思考题 6——编辑距离

孟妍廷　2015202009

2017 年 11 月 2 日

解: 分析: 利用动态规划, 假设当前操作之前的所有操作都满足编辑距离最小, 分析当前操作:
用 f(i,j) 表示当前最小编辑距离, 对于 x[1...m],y[1...n] 根据题意
⑴ 若当前操作为 copy, 则可知此时 x[i]=y[j], 则 f(i,j)=f(i-1,j-1)+copy
⑵ 若当前操作为 replace, 则可知 $x[i] \neq y[j]$, 则 f(i,j)=f(i-1,j-1)+replace
⑶ 若当前操作为 delete, 则 f(i,j)=f(i-1,j)+delete
⑷ 若当前操作为 insert, 则 f(i,j)=f(i,j-1)+insert
⑸ 若当前操作为 twiddle, 则可知 y[j]=x[i-1],y[j-1]=x[i], 则 f(i,j)=f(i-2,j-2)+teiddle
⑹ 若当前操作为 kill, 则 f(i,j)=f(i-1,j-1)+kill+(n-i+1)$times$insert//回到前一状态, 不知是否有误
⑺ 特殊情况为:x,y 中有一个为空串, 还有根据例子可以看到的回溯问题
算法如下:

$solution(x, y, op)$//考虑到回溯问题, 需要记录操作是插入还是复制
  $for\ i = 0\ tom$
    $f[i, 0] = i \times delete$
    $p[i, 0] = delete$
  $for\ j = 0\ to\ n$
    $f[0, j] = j \times insert$
    $p[0, j] = insert$
  $for\ i = 1\ to\ m$
    $for\ j = 1\ to\ n$
      $f[i, j] = \infty$
      $if\ x[i] == y[j]\ and\ op == copy$
        $f[i, j] = f[i - 1, j - 1] + copy$
        $p[i, j] = copy$
      $if\ x[i] \neq y[j]\ and\ f[i - 1, j - 1] + replace < f[i, j]$
        $f[i, j] = f[i - 1, j - 1] + replace$
        $p[i, j] = replace$
      $if\ f[i - 1, j] + delete < f[i, j]$
        $f[i, j] = f[i - 1, j] + delete$
        $p[i, j] = delete$
      $if\ f[i, j - 1] + insert < f[i, j]\ or\ op == insert$//存在两种方式, 导致不同结果
        $f[i, j] = f[i, j - 1] + insert$
        $p[i, j] = insert$
      $if\ i \geq 2\, and\, j \geq 2\, and\, x[i-1] = y[j]\, and\, x[i] = y[j-1]\, and\, f[i-2, j-2]+twiddle < f[i, j]$
        $f[i, j] = f[i - 2, j - 2] + twiddle$
        $p[i, j] = twiddle$
  $for\ i = 0\ to\ m - 1$
    $if\ f[i, n] + kill + (n - i) \times insert < f[m, n]$
      $f[m, n] = f[i, n] + kill + (n - i) \times insert$
      $p[i, n] = kill$
      $rem = i$
  $return\ f\ op$

$print(p, i, j, rem)$
  $if\ i = 0\ and\ j = 0$
    $return$
  $if\ p[i, j]\ =\ copy\ or\ p[i, j]\ =\ replace$
    $i_1 \leftarrow i - 1$

$j_!\leftarrow j-1$
$elseifp[i,j]=delete$
$i_1=i-1$
$j_1=j$
$else\ if\ p[i,j]\ =\ insert$
$i_1=i$
$j_1=j-1$
$else\ if\ p[i,j]=twiddle$
$i_1=i-2$
$j_1=j-2$
$else$
$p[i,j]=kill$
$i_1=rem$
$j_1=j$
$print(p,i_1,j_1)$
$printf\ p[i,j]$

b. 最右对齐问题可改变为:
x[i]=y[i] 且都不是空格相当于复制操作的代价为 +1
$x[i]\neq y[j]$ 相当于替换操作的代价为-1
x[i] 或 y[j] 为空格相当于 insert 操作的代价为-2
在算法中删去旋转和删除操作即可。