Strings

Representação de strings: problemas resolvidos

Prof. Edson Alves

2018

Faculdade UnB Gama

Sumário

UVA 10252 – Commom Permutation

Problema

Given two strings of lowercase letters, a and b, print the longest string x of lowercase letters such that there is a permutation of x that is a subsequence of a and there is a permutation of x that is a subsequence of b

Entrada e saída

Input

Input file contains several lines of input. Consecutive two lines make a set of input. That means in the input file line 1 and 2 is a set of input, line 3 and 4 is a set of input and so on. The first line of a pair contains a and the second contains b. Each string is on a separate line and consists of at most 1000 lowercase letters.

Output

For each set of input, output a line containing x. If several x satisfy the criteria above, choose the first one in alphabetical order.

Exemplo de entradas e saídas

Exemplo de Entrada

```
pretty
women
walking
down
the
street
```

Exemplo de Saída

```
e
nw
et
```

Solução

- Uma string s de tamanho $|s| = n$ tem 2^n subsequências, não necessariamente distintas
- Avaliar todas estas subsequências leva ao TLE
- Porém, se as strings forem visualizadas como conjuntos com elementos repetidos, a maior subsequência comum entre ambas strings é a interseção entre estes conjuntos
- Esta interseção pode ser obtida por meio da função `set_intersection()` da STL
- Por fim, como existem diferentes permutações desta interseção, é preciso ordená-la para que se torne a primeira, na ordem lexicográfica

Solução AC

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

using namespace std;

string solve(string& a, string& b)
{
    sort(a.begin(), a.end());
    sort(b.begin(), b.end());

    string ans;
    set_intersection(a.begin(), a.end(), b.begin(), b.end(),
        back_inserter(ans));

    return ans;
}

```

Solução AC

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

    string a, b;

    while (getline(cin, a), getline(cin, b))
    {
        auto ans = solve(a, b);

        cout << ans << endl;
    }

    return 0;
}
```

Codeforces Beta Round #5 – Problem B: Center Alignment

Problema

Almost every text editor has a built-in function of center text alignment. The developers of the popular in Berland text editor «Textpad» decided to introduce this functionality into the fourth release of the product.

You are to implement the alignment in the shortest possible time. Good luck!

Entrada e saída

Input

The input file consists of one or more lines, each of the lines contains Latin letters, digits and/or spaces. The lines cannot start or end with a space. It is guaranteed that at least one of the lines has positive length. The length of each line and the total amount of the lines do not exceed 1000.

Output

Format the given text, aligning it center. Frame the whole text with characters «*» of the minimum size. If a line cannot be aligned perfectly (for example, the line has even length, while the width of the block is uneven), you should place such lines rounding down the distance to the left or to the right edge and bringing them closer left or right alternatively (you should start with bringing left). Study the sample tests carefully to understand the output format better.

Exemplo de entradas e saídas

Sample Input

```
This  is


Codeforces
Beta
Round
5
```

Sample Output

```
************
* This  is *
*          *
*Codeforces*
*   Beta   *
*  Round   *
*     5    *
************
```

Solução com complexidade $O(n)$

- Primeiramente, é preciso determinar o tamanho da maior linha M, o qual irá determinar a largura do quadro
- Inicialmente, deve ser impressa uma linha com $M + 2$ caracteres '*'
- Para cada linha L, é preciso confrontar seu tamanho com M
- Se for menor, a diferença deve ser dividida igualmente entre os lados esquerdo e direito
- No caso de uma diferença ímpar, o caractere restante deve ser distribuído alternadamente, inicialmente à direita
- Uma variável auxiliar *padding* pode ser usada para manter esta alternância
- Finalmente, deve ser impressa uma nova linha com $M + 2$ caracteres '*'

Solução AC

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

using namespace std;

string solve(const vector<string>& lines)
{
    size_t max_size = 0, padding = 0;

    for (const auto& line : lines)
        max_size = max(max_size, line.size());

    ostringstream os;

    for (size_t i = 0; i < max_size + 2; ++i)
        os << '*' << (i == max_size + 1 ? "\n" : "");

    for (const auto& line : lines)
    {
        auto size = line.size();
        auto diff = max_size - size;

```

Solução AC

```
22          int left = diff / 2;
23          int right = diff / 2;
24
25          if (diff & 1) {
26              left += padding;
27              right += 1 - padding;
28              padding = 1 - padding;
29          }
30
31          os << '*';
32
33          while (left--)
34              os << ' ';
35
36          os << line;
37
38          while (right--)
39              os << ' ';
40
41          os << "*\n";
42      }
```

Solução AC

```
43
44      for (size_t i = 0; i < max_size + 2; ++i)
45          os << '*' << (i == max_size + 1 ? "\n" : "");
46
47      return os.str();
48  }
49
50  int main()
51  {
52      vector<string> lines;
53      string line;
54
55      while (getline(cin, line))
56          lines.push_back(line);
57
58      auto ans = solve(lines);
59
60      cout << ans;
61
62      return 0;
63  }
```

AtCoder Beginner Contest 103 – Problem C: /\/\/\/

Problema

A sequence $a_1, a_2, \ldots, a_n$ is said to be /\/\/\/ when the following conditions are satisfied:

- For each $i = 1, 2, \ldots, n-2, a_i = a_{i+2}$.
- Exactly two different numbers appear in the sequence.

You are given a sequence $v_1, v_2, \ldots, v_n$ whose length is even. We would like to make this sequence /\/\/\/ by replacing some of its elements. Find the minimum number of elements that needs to be replaced.

Entrada e saída

Constraints

- $2 \leq n \leq 10^5$
- n is even.
- $1 \leq v_i \leq 10^5$
- v_i is an integer.

Input

Input is given from Standard Input in the following format:

$$n$$

$$v_1 \ v_2 \ \ldots \ v_n$$

Output

Print the minimum number of elements that needs to be replaced.

Exemplo de entradas e saídas

Exemplo de Entrada	Exemplo de Saída
4 3 1 3 2	1
6 105 119 105 119 105 119	0
4 1 1 1 1	2

Solução

- A solução consiste em diversas etapas
- A primeira delas é construir o histograma dos elementos que estão nos índices ímpares e dos elementos que estão nos índices pares
- Aqui há um *corner case*: se todos os elementos são iguais, é preciso inserir um valor sentinela no histograma, com número de ocorrências iguais a zero
- A etapa seguinte é computar, para cada elemento distinto do histograma, o custo de tornar todos os valores da subsequência que ele pertence iguais a ele
- Estes custos devem ser armazenados em um vetor de pares, onde o primeiro elemento é o custo e o segundo é o número que preencherá todas as posições

Solução

- Feito isso para ambas subsequências, estes pares devem ser ordenados, do menor para o maior custo
- Por fim, se os elementos de menores custos de ambas subsequências forem distinto, a resposta será a soma destes custos
- Se forem iguais, é preciso ver o que é mais barato: trocar o segundo mais barato da sequência de índices ímpares e o mais barato da sequência dos pares ou o contrário
- Esta solução tem complexidade $O(N \log N)$, devido à construção do histograma e da ordenação dos custos

Solução AC com complexidade $O(N \log N)$

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

using namespace std;
using ii = pair<int, int>;

int solve(const vector<int>& vs, int N)
{
    map<int, int> hist[2];
    vector<ii> cost[2];

    for (int k = 0; k < 2; ++k)
    {
        for (int i = k; i < N; i += 2)
            ++hist[k][vs[i]];

        // Corner case: caso todos os elementos sejam iguais, devemos
        // ter uma segunda opção de escolha
        hist[k][100001] = 0;

```

Solução AC com complexidade $O(N \log N)$

```
20          for (const auto& p : hist[k])
21          {
22              int v, n;
23              tie(v, n) = p;
24
25              cost[k].push_back(ii(N/2 - n, v));
26          }
27
28          sort(cost[k].begin(), cost[k].end());
29      }
30
31      enum { ODD = 0, EVEN = 1 };
32
33      auto ans = cost[ODD][0].second == cost[EVEN][0].second ?
34          min(cost[ODD][0].first + cost[EVEN][1].first,
35              cost[ODD][1].first + cost[EVEN][0].first)
36          : cost[ODD][0].first + cost[EVEN][0].first;
37
38      return ans;
39  }
40
```

Solução AC com complexidade $O(N \log N)$

```
41 int main()
42 {
43     ios::sync_with_stdio(false);
44
45     int N;
46     cin >> N;
47
48     vector<int> vs(N);
49
50     for (int i = 0; i < N; ++i)
51         cin >> vs[i];
52
53     auto ans = solve(vs, N);
54
55     cout << ans << '\n';
56
57     return 0;
58 }
```

URI 1242 – Ácido Ribonucleico Alienígena

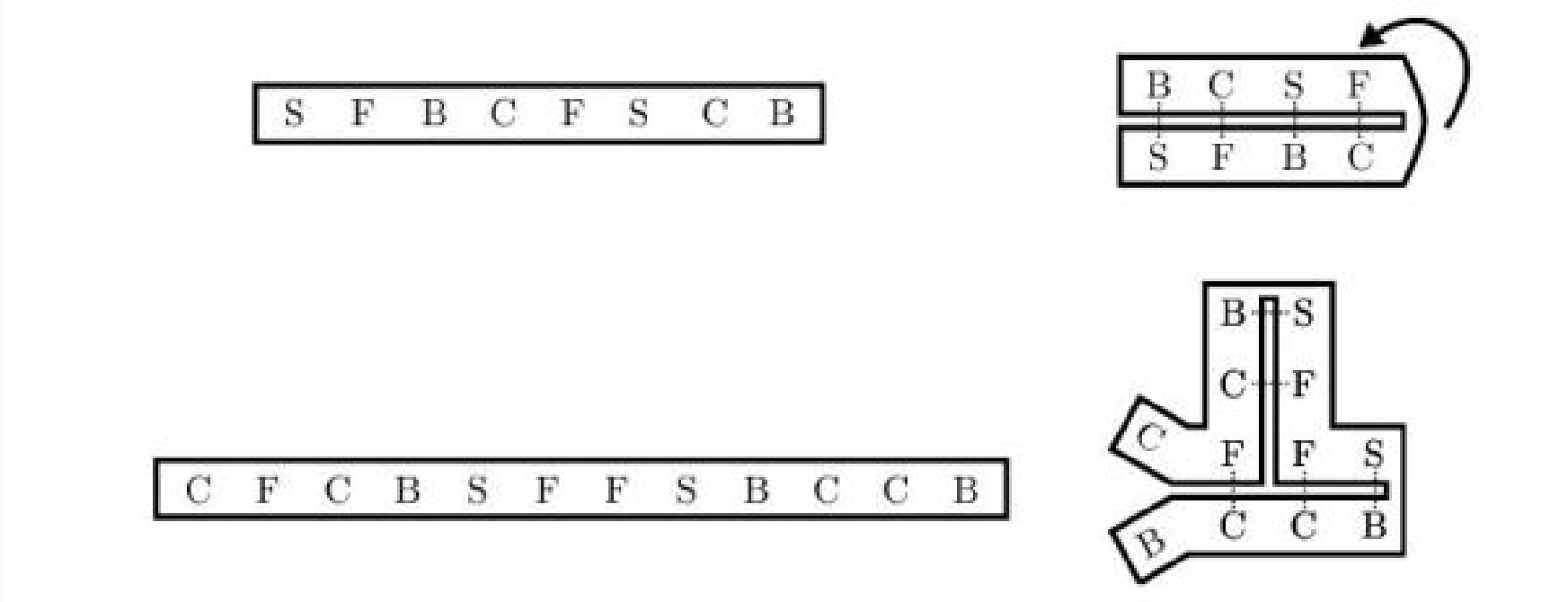
Problema

Foi descoberta uma espécie alienígena de ácido ribonucleico (popularmente conhecido como RNA). Os cientistas, por falta de criatividade, batizaram a descoberta de ácido ribonucleico alienígena (RNAA). Similar ao RNA que conhecemos, o RNAA é uma fita composta de várias bases. As bases são B C F S e podem ligar-se em pares. Os únicos pares possíveis são entre as bases B e S e as bases C e F. Enquanto está ativo, o RNAA dobra vários intervalos da fita sobre si mesma, realizando ligações entre suas bases. Os cientistas perceberam que:

- Quando um intervalo da fita de RNAA se dobra, todas as bases neste intervalo se ligam com suas bases correspondentes;
- Cada base pode se ligar a apenas uma outra base;
- As dobras ocorrem de forma a maximizar o número de ligações feitas sobre fitas;

Problema

As figuras abaixo ilustram dobras e ligacões feitas sobre fitas.



Sua tarefa será, dada a descrição de uma tira de RNAA, determinar quantas ligações serão realizadas entre suas bases se a tira ficar ativa.

Entrada e saída

Entrada

A entrada é composta por diversos casos de teste e termina com EOF. Cada caso de teste possui uma linha descrevendo a sequência de bases da fita de RNAA. Uma fita de RNAA na entrada contém pelo menos 1 e no máximo 300 bases. Não existem espaços entre bases de uma fita da entrada. As bases são 'B', 'C', 'F' e 'S'.

Saída

Para cada instância imprima uma linha contendo o número total de ligações que ocorre quando a fita descrita é ativada.

Exemplo de entradas e saídas

Exemplo de Entrada	Exemplo de Saída
SBC	1
FCC	1
SFBC	0
SFBCFSCB	4
CFCBSFFSBCCB	5

Solução com complexidade $O(N^2)$

- A estratégia para a solução do problema é utilizar duas pilhas
- A primeira deve ser preenchida com todos os elementos da string dada, na ordem da entrada
- A segunda guardará os elementos a serem pareadas, e inicialmente estará vazia
- Em seguida, todos os elementos da primeira fila devem ser processados, do topo à base
- Se a segunda pilha estiver vazia, o elemento processado vai para o seu topo

Solução com complexidade $O(N^2)$

- Caso contrário, tenta-se o pareamento do elemento processado com o topo da pilha
- Se é um pareamento válido, a resposta é incrementada e ambos elementos descartados
- Caso contrário, o elemento processado vai para segunda pilha
- Ao final do processamento, se houve ao menos um pareamento válido, o processamento deve recomeçar, tendo as filas trocadas de posição
- Esta solução é quadrática no número de elementos da string dada

Solução com complexidade $O(N^2)$

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

using namespace std;

const set<string> matches { "SB", "BS", "CF", "FC" };

int solve(const string& base)
{
    stack<char> a, b;
    stack<char> &prev = a, &next = b;

    for (auto c : base)
        a.push(c);

    bool match = true;
    int ans = 0;

    while (match)
    {
        match = false;

```

Solução com complexidade $O(N^2)$

```
22          while (not prev.empty())
23          {
24              auto c = prev.top(); prev.pop();
25
26              if (next.empty())
27              {
28                  next.push(c);
29                  continue;
30              }
31
32              string comb { c, next.top() };
33
34              if (matches.count(comb))
35              {
36                  match = true;
37                  ++ans;
38                  next.pop();
39              } else
40                  next.push(c);
41          }
42
```

Solução com complexidade $O(N^2)$

```
43          swap(next, prev);
44      }
45
46      return ans;
47  }
48
49  int main()
50  {
51      ios::sync_with_stdio(false);
52
53      string base;
54
55      while (getline(cin, base))
56      {
57          auto ans = solve(base);
58
59          cout << ans << '\n';
60      }
61
62      return 0;
63  }
```

Referências

1. UVA 10252 – Common Permutation
2. Codeforces Beta Round #5 – Problem B: Center Aligment
3. AtCoder Beginner Contest 103 – Problem C: /\/\/\/
4. URI 1242 – Ácido Ribonucleico Alienígena