动态规划系列专题讲义

专题三：最长公共子序列

/\*

Name: 动态规划专题之最长公共子序列

Author: 巧若拙

Description: 1808\_公共子序列

描述：我们称序列Z = < z1, z2, ..., zk >是序列X = < x1, x2, ..., xm >的子序列当且仅当存在 严格上升 的序列< i1, i2, ..., ik >，使得对j = 1, 2, ... ,k, 有xij = zj。比如Z = < a, b, f, c > 是X = < a, b, c, f, b, c >的子序列。

现在给出两个序列X和Y，你的任务是找到X和Y的最大公共子序列，也就是说要找到一个最长的序列Z，使得Z既是X的子序列也是Y的子序列。

输入：输入包括多组测试数据。每组数据包括一行，给出两个长度不超过200的字符串，表示两个序列。两个字符串之间由若干个空格隔开。

输出：对每组输入数据，输出一行，给出两个序列的最大公共子序列的长度。

样例输入

abcfbc abfcab

programming contest

abcd mnp

样例输出

4

2

0

\*/

#include<iostream>

#include<cstring>

#include<string>

using namespace std;

const int N = 200;

int B[N+1][N+1];

int B1[N+1][N+1];

int pre[N+1]; //pre[j]相当于B1[i-1][j]

int cur[N+1]; //cur[j]相当于B1[i][j]

string X, Y;

int LCSLength(int i, int j);//自顶向下的备忘录算法

int LCSLength\_1(int n, int m);//动态规划

int LCSLength\_2(int n, int m);//动态规划

void PrintLCS(int i, int j);

int main()

{

while (cin >> X >> Y)

{

memset(B, 0, sizeof(B));

memset(pre, 0, sizeof(pre));

cout << LCSLength(X.length(), Y.length()) << endl;

cout << LCSLength\_1(X.length(), Y.length()) << endl;

cout << LCSLength\_2(X.length(), Y.length()) << endl;

PrintLCS(X.length(), Y.length());

}

return 0;

}

算法2：备忘录算法：自顶而下，需要用到全局变量B[N+1][N+1]。

int LCSLength(int i, int j)

{

if (B[i][j] != 0)

return B[i][j];

if (i == 0 || j == 0)

B[i][j] = //语句1

else if (X[i-1] == Y[j-1])

B[i][j] = //语句2

else

B[i][j] = max( ); //语句3

return B[i][j];

}

问题1：将语句1，语句2和语句3补充完整。

参考答案：

问题1：语句1：B[i][j] = 0;

语句2：B[i][j] = LCSLength(i-1, j-1) + 1;

语句3：B[i][j] = max(LCSLength(i-1, j), LCSLength(i, j-1));

算法2：动态规划：自底而上，需要用到全局变量B1[N+1][N+1]。

int LCSLength\_1(int n, int m)

{

for (int i=1; i<=n; i++)

{

for (int j=1; j<=m; j++)

{

if (X[i-1] == Y[j-1])

B1[i][j] = //语句1

else

B1[i][j] = //语句2

}

}

return B1[n][m];

}

问题1：将语句1和语句2补充完整。

参考答案：

问题1：语句1：B1[i][j] = B1[i-1][j-1] + 1;

语句2：B1[i][j] = max(B1[i-1][j], B1[i][j-1]);

算法3：动态规划：使用2个数组存储记录，需要用到全局变量pre[]和cur[]均初始化为0。

int LCSLength\_2(int n, int m)

{

for (int i=1; i<=n; i++)

{

for (int j=1; j<=m; j++)

{

if (X[i-1] == Y[j-1])

cur[j] = //语句1

else

cur[j] = //语句2

}

for (int j=1; j<=m; j++)

{

pre[j] = cur[j];

}

}

return pre[m];

}

问题1：将语句1和语句2补充完整。

参考答案：

问题1：语句1：cur[j] = pre[j-1] + 1;

语句2：cur[j] = max(pre[j], cur[j-1]);

拓展练习：在算法1中，我们用二维数组B[][]记录了各种解的信息，现在请你根据B[][]记录的信息，设计一个递归函数void PrintLCS(int i, int j);//i和j分别表示字符串X的第i个字符和字符串Y的第j个字符。

参考答案：

void PrintLCS(int i, int j)

{

if (i == 0 || j == 0)

return;

if (X[i-1] == Y[j-1])

{

PrintLCS(i-1, j-1);

cout << "x[" << i-1 << "]= " << X[i-1] << " : "<< "y[" << j-1 << "]= " << Y[j-1] << endl;

}

else if(B[i-1][j] > B[i][j-1])//先向上层走

PrintLCS(i-1, j);

else //再向左边走

PrintLCS(i, j-1);

}

课后练习：

练习1：编辑距离

描述：A和B是2个字符串。要用最少的字符操作将字符串A转换为字符串B。

这里所说的字符操作包括 (1)删除一个字符； (2)插入一个字符； (3)将一个字符改为另一个字符。

将字符串A变换为字符串B所用的最少字符操作数称为字符串A到B的编辑距离，记为 d(A,B)。

试设计一个有效算法，对任给的2 个字符串A和B，计算出它们的编辑距离d(A,B)。

Input：输入的第一行是字符串A，文件的第二行是字符串B。

Output：程序运行结束时，将编辑距离d(A,B)输出。

Sample Input

fxpimu

xwrs

Sample Output

5

练习2：电路布线

描述：在一块电路板的上、下两端分别有n个接线柱。根据电路设计，要求用导线（i，π(i)）将上端接线柱i与下端接线柱π(i)相连。

其中，π(i)，1<=i<=n是{1，2，…，n}的一个排列。导线（i，π(i)）称为该电路板上的第i条连线。对于任何1<=i π(j)。

在制作电路板时，要求将这n条连线分布到若干绝缘层上。在同一层上的连线不相交。

你的任务是要确定将哪些连线安排在第一层上，使得该层上有尽可能多的连线。换句话说，就是确定导线集Nets={ i，π(i)，1<=i<=n}的最大不相交子集。

输入：输入文件第一行为整数n；第二行为用一个空格隔开的n个整数，表示π(i)。

输出：输出文件第一行为最多的连线数m，第2行到第m+1行输出这m条连线（i，π(i)）。

【输入样例】

10

1 8

2 7

3 4

4 2

5 5

6 1

7 9

8 3

9 10

10 6

【输出样例】

4