5-a

大概作法是

dp(I,j) = min(dp[i-1,j-1]+(str1[i] == str2[j]?cost(copy):cost(replace)), dp[i-1][j]+cost(delete), dp[i][j-1]+cost(insert) , dp[i-2][j-2]+cost(twiddle))

kill的話我透過zvalue紀錄dp 從str1 到 str2 完成幾個字元了

若zvalue == string2.length 而且I != string1.length 則kill

Pseudocode:

function editDistance(str1, str2)

dp(i,0) = i

dp(0,j)=j

for each I = 1 to len(str1)

substr1 = str1[i-1]

for each j = 1 to len(str2)

substr2 = str2[j-1]

nowmin = 9999

if zvalue[i-1][j] == len(str2)

zvalue[i][j] = zvalue[i-1][j]

step[i][j] = step[i-1][j]

nowmin = dp[i-1][j]-costList["kill"]

if substr1 == substr2

if dp[i-1][j-1]+costList["copy"] <= nowmin

zvalue[i][j] = zvalue[i-1][j-1]+1

step[i][j] = step[i-1][j-1]+"1"

nowmin = dp[i-1][j-1]+cost(copy)

if i > 1 and j > 1 and substr1 == str2[j-2] and substr2 == str1[i-2]

if dp[i-2][j-2]+costList["twiddle"] <= nowmin

zvalue[i][j] = zvalue[i-2][j-2]+2

step[i][j] = step[i-2][j-2]+"2"

nowmin = dp[i-2][j-2]+ cost(twiddle)

if substr1 != substr2

if dp[i-1][j-1]+costList["replace"] <= nowmin

zvalue[i][j] = zvalue[i-1][j-1]+1

step[i][j] = step[i-1][j-1]+"3"

nowmin = dp[i-1][j-1]+ cost(replace)

if dp[i-1][j]+costList["delete"] <= nowmin

zvalue[i][j] = zvalue[i-1][j]

step[i][j] = step[i-1][j]+"4"

nowmin = dp[i-1][j]+cost(delete)

if dp[i][j-1]+costList["insert"] <= nowmin

zvalue[i][j] = zvalue[i][j-1]+1

step[i][j] = step[i][j-1]+"5"

nowmin = dp[i][j-1]+cost(insert)

if zvalue[i][j] == len(str2) and i != len(str1) and step[i][j][-1] != "6"

step[i][j] = step[i][j]+"6"

if step[i][j][-1] == "6":

nowmin = nowmin+cost(kill)

dp[i][j] = nowmin

function printxstring(xindex)

for each i=0 to len(str1)

if i == xindex

print("[", end='')

print(str1[i], end='')

if i == xindex

print("]", end='')

function printFinalSetp(step)

for i in step[len(str1)][len(str2)]

if i == "1":

zstring = zstring+str1[xindex]

xindex += 1

zindex += 1

print("copy "

elif i == "2":

zstring = zstring + str1[xindex+1]+str1[xindex]

xindex += 2

zindex += 2

print("twiddle ")

elif i == "3"

zstring = zstring + str2[zindex]

xindex += 1

zindex += 1

print("replace ")

elif i == "4"

xindex += 1

print("delete ")

elif i == "5"

zstring = zstring + str2[zindex]

zindex += 1

print("insert ")

elif i == "6"

zstring = zstring

xindex = len(str1)+1

print("kill ")

if i == "2"

printxstring(xindex-2)

else

printxstring(xindex - 1)

print(" ")

print(zstring)

時間複雜度:因為也還是要跑過找個string1 跟string2 把 dp array建起來所以還是O()

空間複雜度：dp(string1.length,string2.length), step(string1.length,string2.length),zvalue(string1.length,string2.length)

所以是3\*string1.length\*string2.length

5-b

定義：

cost(delete)=cost(insert)=2,

cost(copy) = 1

cost(replace) = 1,cost(replace)=1,

cost(kill) =cost(twiddle) = ∞

然後套到剛剛的editDistance 一樣找出最小