# 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**

**设状态dp[i][j]表示正在匹配A[i]与B[j]的最小扩展距离**

**转移方程:**

**如果仅在A[i]前插入空格则有dp[i][j]+k→dp[i][j+1]**

**如果仅在B[i]前插入空格则有dp[i][j]+k→dp[i+1][j]**

**如果两者都插入空格dp[i][j]→dp[i][j]**

**如果两者都不插入空格dp[i][j]→dp[i+1][j+1]+abs(A[i]-B[i])**

**初始条件: dp[0][0]=0**

**最终结果:dp[A.length()][B.length()]**

1. **实验过程**

#include <bits/stdc++.h>

using namespace std;

const int INF = 0x3f3f3f3f;

int k;

string A, B;

vector<vector<int>> dp;

signed main() {

#ifdef ONLINE\_JUDGE

  ios::sync\_with\_stdio(false); cin.tie(nullptr); cout.tie(nullptr);

#endif

  cin >> A >> B >> k;

  int la = A.size(), lb = B.size();

  assert(lb >= la);

  dp = vector<vector<int>>(la+1, vector<int>(lb+1, INF));

  dp[0][0] = 0;

  for (int i = 0; i <= la; ++i) {

    for (int j = 0; j <= lb; ++j) {

      if (i < la && j < lb) dp[i+1][j+1] = min(dp[i+1][j+1], dp[i][j]+abs(A[i]-B[j]));

      if (i < la) dp[i+1][j] = min(dp[i+1][j], dp[i][j]+k);

      if (j < lb) dp[i][j+1] = min(dp[i][j+1], dp[i][j]+k);

    }

  }

  cout << dp[la][lb] << '\n';

  return 0;

}

1. **实验结果**

