AtCoder Beginner Contest 110

Problema C: *String Transformation*

Prof. Edson Alves – UnB/FGA

Problema

You are given strings S and T consisting of lowercase English letters.

You can perform the following operation on S any number of times:

Operation: Choose two distinct lowercase English letters c_1 and c_2, then replace every occurrence of c_1 with c_2, and every occurrence of c_2 with c_1.

Determine if S and T can be made equal by performing the operation zero or more times.

Entrada e saída

Constraints

- $1 \leq |S| \leq 2 \times 10^5$
- $|S| = |T|$
- S and T consists of lowercase English letters.

Input

Input is given from Standard Input in the following format:

$$S$$
$$T$$

Output

If S and T can be made equal, print `'Yes'`; otherwise, print `'No'`.

Exemplo de entradas e saídas

Exemplo de Entrada	Exemplo de Saída
azzel apple	Yes
chokudai redcoder	No
abcdefghijklmnopqrstuvwxyz ibyhqfrekavclxjstdwgpzmonu	Yes

Solução com complexidade $O(|S|)$

- Observe que, segundo as regras da operação dada, a correspondência entre dois caracteres de S e T é biunívoca
- Assim, um caractere c de S deve ser trocado por um caractere d de T que ocorra exatamente o mesmo número de vezes em ambas strings
- Além do número de ocorrências, é preciso determinar a localização destas ocorrências
- Assim, devem ser mantidos dois dicionários D_S e D_T que mantém o registro das substituições a serem realizadas
- Assim, para cada caractere c de S, se ainda não tiver sido definida sua substituição e nem a transformação do caractere d de T, ambas substituições são registradas
- Se uma substituição estiver definida e a outra não, será impossível transformar S em T
- Se a substituição de c já estiver definida mas for diferente de d, também será impossível obter T a partir de S

Solução com complexidade $O(|S|)$

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

using namespace std;

bool solve(const string& S, const string& T)
{
    auto N = S.size();
    map<char, char> s_table, t_table;

    for (size_t i = 0; i < N; ++i)
    {
        auto c = S[i];
        auto d = T[i];
        auto it = s_table.find(c);

```

Solução com complexidade $O(|S|)$

```
16          if (it == s_table.end())
17          {
18              auto jt = t_table.find(d);
19
20              if (jt == t_table.end())
21              {
22                  s_table[c] = d;
23                  t_table[d] = c;
24              } else
25                  return false;
26          } else if (it->second != d)
27                  return false;
28      }
29
30      return true;
31  }
32
```

Solução com complexidade $O(|S|)$

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

    string S, T;
    cin >> S >> T;

    auto ans = solve(S, T);

    cout << (ans ? "Yes" : "No") << '\n';

    return 0;
}
```