编辑距离问题

2018年4月23日

1 题目介绍

1.1 问题描述

给定 2 个字符串 a, b. 编辑距离是将 a 转换为 b 的最少操作次数，操作只允许如下 3 种：

插入一个字符，例如：fj 变为 fxj
删除一个字符，例如：fxj 变为 fj
替换一个字符，例如：jxj 变为 fyj

1.2 算法要求

对于给定的字符串A和字符串B，计算其编辑距离d(A, B)。

1.3 数据输入输出

由文件input.txt给出输入数据。前两行分别有两个字符串A和B。将计算出的编辑距离数输出到文件output.txt。

2 算法设计

假设d(i,j)表示字符串a取前i个和字符串b取前j个时的最短编辑距离，则有：

$$d(0,0) = 0, d(1,0) = 1, d(0,1) = 1.$$

其递推公式为：

$$d(i,j) = \begin{cases} i, & \text{j=0} \\ j, & \text{i=0} \\ min\{d(i-1,j)+1, d(i,j-1)+1, d(i-1,j-1)+(a_i \neq b_j)\}, & \text{其它} \end{cases}$$

如果i和j中有一个是0，那么最短编辑距离就是max(i, j)；否则，就选择采用替换的方式，分别对替换a串末位、替换b串末位和末位是否相同三种情况来做处理。利用动态规划算法实现如下算法：

```
int editdist(string a, string b)
{
    int na = a.size();
    int nb = b.size();
    int **mat = new int *[na + 1];
    for (int i = 0; i != na + 1; i++)
        {
                mat[i] = new int[nb + 1];
        }
        // 动态规划
        mat[0][0] = 0;
        int p, q;
        for (p = 1; p != na + 1; p++)
                mat[p][0] = p;
        for (q = 1; q != nb + 1; q++)
                mat[0][q] = q;
        for (int j = 1; j != na + 1; j++)
        {
                for (int k = 1; k != nb + 1; k++)
                {
                        int Fjk = 0;
                        if (a[j - 1] != b[k - 1])
                        {
                                Fjk = 1;
                        }
```

```
                        mat[j][k] = mini(mat[j - 1][k] + 1, mat[
                            j][k - 1] + 1, mat[j - 1][k - 1] +
                            Fjk);
                }
        }
        int  nEditDis = mat[na][nb];
        for (int m = 0; m != na + 1; m++)
        {
                delete[] mat[m];
        }
        delete[] mat;
        return  nEditDis;
        }
}
```

3 算法分析

3.1 递归式

$$edit(i,j) = \begin{cases} 0, & \text{i=0, j=0} \\ i, & \text{i¿0, j=0} \\ j, & \text{i=0, j¿0} \\ min\{edit(i-1,j)+1, edit(i,j-1)+1, edit(i-1,j-1)\}, & a_i = b_j \\ min\{edit(i-1,j)+1, edit(i,j-1)+1, edit(i-1,j-1)+1\}, & a_i \neq b_j \end{cases}$$

3.2 复杂度

本题的复杂度取决于两个字符串的大小，为$O(mn)$。