Data Mining Project #1

Association Rules

資工所 葉修宏　P76054494

醫資所 楊馥謙　Q56054027

■ 程式清單

1. prepare_data.py

   (將.data 資料轉成以交易編號 group 起來的格式)

2. prepare_data2CSV.py

   (將上個程式產生的結果，再轉成 Weka 可讀的 CSV 檔)

3. fp.py (FP-growth 主程式檔)

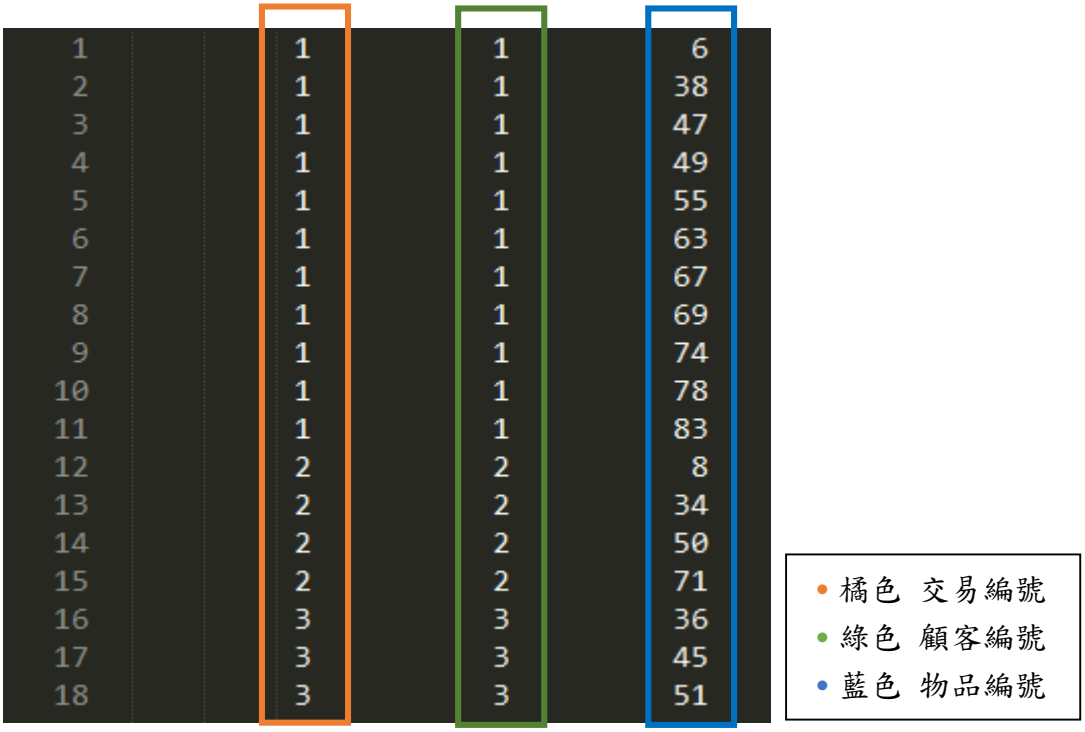
4. AprioriHash.java (Hash Tree 主程式檔)

■ 目的:

主要從交易資料中找出 frequent 的 association rules，並且比較在執行 Apriori Algorithm 使用各種不同的 Data structure 的差異，最後再利用資料探勘分析工具(Weka)與前者做比較。

■ 資料產生(Data Generator)

利用 IBM Generator 產生我們的交易資料，而產生的檔案格式如下圖(一)所示



圖(一) IBM Generator 生成資料

利用程式將資料格式做轉換，變成以交易編號為 Group，後面串接著該次交易所購買的物品，而產生的結果如圖(二)下：

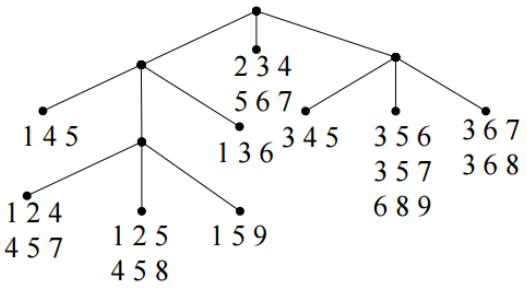
```
1   6 38 47 49 55 63 67 69 74 78 83
2   8 34 50 71
3   36 45 51 52 61 62 63
4   4 5 6 7 17 35 36 39 41 42 52 53 55 73 78 81 85 87
5   9 11 18 35 38 42 63 69 81 93 97
6   8 21 25 39 40 43 59 80
7   7 8 10 14 38 60 69 83 85 93
8   8 17 36 38 43 63 83
9   8 38 45 57 62 73 85
10  0 7 11 12 15 23 26 29 46 50 63 73 85 86 91 92
```

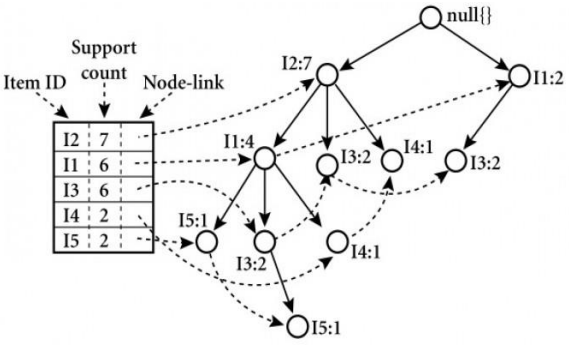
為交易編號 4 的所有物品的編號

圖(二)轉換後資料

■ 演算法-資料結構(Data Generator)

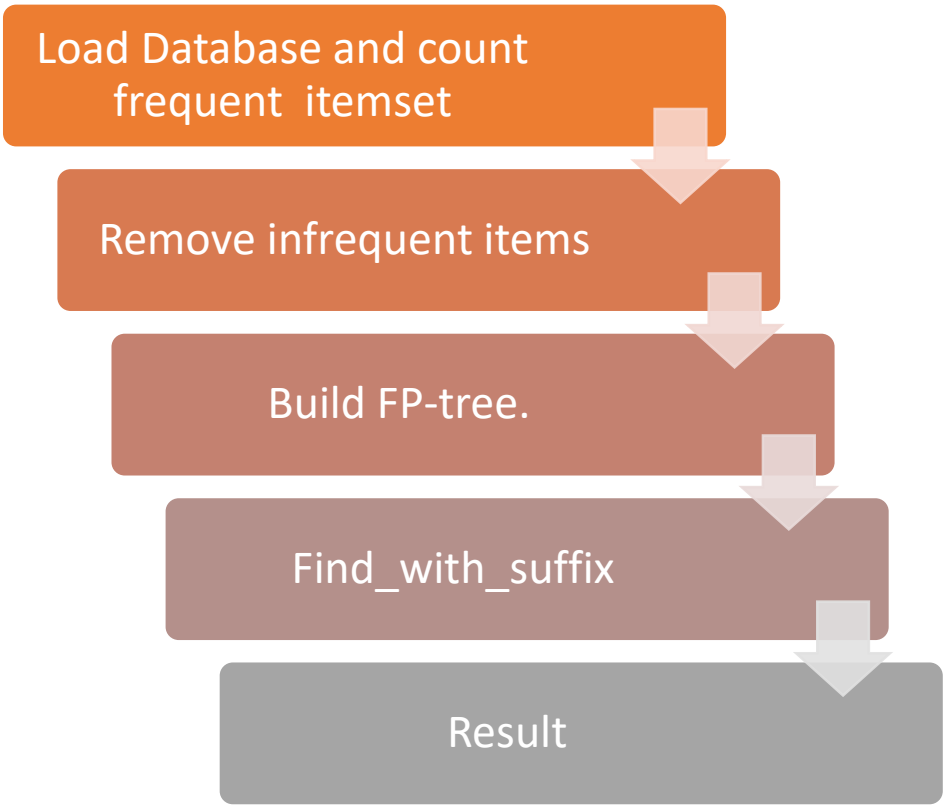
1. Hash Tree
2. FP-growth





- Hash Tree :

- FP_Growth :



■ 使用 Data mining Tool (Weka)

因為 weka 有限制讀寫的檔案規格(*.arff 及*.csv)，所以我們採用最好處理的 csv 來實作並且依照 Weka 的限制做一些更改(Weka 限制每一筆交易的長度都要等長，所以我們將未購買的物品填空值，有購買的物品填入字元”T”)，如下圖所示：

Item_id = 0	Item_id = 1	……..	Item_id = 98	Item_id = 99
T		…….		T
	T			

轉換結果：

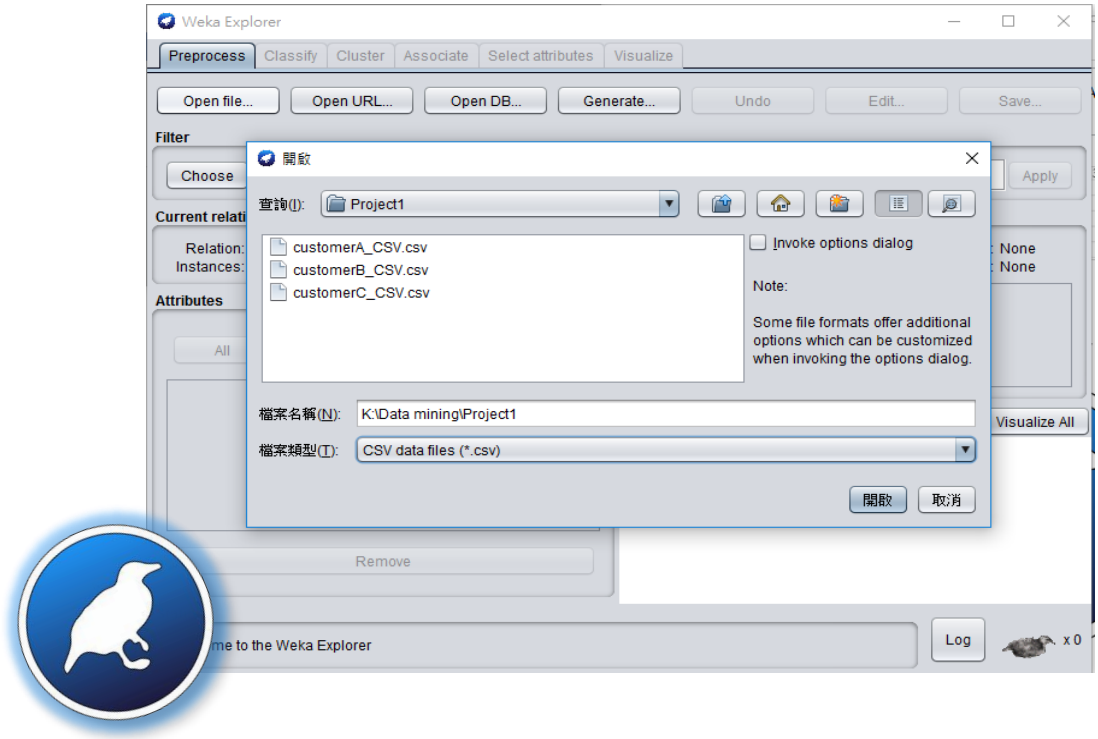
```
6 38 47 49 55 63 67 69 74 78 83
8 34 50 71
36 45 51 52 61 62 63
4 5 6 7 17 35 36 39 41 42 52 53 55 73 78 81 85 87
9 11 18 35 38 42 63 69 81 93 97
8 21 25 39 40 43 59 80
7 8 10 14 38 60 69 83 85 93
8 17 36 38 43 63 83
8 38 45 57 62 73 85
0 7 11 12 15 23 26 29 46 50 63 73 85 86 91 92
```

```
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99
,,,,,,T,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,T,,,,,,,,,T,,T,,,,,,T,,,,,,,T,,,T,,T,,,,,T,,,,T,,,,T,,,,,,,,,,,,,,,
,,,,,,,,T,,,,,,,,,,,,,,,,,,,,,,,,,,,T,,,,,,,,,,,,,,,,,T,,,,,,,,,,,,,,,,,,,,,,T,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
```

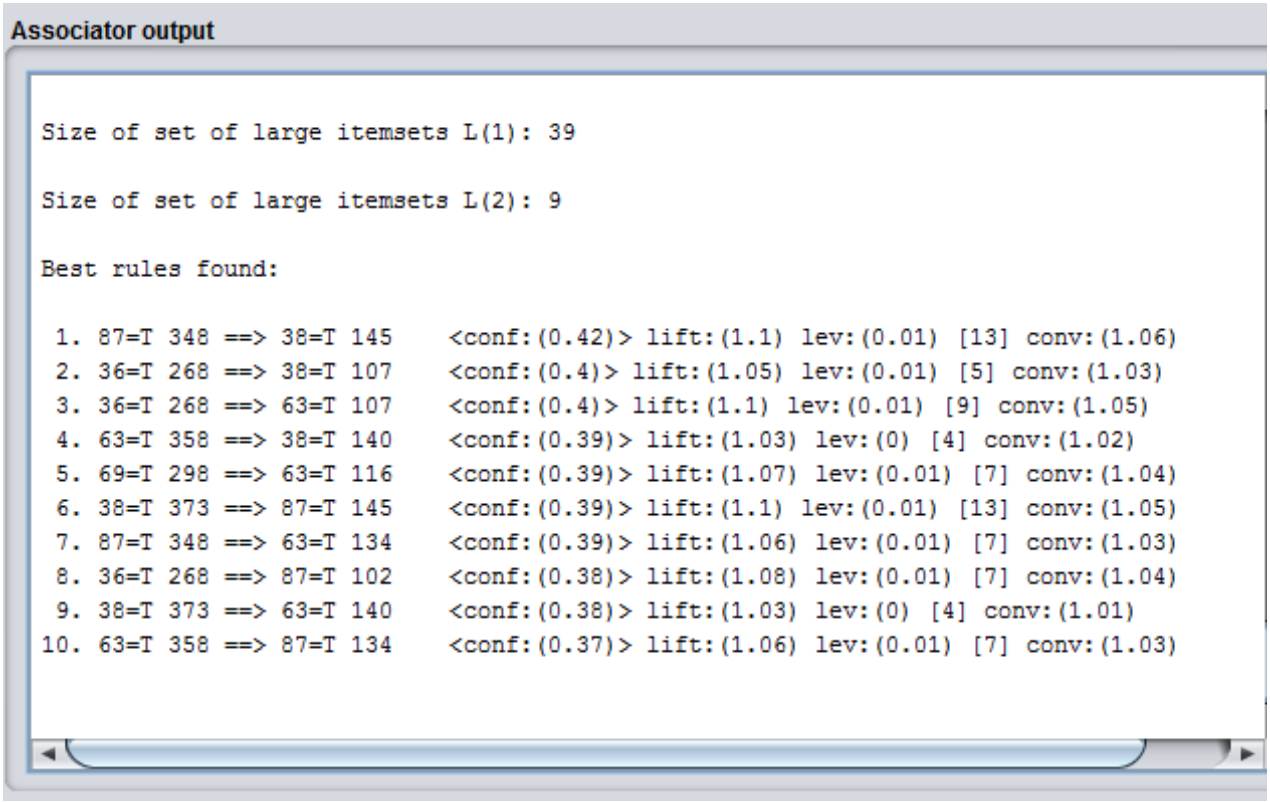
Excel 顯示結果：

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
2							T							
3									T					
4														
5					T	T	T	T						
6										T		T		

打開 Weka 將我們剛剛處理好的 Data 匯入

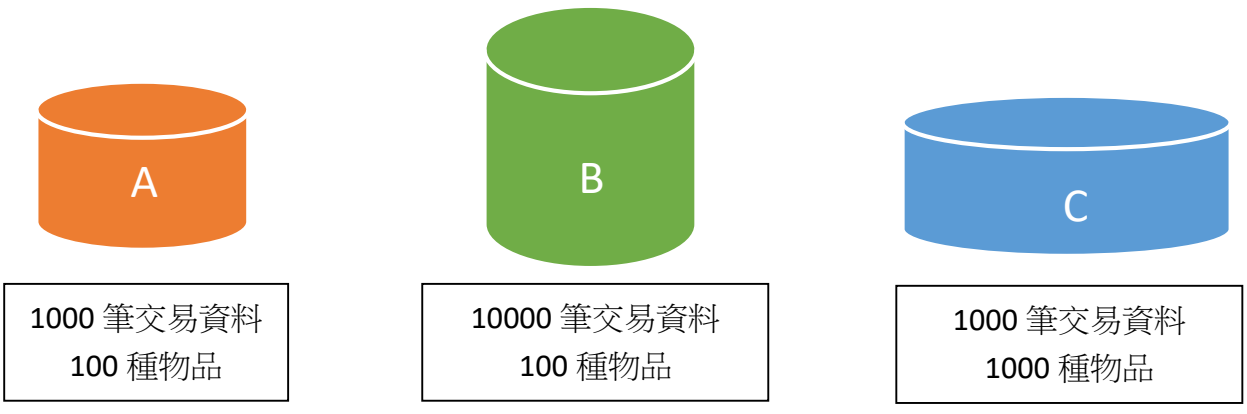


產生結果如下圖



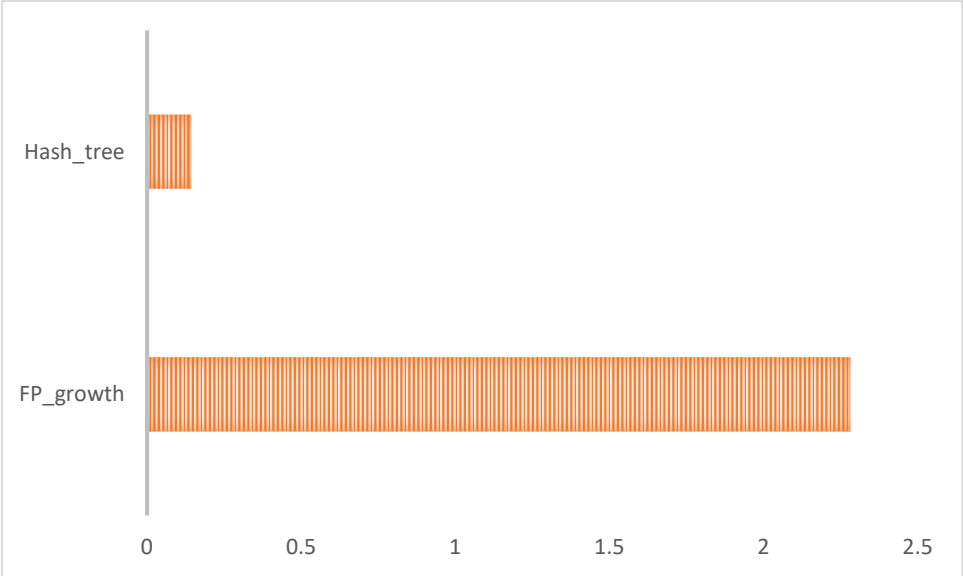
■ 結果比較

我們產生三種不同的 Dataset 來幫我們評估 FP-growth 以及 Hash Tree 的效能



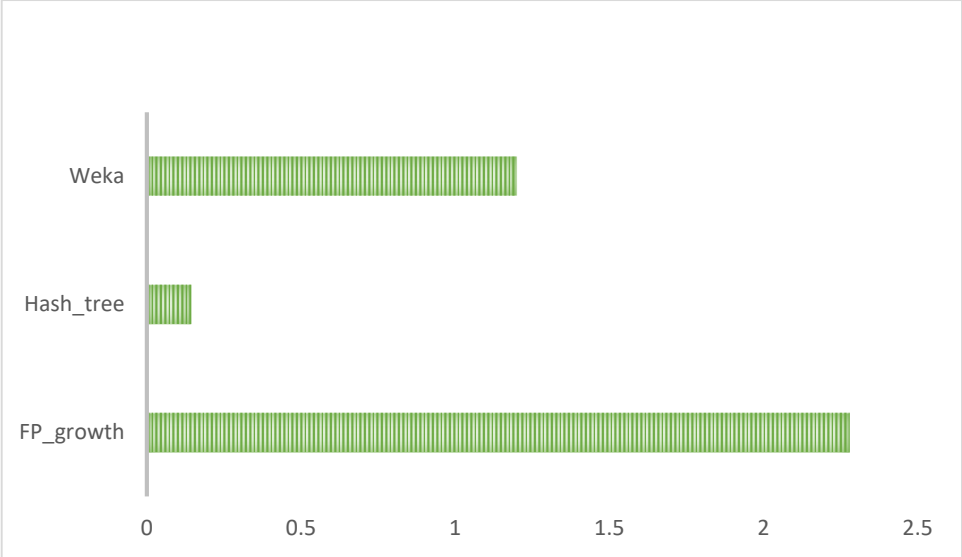
Min_sup / Dataset	0.1	0.05
FP-growth		
A	0.22 s	0.42 s
B	2.06 s	4.56 s
C	0 s	0 s
Hash Tree		
A	0.039 s	0.094 s
B	0.105 s	0.417 s
C	0 s	0 s

FP-growth 和 Hash Tree 在各種不同 dataset 以及 不同的 min_sup 所運行的時間



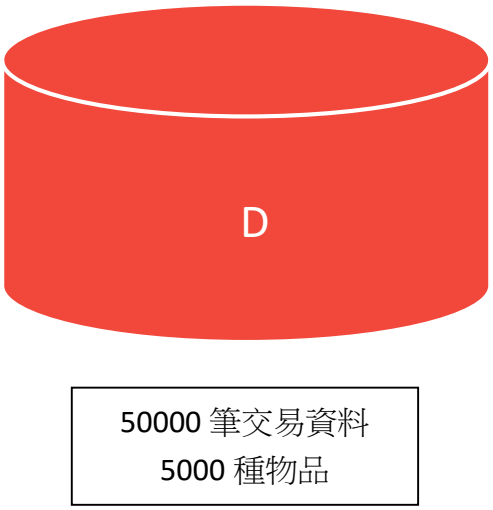
FP-growth 和 Hash Tree 同時運行三個 dataset 的總時間

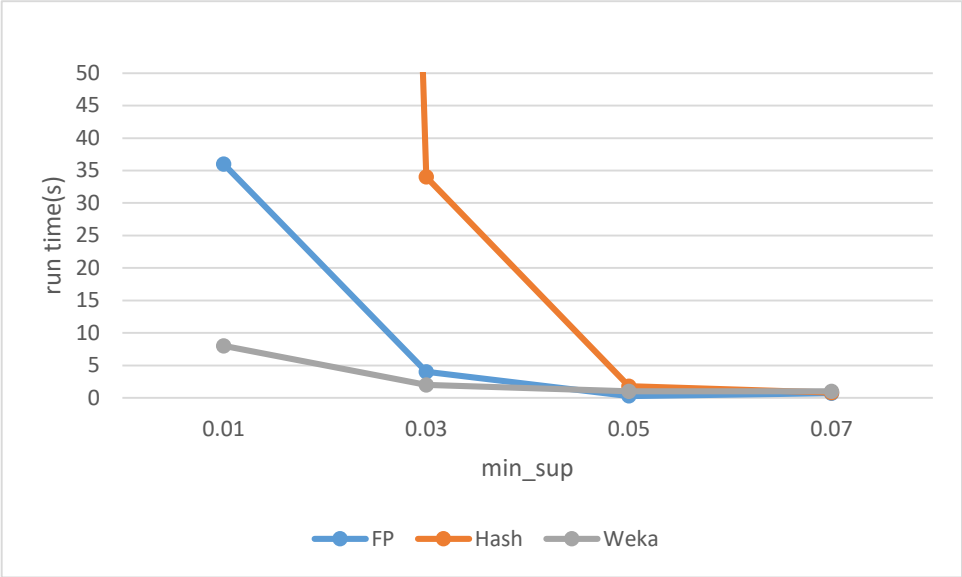
我們把 Weka 所處理的時間也加進去做分析



FP-growth 、 Hash Tree 和 Weka 同時運行三個 dataset 的總時間

由上述各種數據顯示，Hash Tree 的方法在這三個 Dataset 執行時間都是最短的，而為了再做進一步的觀察，我們產生了更大的 Dataset，來探討 FP-growth 是否在這三個處理關聯法則的方法中，表現一無是處。





FP-growth	36 s	Hash Tree	無法執行	Weka	8 s

表一、Min_sup = 0.01 時 ，三種方法的執行時間

根據較大的 Dataset 的結果顯示，我們可以發現原本在 DatasetA、DatasetB 和 DatasetC 執行時間都是最快的 Hash Tree，在 Dataset D Min_Sup = 0.01 時，無法執行，在 Min_Sup = 0.03 所運行的時間是三個方法中最慢的。

■ 結論

經過這次實驗我們可以知道，Hash Tree 在資料量較小的情況下，時間都是最短的，一旦資料量較龐大的時候，Hash Tree 的執行時間呈現指數上升，甚至會因為記憶體限制無法執行，反觀在前面三個 Dataset (A、B 及 C) 表現最差的 FP-growth，在 Dataset D 表現得還不錯，因為 FP-growth 犧牲了時間，建立了較複雜的樹狀結構，來降低記憶體的使用率，以致 FP-growth 對於 Database 有很好的壓縮率，故 FP-growth 適用處理 large Dataset，而降低記憶體的使用率對於 Data mining 是一件非常重要的議題，通常 Data mining 資料都是非常大量，所以我們必須利用所擁有的硬體設備資源與時間去取得平衡，才能達到我們預期的結果。至於 Weka 在這次的實驗中，表現也不遜色，但是 weka 的缺點就是需要了解各種 model 它所接受的 input 的格式，並且 weka 在處理龐大的資料上，若採用比較複雜的 model，所耗費的記憶體量是純程式的好幾倍，所以 weka 通常用來做一般簡易的基礎分析。

------ 以上是為我們 Project1 的報告 ------