**算法分析与设计实验报告**

**第 3 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 杨杰 | 学号 | 201908010705 | | 班级 | 计科1907 |
| 时间 | 5.20 | 地点 | 软件大楼 | | | |
| 实验名称 | 字符距离问题 | | | | | |
| 实验目的 | 通过上机实验，要求掌握字符距离的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 使用动态规划算法，自底向上的计算各个子问题并利用每次计算的结果，避免重复运算，从而降低算法复杂度。  设字符串A和B的子串A[1...i]和B[1...j]的扩展距离是val(i, j)；  依题意，字符串A和B有三种可能的情况：  1）i 和空格匹配。则val(i, j) = val(i-1, j) + k；  2）j 和空格匹配。则val(i, j) = val(i, j-1) + k；  3）扩展距离val(i-1,j-1)加上 i，j 的字符距离。则val(i, j) = val(i-1, j-1) + dist(ai , bi);  由上可知，val(i, j)具有最优子结构性质，且满足如下递推式：  val(i, j) = min{ val(i-1, j) + k，val(i, j-1) + k，val(i-1, j-1) + dist(ai , bi) }  最终即求解dp[len1][len2],len1,len2为两个字符串的长度。 | | | | | |
| 实验步骤 | 1）初始化dp数组的第一行和第一列。  2）按照递推式val(i, j) = min{ val(i-1, j) + k，val(i, j-1) + k，val(i-1, j-1) + dist(ai , bi) }，填写dp数组。  3）dp[len1][len2]即为最短字符距离。 | | | | | |
| 关键代码 | int dist(int i, int j)*//返回a中第i个字符 b中第j个字符的acsii值之差*  {      return abs(a[i - 1] - b[j - 1]);  }  void fun(int len1, int len2, int k)  {      for (int i = 0; i <= len1; ++i)*//有一个字符串是空的*          dp[i][0] = k \* i;      for (int i = 0; i <= len2; ++i)          dp[0][i] = k \* i;      for (int i = 1; i <= len1; ++i)*//递归式*          for (int j = 1; j <= len2; ++j)              dp[i][j] = min(dp[i - 1][j - 1] + dist(i, j), min(dp[i - 1][j] + k, dp[i][j - 1] + k));  } | | | | | |
| 算法复杂度分析 | 从动态规划递推式可知，算法的时间复杂度为O（mn），m和n分别是字符串A和B的长度。算法的空间复杂度也为 O（mn），因为需要一个二维dp数组。 | | | | | |
| 测试结果  （含运行时间） | **小规模数据**      **中规模数据**      **大规模数据** | | | | | |
| 实验心得 | 本次实验让我复习了动态规划算法。  动态规划求解具有以下的性质：  1.最优子结构性质：最优解包含了其子问题的最优解，不是合并所有子问题的解，而是找最优的一条解线路，选择部分子最优解来达到最终的最优解。  2.重叠子问题性质：先计算子问题的解，再由子问题的解去构造问题的解（由于子问题存在重叠，把子问题解记录下来为下一步使用，这样就直接可以从备忘录中读取）。其中备忘录中先记录初始状态。  经过这次实验，我对于用动态规划法求解字符距离问题的相关代码已基本熟悉，算法知识得到了复习与巩固。在写代码与调试的过程中，在解决问题过程中，丰富了个人编程的经历和经验，提高了个人解决问题的能力。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

*//字符距离问题*

#include <iostream>

#include <string>

#include <cmath>

#include <fstream>

#include <windows.h>

using namespace std;

string a, b;

int dp[1000][1000];

int dist(int i, int j)*//返回a中第i个字符 b中第j个字符的acsii值之差*

{

    return abs(a[i - 1] - b[j - 1]);

}

void fun(int len1, int len2, int k)

{

    for (int i = 0; i <= len1; ++i)*//有一个字符串是空的*

        dp[i][0] = k \* i;

    for (int i = 0; i <= len2; ++i)

        dp[0][i] = k \* i;

    for (int i = 1; i <= len1; ++i)*//递归式*

        for (int j = 1; j <= len2; ++j)

            dp[i][j] = min(dp[i - 1][j - 1] + dist(i, j), min(dp[i - 1][j] + k, dp[i][j - 1] + k));

}

int main()

{

    ifstream infile("zifujuli\_in.txt", ios::in);

    ofstream outfile("zifujuli\_out.txt", ios::out);

    infile >> a >> b;*//输入两个字符串*

    int k;

    infile >> k;

    infile.close();

    int len1 = a.size();

    int len2 = b.size();

    double time = 0;

    LARGE\_INTEGER nFreq, nBeginTime, nEndTime;

    QueryPerformanceFrequency(&nFreq);

    QueryPerformanceCounter(&nBeginTime);*//开始计时*

    fun(len1, len2, k);

    QueryPerformanceCounter(&nEndTime);*//停止计时*

    time = (double)(nEndTime.QuadPart - nBeginTime.QuadPart) / (double)nFreq.QuadPart;*//计算程序执行时间单位为s*

    outfile << dp[len1][len2] << endl;

    outfile << "程序耗时" << time \* 1000 << "ms" << endl;

    outfile.close();

    return 0;

}

*//字符距离数据生成器*

#include <iostream>

#include <fstream>

#include <time.h>

#define random(a, b) (rand() % (b - a + 1) + a)

using namespace std;

int main()

{

    ofstream outfile("zifujuli\_in.txt", ios::out);

    srand((int)time(NULL));

    int a = random(1, 50);

    int b = random(1, 50);

    char ch;

    for (int j = 0; j < a; j++)

    {

        ch = random('a', 'z');

        outfile << ch;

    }

    outfile << endl;

    for (int j = 0; j < b; j++)

    {

        ch = random('a', 'z');

        outfile << ch;

    }

    outfile << endl;

    outfile << random(1, 10) << endl;

    outfile.close();

    return 0;

}