题目

A B C D E F G H I J

去做题

F 提交记录:

ID	结果	时间	内存
5789145	AC	402	18852

F Pure Ruby

时间限制：1000ms 内存限制：65536kb

通过率：81/106 (76.42%) 正确率：81/312 (25.96%)

题目描述

给定两个仅包含小写英文字母的字符串 S_1，S_2，试求二者**最长公共子串**的长度，以及二者**最长公共子序列**的长度。

我们称 A 是 S 的**子串**，当且仅当 A 可由 S 删去一段前缀与一段后缀得到，例如 `dest` 和 `star` 都是 `jadestar` 的子串，而 `jar` 不是。

我们称 A 是 S 的**子序列**，当且仅当 A 可由 S 删去一些字符得到，例如 `jet` 和 `jar` 都是 `jadestar` 的子序列，而 `rest` 不是。

我们称 A 是 S_1 与 S_2 的**公共子串**，当且仅当 A 既是 S_1 的子串又是 S_2 的子串。

我们称 A 是 S_1 与 S_2 的**公共子序列**，当且仅当 A 既是 S_1 的子序列又是 S_2 的子序列。

输入

本题测试点包含多组数据。

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

对于每组数据：

第一行，一个字符串 S_1（$1 \leq |S_1| \leq 2000$）。

思路

公共子串和公共子序列是两个不相同的概念, 公共子串要求字符连续, 公共子序列不要求字符连续。

公共子序列

字符串 s1的前 i 位子串和 s2的前 j 位子串的最长公共子序列 f(i,j), 跟我们求 s1的前 i-1 位子串和 s2的前 j-1 位子串的最长公共子序列 f(i-1,j-1) 两者之间有什么关系呢? 或者换一种说法, 如果我们知道的 f(i-1,j-1) 的结果 x, 我们如何根据 x 推得 f(i,j) 呢?

其结论就是, 我们判断 s1.charAt(i) 和 s2.charAt(j) 是否相等, 如果相等, 则 f(i,j) = f(i-1, j-1) + 1; 如果不相等, f(i,j) = max(f(i-1, j), f(i, j-1));

公共子串

最长公共子序列问题中我们探索 f(i, j) 和 f(i-1, j-1) 的公共子序列的关系 那么类比到最长公共子串, 你觉得 f(i, j) 和 f(i-1, j-1) 有什么关系呢?

有趣的一点在于, 我们求解最长公共子串时, 要求连续性, 所以 s1.charAt(i) 是否等于 s2.charAt(j), 只能决定特殊情况下 (i,j) 范围 的最长公共子串 和 (i-1, j-1) 范围 的最长公共子串的关系, 那就是这个最长公共子串必须是以 i-1/j-1 结尾的, 那么当 s1.charAt(i)==s2.charAt(j) 时, 才能满足 f(i, j) = f(i-1, j-1) + 1

也就是这里的 f(i, j) 代表的含义是以 i/j 位置为结尾的最长公共子串的长度 而当s1.charAt(i) != s2.charAt(j) 时, f(i, j) = 0

代码实现

```
#include <iostream>
#include <vector>

using namespace std;

int longestCommonSubstring(const string& s1, const string& s2) {
    int n = s1.length();
    int m = s2.length();

    vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));
    int maxLen = 0;

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
                maxLen = max(maxLen, dp[i][j]);
            }
        }
    }

    return maxLen;
}

int longestCommonSubsequence(const string& s1, const string& s2) {
    int n = s1.length();
    int m = s2.length();

    vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }

    return dp[n][m];
}

int main() {
    int T;
    cin >> T;

    while (T--) {
        string s1, s2;
```

```
        cin >> s1 >> s2;

        int commonSubstring = longestCommonSubstring(s1, s2);
        int commonSubsequence = longestCommonSubsequence(s1, s2);

        cout << commonSubstring << " " << commonSubsequence << endl;
    }

    return 0;
}
```