

AtCoder

AtCoder Beginner Contest 045: *Upsolving*

Prof. Edson Alves - UnB/FGA

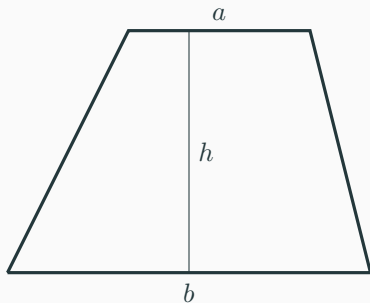
2020

1. A – Trapezoids
2. B – Card Game for Three (ABC Edit)
3. C – Many Formulas
4. D – Snuke's Coloring

A - Trapezoids

Problema

You are given a trapezoid. The lengths of its upper base, lower base, and height are a , b , and h , respectively.



An example of a trapezoid

Find the area of this trapezoid.

Constraints

- $1 \leq a \leq 100$
- $1 \leq b \leq 100$
- $1 \leq h \leq 100$
- All input values are integers.
- h is even.

Input

Input is given from Standard Input in the following format:

a

b

h

Output

Print the area of the given trapezoid. It is guaranteed that the area is an integer.

Exemplo de entradas e saídas

Entrada

3

4

2

4

4

4

Saída

7

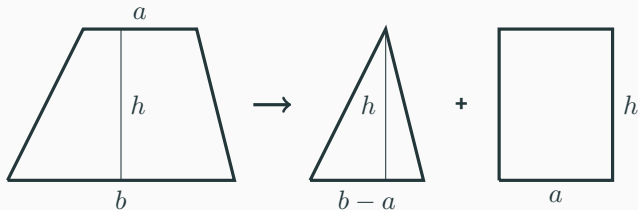
16

Solução

- O problema consiste na aplicação direta da fórmula da área do trapézio:

$$A = \frac{(a + b)h}{2}$$

- Esta fórmula pode ser deduzida a partir da observação que o trapézio pode ser decomposto em um triângulo T e um retângulo R



- Assim,

$$\begin{aligned} A &= A_T + A_R \\ &= \frac{(b-a)h}{2} + ah \\ &= \frac{(a+b)h}{2} \end{aligned}$$

- A aplicação direta da fórmula resulta em uma solução $O(1)$ para o problema

Solução $O(1)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int main()
6 {
7     int a, b, h;
8     cin >> a >> b >> h;
9
10    auto ans = (a + b)*h/2;
11
12    cout << ans << '\n';
13
14    return 0;
15 }
```

B – Card Game for Three (ABC Edit)

Problema

Alice, Bob and Charlie are playing *Card Game for Three*, as below:

- At first, each of the three players has a deck consisting of some number of cards. Each card has a letter 'a', 'b' or 'c' written on it. The orders of the cards in the decks cannot be rearranged.
- The players take turns. Alice goes first.
- If the current player's deck contains at least one card, discard the top card in the deck. Then, the player whose name begins with the letter on the discarded card, takes the next turn. (For example, if the card says 'a', Alice takes the next turn.)
- If the current player's deck is empty, the game ends and the current player wins the game.

Problema

You are given the initial decks of the players. More specifically, you are given three strings S_A , S_B and S_C . The i -th ($1 \leq i \leq |S_A|$) letter in S_A is the letter on the i -th card in Alice's initial deck. S_B and S_C describes Bob's and Charlie's initial decks in the same way.

Determine the winner of the game.

Constraints

- $1 \leq |S_A| \leq 100$
- $1 \leq |S_B| \leq 100$
- $1 \leq |S_C| \leq 100$
- Each letter in S_A, S_B, S_C is 'a', 'b' or 'c'.

Input

Input is given from Standard Input in the following format:

 S_A S_B S_C

Output

If Alice will win, print 'A'. If Bob will win, print 'B'. If Charlie will win, print 'C'.

Exemplo de entradas e saídas

Entrada

aca

accc

ca

abcb

aacb

bccc

Saída

A

C

Solução

- A solução do problema consiste em simular o comportamento descrito do editor
- Seja N o tamanho de uma string S
- A remoção de um caractere do início de uma string não faz parte da API do C++, e mesmo que seja implementada ela terá complexidade $O(N)$, por conta do deslocamento dos caracteres
- Já o método `pop_back()` permite a exclusão do último caractere em $O(1)$
- Assim, se as strings forem invertidas (o método `reverse()` pode ser utilizado para este fim), a simulação fica simplificada e mais eficiente
- Portanto a solução tem complexidade $O(M)$, onde

$$M = \max\{S_A, S_B, S_C\}$$

Solução $O(M)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 char solve(string& sa, string& sb, string& sc)
6 {
7     vector<string> xs { sa, sb, sc };
8     int next = 0;
9
10    for (int i = 0; i < 3; ++i)
11        reverse(xs[i].begin(), xs[i].end());
12
13    while (not xs[next].empty())
14    {
15        auto c = xs[next].back();
16        xs[next].pop_back();
17        next = c - 'a';
18    }
19
20    return char(next + 'A');
21 }
```

Solução $O(M)$

```
22
23 int main()
24 {
25     ios::sync_with_stdio(false);
26
27     string sa, sb, sc;
28     cin >> sa >> sb >> sc;
29
30     auto ans = solve(sa, sb, sc);
31
32     cout << ans << '\n';
33
34     return 0;
35 }
```

C – Many Formulas

Problema

You are given a string S consisting of digits between '1' and '9', inclusive. You can insert the letter '+' into some of the positions (possibly none) between two letters in this string. Here, '+' must not occur consecutively after insertion.

All strings that can be obtained in this way can be evaluated as formulas. Evaluate all possible formulas, and print the sum of the results.

Constraints

- $1 \leq |S| \leq 10$
- All letters in S are digits between '1' and '9', inclusive.

Input

Input is given from Standard Input in the following format:

S

Output

Print the sum of the evaluated value over all possible formulas.

Exemplo de entradas e saídas

Entrada

125

9999999999

Saída

176

12656242944

Solução

- Como a string tem, no máximo, 10 caracteres, é possível gerar e avaliar todas as expressões possíveis
- Seja $N = |S|$
- Há $N - 1$ posições onde podem ser inseridos símbolos '+': antes de qualquer caractere s_i , com $i \in [2, N]$:

$$s_1 \quad - \quad s_2 \quad - \quad \dots \quad - \quad s_N$$

- Assim, são 2^{N-1} expressões distintas, que podem ser representadas por meio de todos os inteiros de $N - 1$ bits, onde um bit ligado indica a presença de um '+' na posição indicada
- Cada expressão pode ser avaliada em $O(N)$, logo a solução tem complexidade $O(N2^N)$

Solução $O(N2^N)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5
6 ll solve(const string& s)
7 {
8     ll ans = 0;
9
10    for (int p = 0; p < (1 << (s.size() - 1)); ++p)
11    {
12        ll res = 0, x = 0;
13
14        for (size_t i = 0; i < s.size(); ++i)
15        {
16            x *= 10;
17            x += (s[i] - '0');
18
19            if (p & (1 << i) or i == s.size() - 1)
20            {
21                res += x;
```

Solução $O(N2^N)$

```
22         x = 0;
23     }
24 }
25
26     ans += res;
27 }
28
29     return ans;
30 }
31
32 int main()
33 {
34     string s;
35     cin >> s;
36
37     auto ans = solve(s);
38
39     cout << ans << '\n';
40
41     return 0;
42 }
```

D - Snuke's Coloring

Problema

We have a grid with H rows and W columns. At first, all cells were painted white.

Snuke painted N of these cells. The i -th ($1 \leq i \leq N$) cell he painted is the cell at the a_i -th row and b_i -th column.

Compute the following:

- For each integer j ($0 \leq j \leq 9$), how many subrectangles of size 3×3 of the grid contains exactly j black cells, after Snuke painted N cells?

Constraints

- $3 \leq H \leq 10^9$
- $3 \leq W \leq 10^9$
- $0 \leq N \leq \min(10^5, H \times W)$
- $1 \leq a_i \leq H$ ($1 \leq i \leq N$)
- $1 \leq b_i \leq W$ ($1 \leq i \leq N$)
- $(a_i, b_i) \neq (a_j, b_j)$ ($i \neq j$)

Input

Input is given from Standard Input in the following format:

```
 $H$      $W$      $N$   
 $a_1$      $b_1$   
 $\vdots$   
 $a_N$      $b_N$ 
```

Output

Print 10 lines. The $(j + 1)$ -th ($0 \leq j \leq 9$) line should contain the number of the subrectangles of size 3×3 of the grid that contains exactly j black cells.

Exemplo de entradas e saídas

Entrada

4 5 8

1 1

1 4

1 5

2 3

3 1

3 2

3 4

4 4

Saída

0

0

0

2

4

0

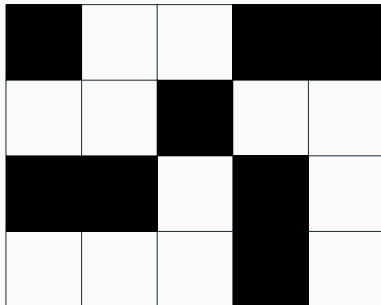
0

0

0

0

Exemplo de entradas e saídas



- Há um total de $(H - 2)(W - 2)$ quadrados 3×3 a serem avaliados
- Como $H, W \leq 10^9$, a verificação individual de cada um destes quadrados leva a um veredito TLE
- Contudo, o número total de pontos pintados N é menor ou igual a 10^5
- Cada ponto pertence a um total de 9 destes quadrados
- Assim, é possível resolver o problema verificando, no máximo, 9×10^5 quadrados, pois todos os demais certamente não terão nenhum ponto pintado

- Considere que cada quadrado seja caracterizado pelo seu canto superior esquerdo
- Cada ponto pintado (x, y) pertence a um dos 9 quadrados cujo canto superior esquerdo tem coordenadas $(x - d_x, y - d_y)$, com $0 \leq d_x, d_y \leq 2$
- Um dicionário deve registrar, para cada possível canto superior esquerdo, quantos pontos pintados fazem parte de seu quadrado associado
- Deve-se atentar que o número de quadrados sem nenhum ponto pintado é a diferença entre o total e os que tem ao menos um ponto pintado
- Esta solução tem complexidade $O(N \log N)$

Solução $O(N \log N)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5 using ii = pair<ll, ll>;
6
7 vector<ll> solve(ll H, ll W, const vector<ii>& ps)
8 {
9     map<ii, int> hs;
10
11     for (auto p : ps)
12     {
13         auto x = p.first, y = p.second;
14
15         for (int i = -2; i <= 0; ++i)
16         {
17             for (int j = -2; j <= 0; ++j)
18             {
19                 auto u = x + i, v = y + j;
20
```

Solução $O(N \log N)$

```
21         if (1 <= u and u <= H - 2 and 1 <= v and v <= W - 2)
22             ++hs[ii(u, v)];
23     }
24 }
25 }
26
27 vector<ll> ans(10, 0);
28
29 for (auto p : hs)
30     ans[p.second]++;
31
32 ans[0] += (H - 2)*(W - 2) - hs.size();
33
34 return ans;
35 }
36
37 int main()
38 {
39     ios::sync_with_stdio(false);
40
```

Solução $O(N \log N)$

```
41  int H, W, N;
42  cin >> H >> W >> N;
43
44  vector<ii> xs(N);
45
46  for (int i = 0; i < N; ++i)
47      cin >> xs[i].first >> xs[i].second;
48
49  auto ans = solve(H, W, xs);
50
51  for (auto x : ans)
52      cout << x << '\n';
53
54  return 0;
55 }
```

1. [AtCoder Beginner Contest 045 – Problem A: Trapezoids](#)
2. [AtCoder Beginner Contest 045 – Problem B: Game Card for Three \(ABC Edit\)](#)
3. [AtCoder Beginner Contest 045 – Problem C: Many Formulas](#)
4. [AtCoder Beginner Contest 045 – Problem D: Snuke's Coloring](#)