算法设计与分析-work5

yu wang

May 2024

1 问题一

算法:

```
editDistance(s1,s2)
n=s1.length()
m=s2.length()
Let f[0..n][0..m] be a new array
for i=1 to n do
    f[i][0]=i
for j=1 to m do
    f[0][j]=j
for i=1 to n do
    for j=1 to m do
        if f[i-1][j-1]<f[i][j-1]
            f[i][j]=f[i-1][j-1]
        else
            f[i][j]=f[i][j-1]
        if f[i][j]>f[i - 1][j]
            f[i][j]=f[i-1][j]
        f[i][j] = f[i][j] + 1
        if A[i - 1] == B[j - 1]
            f[i][j] = f[i - 1][j - 1]
return f[n][m]
```

证明正确性:

首先我们证明该问题满足最优子结构的性质。假设我们有一个最优编辑方案，它将 s1[1.] 和 s2[1..m] 编辑为相等的两个字符串。那么我们选取针对字符串的操作中的最大下标的那一个，于是我们有三种可能，第一种可能是这是一个插入操作，第二种可能是这是一个删除操作，第三种可能这是一个修改操作。

首先我们考虑插入，如果这是一个插入操作, 说明操作到这一步时两个字符串长度是不等，那么我们对 s1 进行插入与 s2 对应的字符，把它们去掉，剩余的两个子串进行操作的次数必须是最小的，否则我们有一种对剩余的两个子串更小的操作方案, 操作完后将被去掉的字符放回后插入，使得总的操作次数更小了，于是得到了矛盾

我们考虑删除操作，如果这是一个删除操作，说明我们接下来要将该字符删除后的 s1 与 s2 相同的操作次数是最小的。否则我们就有另外的操作, 使得删除该字符后的 s1 与另 s2 做若干次操作后相同的操作次数更小, 然后我们再删除这个字符，使得总的操作次数变得更小了，于是得到了矛后。

接下来我们考虑修改操作，如果这时一个修改操作，说明当我们操作到这一步时两个字符串的长度应该是相等的了。那么我们这时候选取修改操作修改的字符和 s2 中与之对应的字符, 把它们去掉, 剩余的两个子串进行操作的次数必须是最小的, 否则我们有一种对剩余的两个子串更小的操作方案, 操作完后将被去掉的字符放回后修改，使得总的操作次数更小了，于是又得到了矛盾。因此总而言之，该问题满足最优子结构的性质。

再然后我们将解空间划分为若干个子空间。解空间可以划分为:(1) 插入 s1[n+1],(2) 删除 s1[n],(3) 修改 s1[n],(4) 对 s1[n] 不操作。其中 (3) 和 (4) 两个子空间只存在一个。当 s1[n] s2[m] 时存在解空间 (3)，当 s1[m]= s2[m] 时存在解空间 (4)。对于子空间 (1)，最短编辑距离为 f[n][m-1]+1; 对于子空间 (2)，最短编辑距离为 f[n-1][m]+1; 对于子空间 (3)，最短编辑距离为 f[n-1][m-1]+1; 对于子空间 (4)，最短编辑距离为 f[n-1][m -1]。

下面我们我们来证明算法的正确性:

首先，确定谓词 P（n）:

.P(n)：该算法能够求解出字符串 s1 前 n 个字符与字符串 s2 相同的最少操作次数

第二步是证明基本情况 P（0）和 P（1）：当数组为空时，操作数为 0, 当只有一个字符时，根据上面的四种子空间情况，最小操作次数不是 0 就是 1。

第三步是证明一般情况 $\forall n \in N(P(0) \wedge P(1) \wedge \cdots \wedge P(n) \Rightarrow P(n+1))$，假设对于任意字符个数为 0,1,2，...,n 的字符串 s1 的子串，该算法都能够求解出字符串 s1 前 n 个字符与字符串 s2 相同的最少操作次数。在 n+1 长的字符串，对于第 n+1 个字符，如果字符不相同，那么会进行插入修改或者删除的操作，不同的操作求解出的前 n 个字符相同的最小操作数不同，我们取最小的操作数为 x，则 n+1 长的字符串最小操作数可以记作为 x+1, 否则，不需要进行操作，求出最小的操作数为 y，比较 x+1 和 y 得出最小的操作数。综上，我们可以证明该算法能够求解出字符串 s1 前 n 个字符与字符串 s2 相同的最少操作次数。

时间复杂度为:$\Theta(n \times m)$

n 和 m 是 s1，s2 字符串的长度，在调用这个函数时使用了双层循环填充二维数组，每个位置填充都需要常数时间操作，需要 nxm, 对于其他的初始化等时间可以忽略不计