# C4-F题解

## 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≤ $T$ ≤10），表示数据组数。

对于每组数据：

第一行，一个字符串 $S_1$（1≤| $S_1$ |≤2000）。

第二行，一个字符串 $S_2$（1≤| $S_2$ |≤2000）。

## 输出

对于每组数据：

输出一行，两个整数，分别为 $S_1$ 与 $S_2$ 最长公共子串的长度，以及最长公共子序列的长度。

## 输入样例

```
1
lapispureruby
jadestarsepia
```

## 输出样例

```
2 4
```

## 题解

这个题可以拆成两个题：**最长公共子串**，**最长公共子序列**，分开解决

1. 最长公共子串
    - 变量设置：`str1[N],str2[N]` 两个字符串(下标从0开始)

- 状态表示：`f[i][j]` 表示以`str1`的第`i`个字符结尾的子串和以`str2`第`j`个字符结尾的子串的最长公共子串，这样可以求出来以两个字符串中任意两个字母结束的最长公共子串，在计算的过程中保存最大值输出即可。（`f[i][j]`表示的是第几个字符，所以会和`str`有一个错位）
- 状态转移：`f[i][j]` 规定了子串结束的位置，所以

  1）当`str1[i-1]==str2[j-1]`时，以`str1`的第`i`个字符结尾的子串和以`str2`第`j`个字符结尾的子串的最长公共子串的长度等于以`str1`的第`i-1`个字符结尾的子串和以`str2`第`j`-1个字符结尾的子串的最长公共子串的长度`+1`；

  2）当`str1[i-1]!=str2[j-1]`时，以`str1`的第`i`个字符结尾的子串和以`str2`第`j`个字符结尾的子串的最长公共子串的长度为`0`

  总结状态转移方程：

$$f[i,j]=\begin{cases}0 & i=0 or j=0\\ f[i-1,j-1]+1, & str1[i-1]==str2[j-1]\\ 0, & str1[i-1]!=str2[j-1]\end{cases}$$

  注意，这里的`i,j`是指第`i`和第`j`个字符，对应`str`中的下标是`i-1,j-1`，所以错了一位

2. 最长公共子序列(LCS)

- 变量设置：`str1[N],str2[N]`两个字符串（下标从0开始）
- 状态表示：`g[i][j]`表示`str1`的前`i`个字符的子串和以`str2`前`j`个字符的子串的最长公共子序列，最终求的值为`g[strlen(str1)][strlen(str2)]`的值（`g[i][j]`表示的是第几个字符，所以会和`str`有一个错位）
- 状态转移：

  1）当`str1[i-1]==str2[j-1]`时，最长公共子序列在`g[i-1][j-1]`的基础上`+1`

  2）当`str1[i-1]!=str2[j-1]`时，最长公共子序列为`g[i-1][j]`和`g[i][j-1]`中较大的一个

  总结状态转移方程：

$$g[i,j]=\begin{cases}0 & i=0 or j=0\\ g[i-1,j-1]+1, & str1[i-1]==str2[j-1]\\ max(g[i-1][j],g[i][j-1]), & str1[i-1]!=str2[j-1]\end{cases}$$

  注意，这里的`i,j`是指第`i`和第`j`个字符，对应`str`中的下标是`i-1,j-1`，所以错了一位

# 代码

```cpp
#include <iostream>
#include <algorithm>
#include <string.h>
using namespace std;
const int N = 2010;
char s1[N], s2[N];
int f[N][N];
int main()
{
  int T;
  scanf("%d", &T);
  while (T--)
  {
```

```
    scanf("%s", s1);
    scanf("%s", s2);
    int len1 = strlen(s1);
    int len2 = strlen(s2);
    memset(f, 0, sizeof(f));
    int maxx = 0;
    for (int i = 1; i <= len1; i++)
    {
      for (int j = 1; j <= len2; j++)
      {
        if (s1[i - 1] == s2[j - 1])
        {
          f[i][j] = f[i - 1][j - 1] + 1;
          if (f[i][j] > maxx)
            maxx = f[i][j];
        }
      }
    }
    printf("%d ", maxx);
    memset(f, 0, sizeof(f));
    for (int i = 1; i <= len1; i++)
    {
      for (int j = 1; j <= len2; j++)
      {
        if (s1[i - 1] == s2[j - 1])
        {
          f[i][j] = f[i - 1][j - 1] + 1;
        }
        else
          f[i][j] = max(f[i - 1][j], f[i][j - 1]);
      }
    }
    printf("%d\n", f[len1][len2]);
  }
  return 0;
}
```

## 标程代码

```
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

const int N = 2005;

int n1, n2;
char s1[N], s2[N];
int f[N][N], g[N][N];
int maxf, maxg;

void solve()
{
    scanf("%s", s1 + 1);
```

```
    scanf("%s", s2 + 1);
    n1 = strlen(s1 + 1);
    n2 = strlen(s2 + 1);
    maxf = maxg = 0;
    for (int i = 1; i <= n1; i++)
        for (int j = 1; j <= n2; j++)
        {
            f[i][j] = s1[i] == s2[j] ? f[i - 1][j - 1] + 1 : 0;
            g[i][j] = max({g[i - 1][j - 1] + (s1[i] == s2[j]), g[i - 1][j], g[i][j - 1]});
            maxf = max(maxf, f[i][j]);
            maxg = max(maxg, g[i][j]);
        }
    printf("%d %d\n", maxf, maxg);
}

int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
        solve();
    return 0;
}
```