

## 最长公共子序列

给定两个字符串 `text1` 和 `text2`，返回这两个字符串的最长 **公共子序列** 的长度。如果不存在 **公共子序列**，返回 0。

一个字符串的 **子序列** 是指这样一个新的字符串：它是由原字符串在不改变字符的相对顺序的情况下删除某些字符（也可以不删除任何字符）后组成的新字符串。

- 例如，"ace" 是 "abcde" 的子序列，但 "aec" 不是 "abcde" 的子序列。

两个字符串的 **公共子序列** 是这两个字符串所共同拥有的子序列。

### 示例 1：

输入：text1 = "abcde", text2 = "ace"

输出：3

解释：最长公共子序列是 "ace"，它的长度为 3。

### 示例 2：

输入：text1 = "abc", text2 = "abc"

输出：3

解释：最长公共子序列是 "abc"，它的长度为 3。

### 示例 3：

输入：text1 = "abc", text2 = "def"

输出：0

解释：两个字符串没有公共子序列，返回 0。

		b	d	c	a	b	a
i		0	0	0	0	0	0
0		0	0	0	0	0	0
1	a	0	0	0	0	1	1
2	b	0	1	1	1	1	2
3	c	0	1	1	2	2	2
4	b	0	1	1	2	2	3
5	d	0	1	2	2	2	3
6	a	0	1	2	2	3	3
7	b	0	1	2	2	3	4

```

class Solution {
public:
    int longestCommonSubsequence(string text1, string text2) {
        vector<vector<int>> dp(text1.size()+1,vector<int>(text2.size()+1,0)
    );
        int i,j;
        for(i=1;i<=text1.size();i++)
        {
            for(j=1;j<=text2.size();j++)
            {
                if(text1[i-1]==text2[j-1])
                {
                    dp[i][j]=dp[i-1][j-1]+1;
                }
                else
                    dp[i][j]=max(dp[i][j-1],dp[i-1][j]);
            }
        }
        return dp[text1.size()][text2.size()];
    }
};

```