C++方向编程题答案

答案说明：

> 大家如果对本次题目或者答案有问题，可以联系下方的出题老师答疑。
>
> **出题老师：**
>
> 选择题：张文超　qq：3627274478
>
> 代码题：时亮益　qq：569334855

第七周

day39

> 题目ID：25946 字符串计数
>
> 链接：https://www.nowcoder.com/questionTerminal/f72adfe389b84da7a4986bde2a886ec3
>
> **【题目解析】**
>
> 题目意思：按照字典序列：找到s1和s2之间长度在len1和len2范围内的字符串个数。直接做不好处理，此处需要转化思路，找到一个合适的模型：因为从'a'~'z'，刚好有26个字母，因此可以将s1和s2看成是26进制数据，题目就变得简单了，将其转化为：从s1和s2之间有多少个不同数字，最后求解出长度不同的数组的个数即可。
>
> **【解题思路】**
>
> 1. 循环接受收入，保证所有测试用例可以验证到
> 2. 将s1和s2补齐到len2位，因为在字典序列中s1比s2靠前，因此s1后序所有位补'a'，s2后补'z'+1
> 3. 确认s1和s2两个字符串每个字符位置上的差值
> 4. 确认len1和len2之间不同字符的个数
> 5. 注意输出时需要模1000007

```
/*
补齐字符串，按照26进制进行计算
*/
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#define N 1000007
#include<math.h>
using namespace std;

int main()
{
 string s1,s2;
 int len1,len2;
 while(cin>>s1>>s2>>len1>>len2)
 {
     //将该字符串看成是26进制数，为了简单起见，将s1和s2补长到len2长度
     // 注意：s2补的是'z'+1, 因为'z' - 'a' = 25
     s1.append(len2-s1.size(),'a');
     s2.append(len2-s2.size(),(char)('z'+1));
```

```
        // 确认s1和s2的两个字符串每个位置上的差值
        vector<int> array;
        for(int i=0;i<len2;i++){
            array.push_back(s2[i]-s1[i]);
        }

        // 确认len1和len2之间可组成的不同字符串的个数
        int result = 0;
        for(int i=len1;i<=len2;i++){
            for(int k=0;k<i;k++){
                result += array[k]*pow(26,i-1-k);
            }
        }
        //所有字符串最后都不包含是s2自身，所以最后要减1;
        cout<<(result-1)%N<<endl;
    }
    return 0;
}
```

806 最长公共子序列

链接：https://www.nowcoder.com/questionTerminal/9ae56e5bdf4f480387df781671db5172

【题目解析】

题目要求比较简单：**获取两个字符串的最长公共的子序列**，注意：**子序列即两个字符串中公共的字符，但不一定连续。**

【解题思路】

```
例子:
s = "abcd"    长度m = 4
t = "becd"    长度n = 4
则s和t的最长公共子序列(LCS Largest Common Subsequence)的长度为3("bcd")
```

直接进行求解，不太好求解，我们可以使用动规的思想来进行处理。

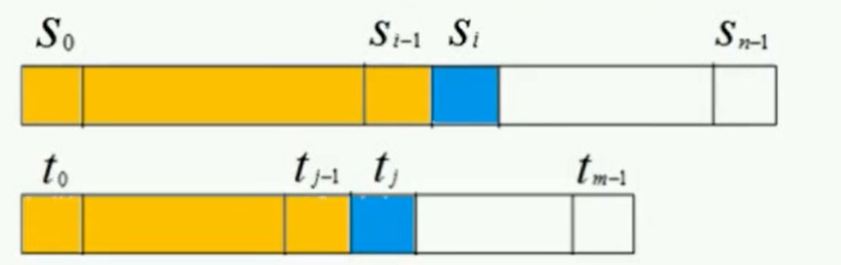
分析：

已知两个字符串S${0}$S${1}$.....S${i}$.....S${{m-1}}$和t${0}$t${1}.....$t${j}$.....t${{n-1}}$

从题干中可以提取出问题：求字符串s和t的最长公共子序列

假设LCS(m,n)为长度为m的字符串s与长度为n的字符串t的最长公共子序列，直接进行求解时不太好求解，那么可以将问题简化为其子问题，假设s和t的长度为任意值i和j，即求LCS(i,j)：

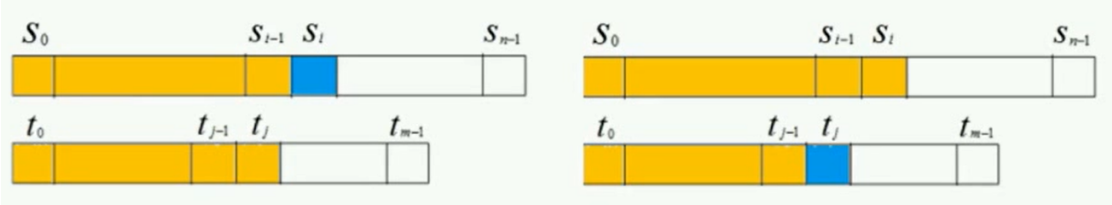
1. **当S${i}$ == t${j}$ 时**



此时S${0}$S${1}$.....S${{i-1}}$S${{i}}$和t${0}$t${1}.....$t${{j-1}}$t${j}$的最长子序列为：S${0}$S${1}$.....S${{i-1}}$和t${0}$t${1}.....$t${{j-1}}$的最长子序列+1

即：**LCS(i,j) = LCS(i-1, j-1)+1**即在其子序列的基础上加上s${_i}$

2. **当S${*i*}$!= *t${j}$* 时**



此时S${*0*}$S${1}$.....S${*{i-1}*}$S${{i}}$和t${*0*}$t${1}.....$t${*{j-1}*}$t${j}$的最长子序列为：S${*0*}$S${1}$.....S${*{i-1}*}$*和*t${0}$t${*1*}.....*$t${*{j-1}}$t${*j*}$的最长子序列与*S${0}$S${*1*}$.....S${*{i-1}}$S${*i*}$和*t${0}$t${*1*}.....$t${*{j-1}}$最长子序列的最大值，即：

LCS(i,j) = max(LCS(i-1,j), LCS(i,j-1));

经过上述分析之后，状态方程可以建立：

假设：$dp[i][j]$为长度为i的字符串S和长度为j的字符串T的最长公共子串，则有如下的地推公式

当$s[i] == t[j]$时：$dp[i][j] = dp[i-1][j-1] + 1$

当$s[i] != t[j]$时：$dp[i][j] = max(dp[i-1][j], dp[i][j-1])$

按照上述地推公式，逐个字符去进行检测，最后返回$dp[m][n]$

```
#include <iostream>
using namespace std;

#include <string>
#include <vector>

int LCS(const string& m, const string& n)
{
 size_t mLen = m.size(), nLen = n.size();
 vector<vector<int>> dp(mLen+1, vector<int>(nLen+1));
 for(size_t i = 1; i <= mLen; ++i)
 {
     for(size_t j = 1; j <= nLen; ++j)
     {
         // 如果m[i-1]与n[j-1]相等，则公共字符列为：dp[i-1][j-1]+1
         if(m[i-1] == n[j-1])
         {
             dp[i][j] = dp[i-1][j-1] + 1;
         }
         else
         {
             // 否则：为上一步的最大值
             dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
         }
     }
 }

 return dp[mLen][nLen];
}

int main()
{
 // 循环处理所有的测试用例
```

```
    string m,n;
    while(cin>>m>>n)
    {
        cout<<LCS(m,n)<<endl;
    }
    return 0;
}
```