String painter

2021 年 5 月 21 日

String painter

There are two strings A and B with equal length. Both strings are made up of lower case letters. Now you have a powerful string painter. With the help of the painter, you can change a segment of characters of a string to any other character you want. That is, after using the painter, the segment is made up of only one kind of character. Now your task is to change A to B using string painter. What' s the minimum number of operations?

input

Input contains multiple cases. Each case consists of two lines: The first line contains string A. The second line contains string B. The length of both strings will not be greater than 100.

Output

A single line contains one integer representing the answer.

Sample Input

```
zzzzzfzzzzz
abcdefedcba
abababababab
cdcdcdcdcdcd
```

Sample Output

```
6
7
```

空白串转 B 串

暴力原型，以 i 为起点向后枚举终点把区间内的点都染成 $B[i]$，总共有 $n!$ 种方案。

剪枝优化，1 为起点时终点为 n，其他 i 点为起点时若已被染成 $B[i]$，则跳过该点。

分类标准，若最终 n 点被其他起点的染色区域覆盖，则该区间可分为多段子区间，否则表明 $B[1]=B[n]$，且 n 点可从该区间去掉。

状态数组，$g[l][r]$ 表示在区间 $[l,r]$ 上空白串转 B 串的最小操作数。

转移方程，$g[l][r]=min\{g[l][k]+g[k+1][r]\}$，若 $B[l]=B[r]$，则 $g[l][r]=g[l][r-1]$。

A 串转 B 串

分类标准，若 $A[n]=B[n]$，则 n 点可从该区间去掉，否则最终覆盖 n 点染色区间被视为空白字串。

状态数组，$f[r]$ 表示在区间 $[1,r]$ 上 A 串转 B 串的最小操作数。

转移方程，若 $A[r]=B[r]$，则 $f[r]=f[r-1]$，否则 $f[r]=min\{f[k]+g[k+1][r]\}$。

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

using namespace std;

long long g[101][101], f[101];

int main() {
    string s, t;
    while(cin >> s >> t) {
        int n = s.size();
        for(int i = 1; i <= n; ++ i) g[i][i] = 1;
        for(int i = 2; i <= n; ++ i) {
            for(int l = 1; l <= n-i+1; ++ l) {
                int r = l + i - 1;
                g[l][r] = INT_MAX;
                if (t[l-1] == t[r-1]) g[l][r] = g[l][r-1];
                for(int k = l; k < r; ++ k) {
                    g[l][r] = min(g[l][r], g[l][k] + g[k+1][r]);
                }
```

```
            }
        }
        for(int r = 1; r <= n; ++ r) {
            if (s[r-1] == t[r-1]) f[r] = f[r-1];
            else {
                f[r] = INT_MAX;
                for(int k = 0; k < r; ++ k) {
                    f[r] = min(f[r], f[k] + g[k+1][r]);
                }
            }
        }
        cout << f[n] << "\n";
    }
    return 0;
}
```