# 实验二：动态规划

**一、实验目的**

理解动态规划的基本思想，理解动态规划算法的两个基本要素最优子结构性质和子问题的重叠性质。熟练掌握典型的动态规划问题。掌握动态规划思想分析问题的一般方法，对较简单的问题能正确分析，设计出动态规划算法，并能快速编程实现。

**二、实验内容**

对于长度相同的2个字符串A和B，其距离定义为相应位置字符距离之和。2个非空格字符的距离是它们的ASCII码之差的绝对值。空格与空格的距离为0，空格与其他字符的距离为一定值k。

在一般情况下，字符串A和B的长度不一定相同。字符串A的扩展是在A中插入若干空格字符所产生的字符串。在字符串A和B的所有长度相同的扩展中，有一对距离最小的扩展，该距离称为字符串A和B的扩展距离。

算法要求如下

1、 数据输入：第1行是字符串A，第2行是字符串B，第3行是空格与其他字符的距离定值k。

2、 输出：字符串A和B的扩展距离。

例如

输入：

cmc

snmn

2

输出：10

注：设字符串A和B的子串A[1..i]和B[1..j]的扩展距离为val(i,j)，则val(i,j)具有最优子结构性质，递归定义为：val(i,j)=min{val(i-1,j)+k, val(i,j-1)+k, val(i-1,j-1)+dist(ai,bj)}

1. **动态规划基本思想**

**(1)描述字符串A和B扩展距离问题的最优解结构特征：**

1.若A和B都为空串时，显然两个空串的扩展距离为0；

2.若A串的扩展的最后一个字符是字母，B串的扩展的最后一个字符是空格，则A串和B串的扩展距离为val(i,j) = val(i,j-1) + k；

3.若A串的扩展的最后一个字符是空格，B串的扩展的最后一个字符是字母，则A串和B串的扩展距离为val(i,j) = val(i-1,j) + k；

4.若A串的扩展的最后一个字符是字母a，B串的扩展的最后一个字符是字母b，则A串和B串的扩展距离为val(i,j) = val(i-1,j-1) + dist(a,b)；

**(2)递归地定义一个最优解的值**

设val[i,j]表示字符串A和B的扩展距离的最优解。

根据(1)的最优解结构描述，得到如下的递归公式：

(1) A和B为空串:

val[i,j] = 0

(2) A和B中有一个不为空串:

val[i,j] = min{val[i-1,j]+k,val[i,j-1]+k, val[i-1,j-1]+dist(ai,bj)}

**(3)自底向上计算一个最优解的值**

先初始化val[0][0] = 0（A和B为空串时）

然后采用从左到右，从上到下的计算顺序，根据递归公式来填充矩阵val的每一项，最后得到val[A.length][B.length]，即为字符串A和B的扩展距离。

计算的详细过程，见实验过程代码所示。

**(4)从已计算的信息中构造一个最优解**

这里题目没有要求打印出A和B的距离最小的扩展，所以我没有在这里继续作分析。

1. **实验过程**

**实验所用开发平台为vscode**

**实验代码描述语言为C++**

#include <iostream>

#include <cmath>

#include <cstdlib>

using namespace std;

#define MAX 10000 //标识最大的可能整数

int val[500][500]; //最优扩展距离矩阵 val(i,j)=min{val(i-1,j)+k, val(i,j-1)+k, val(i-1,j-1)+dist(ai,bj)}

string strA; //字符串A

string strB; //字符串B

int k; //定值k

//返回字符a和b的ASCII码的差的绝对值

int dist(char a, char b)

{

return abs(a - b);

}

int dp()

{

int lenA, lenB;

int tmp;

val[0][0] = 0; //当strA和strB是空串时,最优扩展距离为 0

lenA = strA.length() - 1;

lenB = strB.length() - 1;

//cout<<lenA<<endl; cmc 4

//cout<<lenB<<endl; snmn 5

for (int i = 0; i <= lenA; i++)

{

for (int j = 0; j <= lenB; j++)

{

if (i + j)

{

val[i][j] = MAX;

//A串最后一个字符是空格,B串最后一个字符是字母

tmp = val[i-1][j] + k;

if (i && (tmp < val[i][j]))

{

val[i][j] = tmp;

}

//A串最后一个字符是字母,B串最后一个字符是空格

tmp = val[i][j-1] + k;

if (j && (tmp < val[i][j]))

{

val[i][j] = tmp;

}

//A串和B串最后一个字符都是字母

tmp = val[i-1][j-1] + dist(strA[i],strB[j]);

if ((i\*j) && (tmp < val[i][j]))

{

val[i][j] = tmp;

}

}

}

}

return val[lenA][lenB];

}

int main()

{

cin>>strA;

cin>>strB;

cin>>k;

strA = ' ' + strA;

strB = ' ' + strB;

cout<<dp()<<endl;

// for (int i = 0; i < strA.length(); i++)

// {

// for (int j = 0; j < strB.length(); j++)

// {

// cout<<val[i][j]<<' ';

// }

// cout<<endl;

// }

system("pause");

return 0;

}

1. **实验结果**

测试用例：

输入：

cmc

snmn

2

输出：

10

