AtCoder

AtCoder Beginner Contest 045: *Upsolving*

Prof. Edson Alves - UnB/FGA

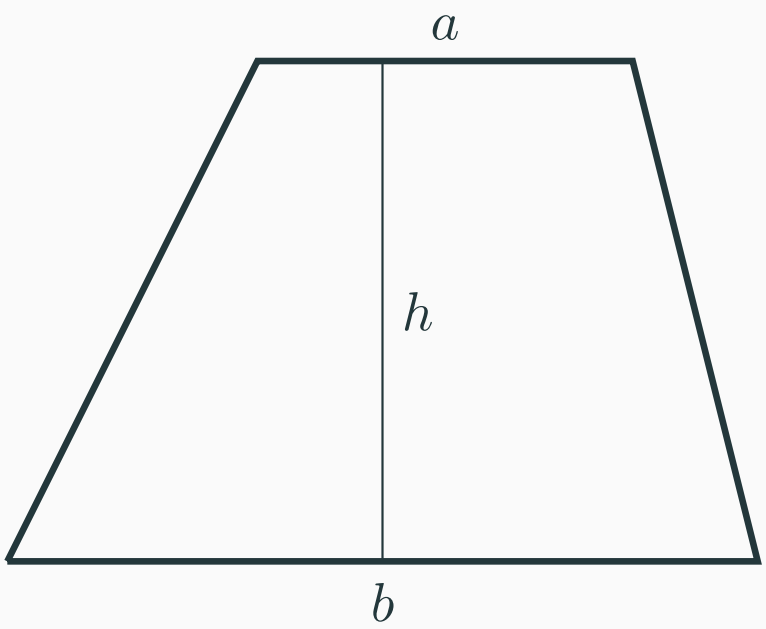
2020

Sumário

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.

Entrada e saída

Constraints

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

Entrada e saída

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

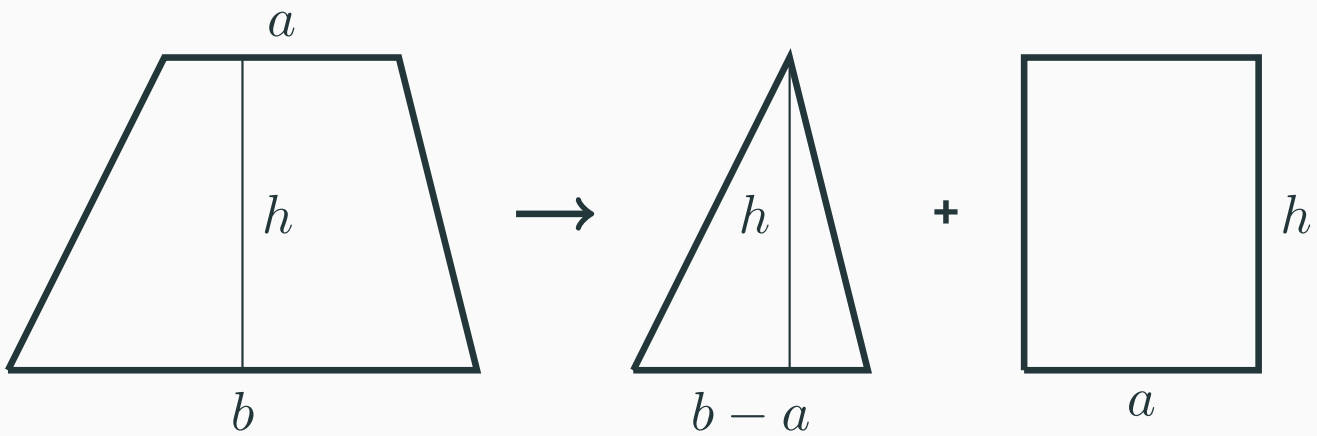
Entrada	Saída
3 4 2	7
4 4 4	16

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



Solução

- 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 $O(1)$

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

using namespace std;

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

    auto ans = (a + b)*h/2;

    cout << ans << '\n';

    return 0;
}
```

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.

Entrada e saída

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'`.

Entrada e saída

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	Saída
aca accc ca	A
abcb aacb bccc	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 $O(M)$

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

using namespace std;

char solve(string& sa, string& sb, string& sc)
{
    vector<string> xs { sa, sb, sc };
    int next = 0;

    for (int i = 0; i < 3; ++i)
        reverse(xs[i].begin(), xs[i].end());

    while (not xs[next].empty())
    {
        auto c = xs[next].back();
        xs[next].pop_back();
        next = c - 'a';
    }

    return char(next + 'A');
}
```

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.

Entrada e saída

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	Saída
125	176
9999999999	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 $i \in [2, N]$:

$$s_1 \quad \_ \quad s_2 \quad \_ \quad \cdots \quad \_ \quad s_N$$

- 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 $O(N2^N)$

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

using namespace std;
using ll = long long;

ll solve(const string& s)
{
    ll ans = 0;

    for (int p = 0; p < (1 << (s.size() - 1)); ++p)
    {
        ll res = 0, x = 0;

        for (size_t i = 0; i < s.size(); ++i)
        {
            x *= 10;
            x += (s[i] - '0');

            if (p & (1 << i) or i == s.size() - 1)
            {
                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?

Entrada e saída

Constraints

- $2 \leq |s| \leq 10^5$
- s consists of lowercase letters.

Partial Score

- 200 points will be awarded for passing the test set satisfying $2 \leq N \leq 100$.

Entrada e saída

Input

Input is given from Standard Input in the following format:

s

Output

If there exists no unbalanced substring of s, print `'-1 -1'`.

If there exists an unbalanced substring of s, let one such substring be $s_a s_{a+1} \ldots s_b$ $(1 \leq a < b \leq |s|)$, and print `'`$a\ b$`'`. If there exists more than one such substring, any of them will be accepted.

Exemplo de entradas e saídas

Entrada	Saída
`needed`	`2 5`
`atcoder`	`-1 -1`

Solução

- Seja N o tamanho da string s
- Há $O(N^2)$ substrings de s, e a verificação de cada substring pode ser feita em $O(N)$, o que levaria a uma solução $O(N^3)$, que receberia veredito TLE, pois $N \leq 10^5$
- É preciso, portanto, encontrar uma forma eficiente de se processar as substrings e também de reduzir o número de substrings a serem verificadas

Solução

- Como qualquer string desbalaceada pode ser dada como resposta do problema, o princípio da casa dos pombos pode reduzir drasticamente o número de substrings a serem verificadas
- Uma string desbalanceada tem mais da metade de seus caracteres repetidos
- Se N é par e s é desbalanceada, ela possui ao menos um par de caracteres consecutivos iguais
- Como uma string com $N = 2$ com dois caracteres iguais é desbalanceada, é suficiente indicar estes caracteres ao invés da string inteira

Solução

- Se N é ímpar, é possível que s seja desbalanceada e que não tenha dois caracteres consecutivos iguais
- Isto só ocorre se o número de repetições é igual a $(N+1)/2$ e o caractere repetido aparece em todos os índices ímpares de s:

$$s = cs_2cs_4 \dots s_{N-1}c$$

- Porém, se $N = 3$ e s é da forma cs_2c, ela também será desbalanceada e poderá ser utilizada como resposta ao invés de s
- Portanto, basta checar apenas estes dois casos, gerando uma solução $O(N)$

Solução $O(|s|)$

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

using namespace std;

pair<int, int> solve(const string& s)
{
    int N = (int) s.size();

    // Caso 1: dois caracteres consecutivos
    for (int i = 1; i < N; ++i)
        if (s[i] == s[i - 1])
            return { i, i + 1 };

    // Caso 2: três caracteres consecutivos, extremos iguais
    for (int i = 2; i < N; ++i)
        if (s[i] == s[i - 2])
            return { i - 1, i + 1 };

    return { -1, -1 };
}

```

Solução $O(|s|)$

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

Referências

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