國立臺南大學資訊工程學系

資工三「演算法」課程

第四次作業

**題目: Plagiarism Detection based on**

**Edit Distance**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 黃詩豪 |
| 學號 | ： | S10259002 |

老師：陳宗禧

中華民國 104年12月27日

# 目錄

1. **簡介及問題描述…………………………………………………………………1**
   1. **簡介………………………………………………………………………………………2**
   2. **問題……………………………………………………………………………………4**
2. **理論分析……………………………………………………………………………8**
3. **演算法則…………………………………………………………………………10**
4. **程式設計環境架構.………………………………………………………………12**
5. **程式.………………………………………………………………………………14**
6. **執行結果、討論與心得…………………………………………………………18**

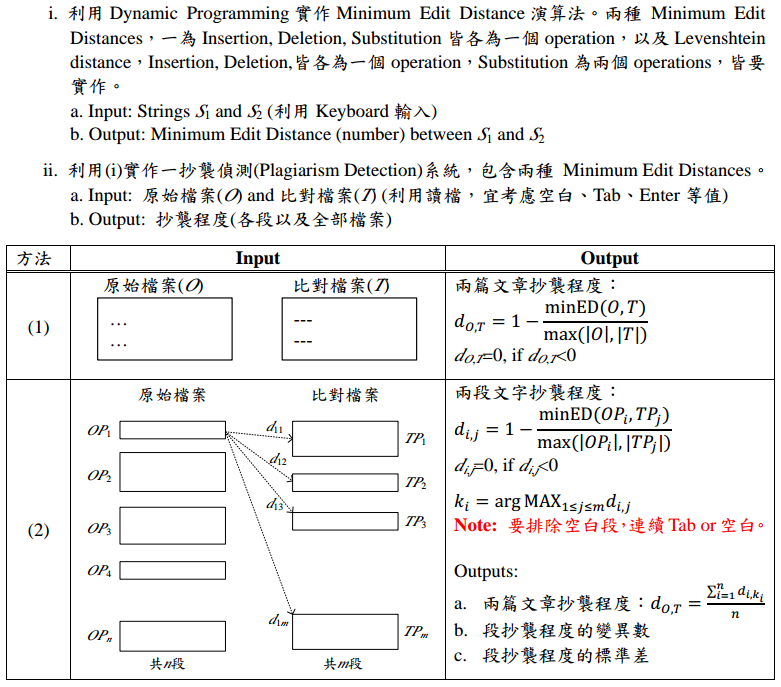
參考文獻………………………………………………………………………………22

**(一) 簡介及問題描述**

1. 簡介

利用 Edit Distance 實作一抄襲偵測(Plagiarism Detection)系統

2. 問題



**(二) 理論分析**

1. 利用動態規劃（Dynamic Programming）實作Minimum Edit Distance演算法

定義Edit Distance：有三種操作要計算，分別是Insertion與Deletion且視為一次的操作，另外，Substitution可視為一次或兩次（Levenshtein）的操作。底下為動態規劃中遞迴的關係，s1與s2表兩字串，其中n = s1.length, m = s2.length：

最後的D(n, m)即為s1與s2的Minimum edit distance。因為使用動態規劃的關係，不重複計算已計算過的情況，且使用表格記錄，因此時間複雜度為O(nm)，空間複雜度為O(nm)。

**(三) 演算法則**

1. 第一個演算法(Algorithm)

**ALGORITHM** *minimumEditDis*(origin[s0s1s2..sn-1], test[s0s1s2..sm-1])

//**Input:** origin and test are strings, with length n and m respectively

//**Output:** minimum edit distance

*table*[*n,m*] ← { *0* }

**for** *i* ← *0* **to** *n* **do**

*table*[*i,* *0*] ← *i*

**for** *j* ← 0 **to** m **do**

*table*[*0, j*] ← *j*

**for** *i* ← *1* **to** *n* **do**

**for** *j*← *1* **to** *m* **do**

*table*[*i, j*] ← min(*table*[*i-1, j*] + *1*, *table*[*i, j-1*] + *1*, *table*[*i-1, j-1*] + *1*)

**return** *table*[*n, m*]

1. 演算法時間複雜度(time complexity)

建立一個表格，初始化表格，需要O(n)與O(m)時間，計算edit distance需要O(nm)，故時間複雜度為O(nm)，n與m為字串的字元數量

1. 演算法空間複雜度(space complexity)

程式執行過程會因為輸入的字串長度不同，所以會動態分配記憶體，故空間複雜度為O(nm)，其中n與m為字串的字元數量

2. 第二個演算法(Algorithm)

**ALGORITHM** *levenshteinEditDis*(origin[s0s1s2..sn-1], test[s0s1s2..sm-1])

//**Input:** origin and test are strings, with length n and m respectively

//**Output:** minimum edit distance

*table*[*n,m*] ← { *0* }

**for** *i* ← *0* **to** *n* **do**

*table*[*i,* *0*] ← *i*

**for** *j* ← 0 **to** m **do**

*table*[*0, j*] ← *j*

**for** *i* ← *1* **to** *n* **do**

**for** *j*← *1* **to** *m* **do**

*table*[*i, j*] ← min(*table*[*i-1, j*] + *1*, *table*[*i, j-1*] + *1*, *table*[*i-1, j-1*] + *2*)

**return** *table*[*n, m*]

1. 演算法時間複雜度(time complexity)

建立一個表格，初始化表格，需要O(n)與O(m)時間，計算edit distance需要O(nm)，故時間複雜度為O(nm)，n與m為字串的字元數量

1. 演算法空間複雜度(space complexity)

程式執行過程會因為輸入的字串長度不同，所以會動態分配記憶體，故空間複雜度為O(nm)，其中n與m為字串的字元數量

**(四) 程式設計環境架構**

程式設計語言、工具、環境與電腦硬體等規格說明…

1. 程式語言

C in Linux GCC

OS: Linux Mint 17.2 "Rafaela" – Cinnamon (64-bit)

2. 程式開發工具

Eclipse，Version: Mars Release (4.5.0)

Eclipse C/C++ Plugin: CDT 8.8.0 for Eclipse Mars

Compiler: Linux GCC

3. 電腦硬體

CPU: Intel Xeon E3-1231 v3，Main Memory: 32GB

**(五) 程式 (含source code, input code, and output code)**

程式含source code, input code, and output code等…

1. 主程式

見附件Source\_code.zip

2. Input Code Format

使用老師給的測試文字檔

(1) TestFile1.txt，TestFile11.txt，TestFile12.txt

(2) TestFile2.txt，TestFile21.txt，TestFile22.txt

3. Output Code Format

(1)

|  |  |  |
| --- | --- | --- |
|  | TestFile11.txt | TestFile12.txt |
| TestFile1.txt | 方法一：  使用第一個演算法的抄襲率：0使用第二個演算法的抄襲率：0花費時間： 3.8 (ms)  方法二：  使用第一個演算法的抄襲率：0.184  變異數：0.017  標準差：0.13  使用第二個演算法的抄襲率：0.074  變異數：0.003  標準差：0.055  花費時間：4.6 (ms) | 方法一：  使用第一個演算法的抄襲率：0.793使用第二個演算法的抄襲率：0.772花費時間： 2.8 (ms)  方法二：  使用第一個演算法的抄襲率：0.667  變異數：0.222  標準差：0.471  使用第二個演算法的抄襲率：0.667  變異數：0.222  標準差：0.471  花費時間：28.7 (ms) |

(2)

|  |  |  |
| --- | --- | --- |
|  | TestFile21.txt | TestFile22.txt |
| TestFile2.txt | 方法一：  使用第一個演算法的抄襲率：0.921使用第二個演算法的抄襲率：0.921花費時間： 0 (ms)  方法二：  使用第一個演算法的抄襲率：0.733  變異數：0.141  標準差：0.375  使用第二個演算法的抄襲率：0.733  變異數：0.141  標準差：0.375  花費時間：0 (ms) | 方法一：  使用第一個演算法的抄襲率：0.237使用第二個演算法的抄襲率：0.171花費時間： 0 (ms)  方法二：  使用第一個演算法的抄襲率：0.56  變異數：0.18  標準差：0.424  使用第二個演算法的抄襲率：0.511  變異數：0.214  標準差：0.463  花費時間：0.8 (ms) |

**(五) 執行結果、討論與心得**

執行結果與討論 (執行時間、problem *n*的大小等問題討論)等…

1. 執行結果

Output of program:

分別輸出兩種抄襲系統方法，方法一輸出抄襲率，方法二抄襲率、變異數與標準差

2. 討論

執行時間、問題大小等問題討論

1. Running Time

使用方法一的抄襲系統是整個文件計算edit distance，所以抄襲率可能會因為編寫或跳行、空格等問題影響，因此開發方法二，使用段落比對（一句一句比對），然後從每對中選出抄襲率最高的表示那一句話的抄襲率，所以抄襲率相較於方法一會提升，但是所花費的時間較久，因為是每一句一句比對，需要加上O(nm)的每對句子計算edit distance的時間。

(2) Problem size *n*

當比對的字串越長時，所需要的時間與額外的空間會越高，因為需要O(nm)的空間儲存計算結果，且時間也是O(nm)，所以n或m越高所花的時間與空間也越高。

3. 心得

這次的作業當中，minimum edit distance可以比對兩字串的相似度，對於在抄襲系統上很有幫助，可以初步判斷是否以複製的方式抄襲，對於學生間的作業或是考試都可以使用，也就可以判斷是否有抄襲的情況發生，不過抄襲系統還需要改進，因為語意上的分析在此無法做到，但可以改進的地方比較像是文字的格式化與解析，例如：連續的空白，跳行等，希望未來將文字的格式化與解析做進去，以增加系統的準確度。

**參考文獻**

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms," Third Edition, The MIT Press, 2009.
2. R.C.T. Lee, S.S. Tseng, R.C. Chang, and Y.T.Tsai, "Introduction to the Design and Analysis of Algorithms," McGraw-Hill, 2005.
3. Anany V. Levitin, "Introduction to the Design and Analysis of Algorithms," 2nd Edition, Addison Wesley, 2007.
4. Richard Neapolitan and Kumarss Naimipour, "Foundations of Algorithms," Fourth Edition, Jones and Bartlett Publishers, 2010.