**編輯距離計算字串的相似度**

問題：輸入一個長度為n的單詞，給定一個字典包含m筆長度不等的字串，設計一個拼寫檢查器來檢查輸入單詞是否拼寫正確。如果單詞拼寫錯誤，請列出其最近的單詞，其中計算的編輯距離小於或等於給定字典中的3，以編輯距離的升序列出單詞。比對設有以下幾種操作：替換（R），插入（I）和刪除（D）且都是相同的操作。尋找到轉換一個字串插入到另一個需要修改的最小數量。

1. **定義**

查詢的字串是模糊的，但字典中的是準確的，可以用編輯距離計算一下字串之間的相似度，是指兩個字串之間，由一個轉換成另一個所需的最少編輯操作次數。

編輯距離又稱Leveinshtein距離，是由俄羅斯科學家Vladimir Levenshtein提出。編輯距離是計算兩個文本相似度的演算法之一，以字串為例，字串a和字串b的編輯距離是將a轉換成b的最小操作次數，這裡的操作包括三種：

插入一個字元

刪除一個字元

替換一個字元

舉個例子，kitten和sitting的編輯距離是3，kitten -> sitten(k替換為s) -> sittin(e替換為i) -> sitting(插入g)，至少要做3次操作。

**二、演算法原理**

該演算法的解決是基於動態規劃的方法，具體如下：   
設 s 的長度為 n，t 的長度為 m。

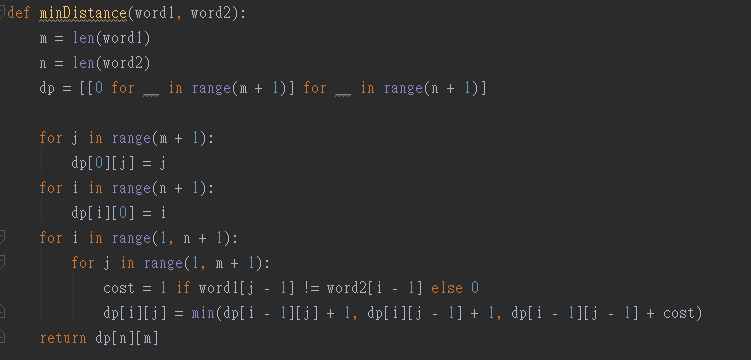
構建一個陣列 d[0..m, 0..n]。   
將第0行初始化為 0..n，第0列初始化為0..m。

那麼邊界條件：

    d[i][0]=i, 0=<i<=m  
    d[0][j]=j, 0=<j<=n

依次檢查 s 的每個字母(i=1..n)。   
依次檢查 t 的每個字母(j=1..m)。   
如果 s[i]=t[j]，則 cost=0；如果 s[i]!=t[j]，則 cost=1。將 d[i,j] 設置為以下三個值中的最小值：   
緊鄰當前格上方的格的值加一，即 d[i-1,j]+1   
緊鄰當前格左方的格的值加一，即 d[i,j-1]+1   
當前格左上方的格的值加cost，即 d[i-1,j-1]+cost   
重複上述檢查步驟直到迴圈結束。d[n,m]即為編輯距離。

程式實作如下:



1. **執行結果**
2. **討論**