – Iroha and a Grid

A - Iroha and Haiku

Problema

Iroha loves Haiku. Haiku is a short form of Japanese poetry. A Haiku consists of three phrases with 5, 7 and 5 syllables, in this order.

To create a Haiku, Iroha has come up with three different phrases. These phrases have A, B and C syllables, respectively. Determine whether she can construct a Haiku by using each of the phrases once, in some order.

Entrada e saída

Constraints

- $1 \leq A, B, C \leq 10$

Input

Input is given from Standard Input in the following format:

A B C

Output

If it is possible to construct a Haiku by using each of the phrases once, print `'YES'` YES (case-sensitive). Otherwise, print `'NO'`.

Exemplo de entradas e saídas

Entrada	Saída
5 5 7	YES
7 7 5	NO

Solução

- O problema consiste em saber se os números A, B e C consistem na sequência 5, 7, 5, em alguma ordem
- Há um total de $3! = 6$ permutações possíveis dos valores de A, B e C
- Contudo, como o 5 se repete na sequência, basta verificar apenas três destas seis permutações
- Outra solução possível é armazenar os valores de A, B e C em um vetor, ordená-lo e, em seguida, compará-lo com a sequência 5, 5, 7
- Qualquer uma das duas abordagens resolve o problema em $O(1)$

Solução $O(1)$

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

using namespace std;

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

    int A, B, C;
    cin >> A >> B >> C;

    if ((A == 5 and B == 5 and C == 7) or
        (A == 5 and B == 7 and C == 5) or
        (A == 7 and B == 5 and C == 5))
            cout << "YES\n";
    else
            cout << "NO\n";

    return 0;
}
```

B - Iroha Loves Strings

Problema

Iroha has a sequence of N strings $S_1, S_2, \ldots, S_N$. The length of each string is L.

She will concatenate all of the strings in some order, to produce a long string.

Among all strings that she can produce in this way, find the lexicographically smallest one.

Here, a string $s = s_1 s_2 s_3 \ldots s_n$ is *lexicographically smaller* than another string $t = t_1 t_2 t_3 \ldots t_m$ if and only if one of the following holds:

- There exists an index ($1 \leq i \leq \min(n, m)$), such that $s_j = t_j$ for all indices j ($1 \leq j < i$) , and $s_i < t_i$.
- $s_i = t_i$ for all integers i ($1 \leq i \leq \min(n, m)$), and $n < m$.

Entrada e saída

Constraints

- $1 \leq N, L \leq 100$
- For each i, the length of S_i equals L.
- For each i, S_i consists of lowercase letters.

Entrada e saída

Input

Input is given from Standard Input in the following format:

$$
\begin{array}{ll}
N & L \\
S_1 & \\
S_2 & \\
\vdots & \\
S_N &
\end{array}
$$

Output

Print the lexicographically smallest string that Iroha can produce.

Exemplo de entradas e saídas

Entrada	Saída
3 3 dxx axx cxx	axxcxxdxx

Solução

- O operador de comparação padrão de C++ ordena as strings em ordem lexicográfica
- Como todas as strings da entrada tem um mesmo tamanho L, apenas o primeiro critério da ordenação lexicográfica apresentado no texto é relevante para este problema
- Este fato simplifica a solução
- Suponha uma ordenação $i_1, i_2, \ldots, i_N$ dos índices das strings e as strings $s_i = S_{i_1} S_{i_2} \ldots S_{i_N}$ resultante

Solução

- Se $S_{i_j} < S_{i_k}$ com $j > k$, a troca de posição entre os índices i_j e i_k produz uma string lexicograficamente menor do que s_i
- Assim, a menor string que pode ser produzida, segundo a ordem lexicográfica, é aquela cuja ordenação é tal que $S_{i_1} \leq S_{i_2} \leq \ldots \leq S_{i_N}$
- Esta ordenação pode ser produzida por meio da ordenação do vetor que armazena as strings dados na entrada
- Para este fim, a função `sort()` da STL do C++ é suficiente
- Esta solução tem complexidade $O(LN \log N)$

Solução $O(LN \log N)$

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

using namespace std;

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

    vector<string> xs(N);

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

    sort(xs.begin(), xs.end());

    for (int i = 0; i < N; ++i)
        cout << xs[i] << (i + 1 == N ? "\n" : "");

    return 0;
}
```

C - Iroha's Obsession

Problema

Iroha is very particular about numbers. There are K digits that she dislikes: $D_1, D_2, \ldots, D_K$.

She is shopping, and now paying at the cashier. Her total is N yen (the currency of Japan), thus she has to hand at least N yen to the cashier (and possibly receive the change).

However, as mentioned before, she is very particular about numbers. When she hands money to the cashier, the decimal notation of the amount must not contain any digits that she dislikes. Under this condition, she will hand the minimum amount of money.

Find the amount of money that she will hand to the cashier.

Entrada e saída

Constraints

- $1 \leq N < 10000$
- $1 \leq K < 10$
- $0 \leq D_1 < D_2 < \ldots < D_K \leq 9$
- $\{D_1, D_2, \ldots, D_K\} \neq \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Input

Input is given from Standard Input in the following format:

N K
D_1 D_2 ... D_K

Output

Print the amount of money that Iroha will hand to the cashier.

Exemplo de entradas e saídas

Entrada	Saída
1000 8 1 3 4 5 6 7 8 9	2000
9999 1 0	9999

Solução

- Seja D o menor dígito que Iroha gosta e k o número de dígitos decimais de N
- O número $x = DD \ldots D$, formado por $k + 1$ dígitos D, é um número válido para Iroha e maior do que N
- Assim, é possível procurar, um a um, pelo menor número válido, pois ele será maior ou igual a N inferior a $10N$
- Cada número $y \in [N, 10N)$ pode ser verificado em $O(\log y)$
- Assim a solução de busca completa teria complexidade $O(N \log N)$

Solução $O(N \log N)$

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

using namespace std;

bitset<10> ds;

bool is_valid(int n)
{
    while (n)
    {
        auto d = n % 10;
        n /= 10;

        if (ds[d])
            return false;
    }

    return true;
}

```

Solução $O(N \log N)$

```
21 int solve(int N)
22 {
23     while (not is_valid(N))
24         ++N;
25
26     return N;
27 }
28
29 int main() {
30     int N, K, d;
31     cin >> N >> K;
32
33     while (K--) {
34         cin >> d;
35         ds[d] = true;
36     }
37
38     cout << solve(N) << '\n';
39
40     return 0;
41 }
```

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

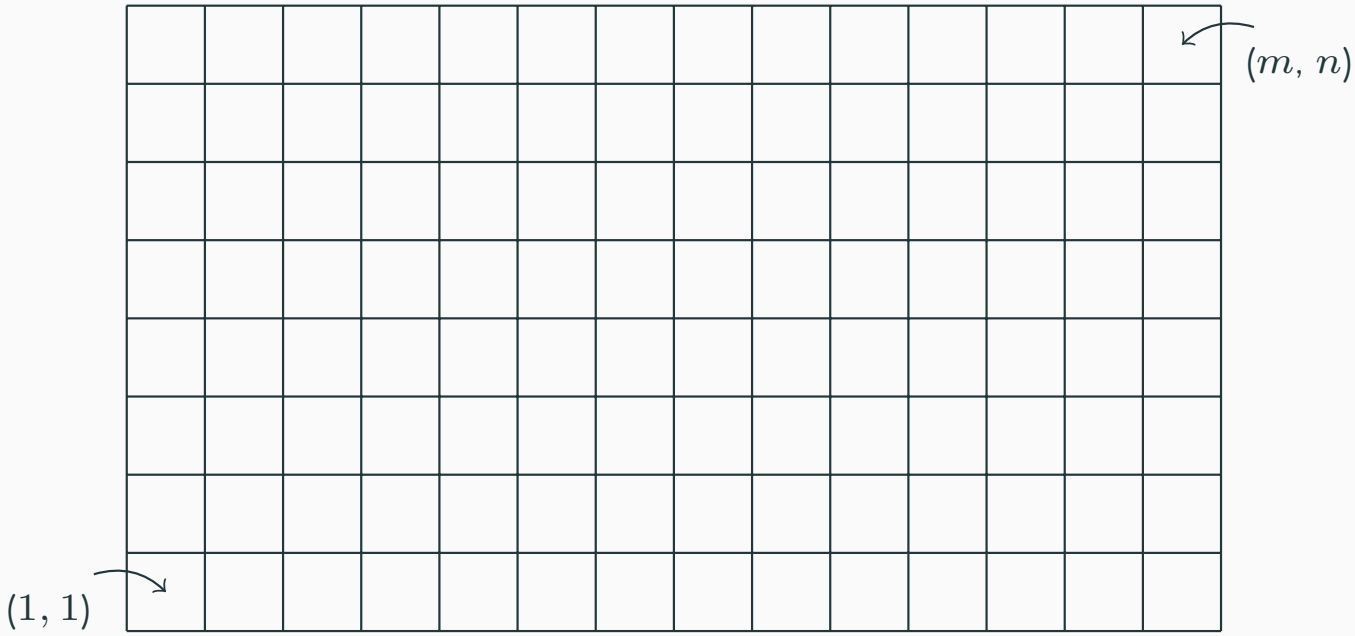
Print the number of ways she can travel to the bottom-right cell, modulo $10^9 + 7$.

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



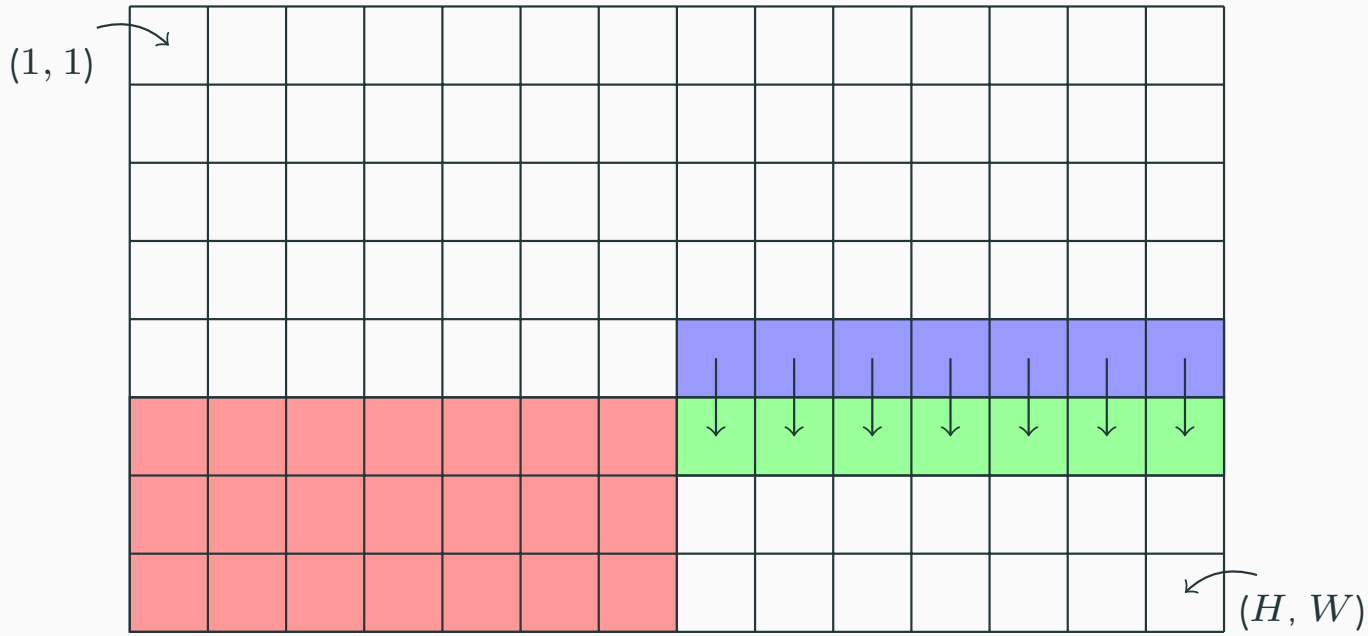
Solução

- 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 042 - Problem A: Iroha and Haiku (ABC Edition)
2. AtCoder Beginner Contest 042 - Problem B: Iroha Loves Strings (ABC Edition)
3. AtCoder Beginner Contest 042 - Problem C: Iroha's Obsession
4. AtCoder Beginner Contest 042 - Problem D: Iroha and a Grid