

算法分析与设计E6-C

22373340 詹佳博

题目描述

•

多彩字符串

时间限制: 1000 ms 内存限制: 65536 kb
总通过人数: 179 总提交人数: 186

题目描述

对于一个字符串 S ，我们定义它的「多彩值」 $f(S)$ 为该字符串中不同字符的数量。例如 $f(\text{abcd}) = 4$, $f(\text{owo}) = 2$ 。因为字符串 abcd 中有 $\text{a}, \text{b}, \text{c}, \text{d}$ 四个不同的字符，字符串 owo 中有 o, w 两个不同的字符。

我们称两个字符串 S 和 T 是「多彩对应」的当且仅当 $|S| = |T|$ 且对任意 $1 \leq i \leq |S|$ ，有 $f(S[i, j]) = f(T[i, j])$ ，其中 $S[i, j]$ 表示字符串 S 的第 i 个字符到第 j 个字符构成的子串。

例如：字符串 OwO 和 qAq 是多彩对应的，因为 $f(\text{O}) = f(\text{q}) = 1$, $f(\text{Ow}) = f(\text{qA}) = 2$, $f(\text{OwO}) = f(\text{qAq}) = 2$, $f(\text{w}) = f(\text{A}) = 1$, $f(\text{wO}) = f(\text{Aq}) = 2$, $f(\text{O}) = f(\text{q}) = 1$ 。但 bu 和 aa 不是多彩对应的，因为 $f(\text{bu}) = 2$ 而 $f(\text{aa}) = 1$ 。

现给出两个长度相同的字符串（仅由大小写字母和数字组成），请你判断他们是不是多彩对应的。

输入

第一行一个正整数 T ($1 \leq T \leq 10^5$) 表示数据组数。

接下来 T 组数据：

第一行一个正整数 n ($1 \leq n \leq 10^5$) 表示字符串的长度。

第二行一个长度为 n 的字符串 S 。

第三行一个长度为 n 的字符串 T 。

单个测试点保证 T 组数据的字符串长度和不超过 10^6 。

输出

对于每组数据，若给出的两个字符串是多彩对应的，则输出 **Yes**，否则输出 **No**。

题目分析

简而言之，就是判断两个字符串是否“相似”。

解题思路：

- 用数学归纳法，可得知上字符串的一个字符必恒对应下字符串的某一个字符，相应的下字符串某一个字符也相对应上字符串一个字符，用两个hash求解。
- G题思路：用一个过渡字符串来判断上下字符串是否相等，而过渡字符串恒以输入字符串只要出现与前面不同的新字符情况，便以abc顺序哈希。

设某一个字符同时对多个字符: 当 $n=2$ 时 $a \rightarrow b, c$

则有 $\text{string } A = "aa"$
 $\text{string } B = "bc"$ 明显不符合题意

① 当 $n=1$ 时, 必成立

② 当 $n=2$ 时, 由题意必成立.

③ 当 $n=k$ 时, 如果 $\text{string } A = "x_1 x_2 \dots x_k"$
 $\text{string } B = "y_1 y_2 \dots y_k"$ 相互对应.

$n=k+1$ 时, 在 A 中任意插入一个对应字符, 必保证 A, B 合法

得证

设 $A = "acdbccabd"$
 $B = "xyzeyyxex"$

题目求解

方法1:

```
#include <bits/stdc++.h>
using namespace std;
string a, b;
int hasha[130], hashb[130];

void init() {
    memset(hasha, 0, sizeof hasha);
    memset(hashb, 0, sizeof hashb);
}
```

```

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int t;
    cin >> t;
    while (t--) {
        init();
        int flag = 0;
        int len;
        cin >> len >> a >> b;
        for (int i = 0; i < len; i++) {
            if (hasha[a[i]] == 0) {
                hasha[a[i]] = b[i];
            }
            if (hashb[b[i]] == 0) {
                hashb[b[i]] = a[i];
            }

            if(hasha[a[i]]!=b[i] || hashb[b[i]]!=a[i]){
                flag=1;
                cout<<"No"<<endl;
                break;
            }
        }
        if(!flag){
            cout<<"Yes"<<endl;
        }
    }
}

```

方法2:

```

#include <bits/stdc++.h>
using namespace std;
string a, b;
unordered_map<char, char> m;

string change(string in) {
    string out = "";
    m.clear();
    char start = 'a';
    for (auto c : in) {
        if (m.count(c) == 0) {
            m[c] = start;
            start++;
        }
        out += m[c];
    }
    return out;
}

```

```
int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int t;
    cin >> t;
    while (t--) {
        int temp;
        cin >> temp;
        cin >> a >> b;
        a = change(a);
        b = change(b);
        if (a == b) {
            cout << "Yes" << '\n';
        } else {
            cout << "No" << '\n';
        }
    }
}
```

时间复杂度

该算法时间复杂度特别直观。只有哈希的 $O(n)$ 。