生科大四　　C54046238　　李承學

一、 執行環境

1. 作業系統：Ubuntu 18.04
2. Python 版本：Python 3.6.6

二、 程式檔案

1. bruteforce.py

   利用暴力法搜尋 frequent itemset

2. apriori.py

   利用 Apriori 搜尋 frequent itemset

3. apriori_hashtree.py

   利用 Apriori 搜尋 frequent itemset，其中利用 hash tree 來計算 candidate 出現的次數

4. fp.py

   利用 FP growth 搜尋 frequent itemset

以上程式執行要加參數，參數的格式完全相同

python3 bruteforce.py file [-t threshold]

file 為輸入的檔案，格式為 csv 檔，以下有資料的說明

threshold 是 0 - 100 之間的一個浮點數，代表 minimum support 設定的%數，例如有 1000 個 transactions，設定 5 代表 minimum support 為 1000 的 5%，即為 50。在未設定這項參數下的 threshold 預設為 10%。執行結果會直接印在螢幕上，若要存成檔案，要用>重導向至檔案。下面的測試結果就不放暴力法了，因為要執行很久

三、 Generator 資料

資料位於 data/generator/，程式不能直接讀取，要先轉換格式，執行

python3 data/generator/convert_generator.py file

file 為任意一個 generator 產生的 data 檔案，執行完可看到在該路徑下多了一個 csv 檔，即可作為程式輸入的檔案

四、 kaggle 資料

1. 資料下載於 https://www.kaggle.com/xvivancos/transactions-from-a-bakery
2. 輸入以下指令將 kaggle 的資料 data/kaggle/BreadBasket_DMS.csv 轉換格式

cd data/kaggle/

python3 convert.py BreadBasket_DMS.csv bakery.csv

3. 執行完會在 data/kaggle/底下得到另一個檔案 bakery.csv，分別輸入以下指令來使用 Apriori、Apriori hash tree、FP growth 尋找 frequent itemset

python3 apriori.py data/kaggle/stocks.csv -t 0.2 > result/kaggle/apriori

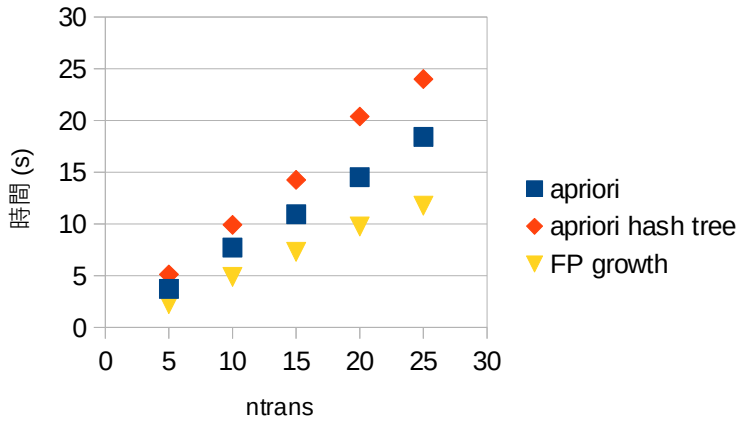
python3 apriori_hashtree.py data/kaggle/stocks.csv -t 0.2 > result/kaggle/apriori_hashtree

python3 fp.py data/kaggle/stocks.csv -t 0.2 > result/kaggle/fp

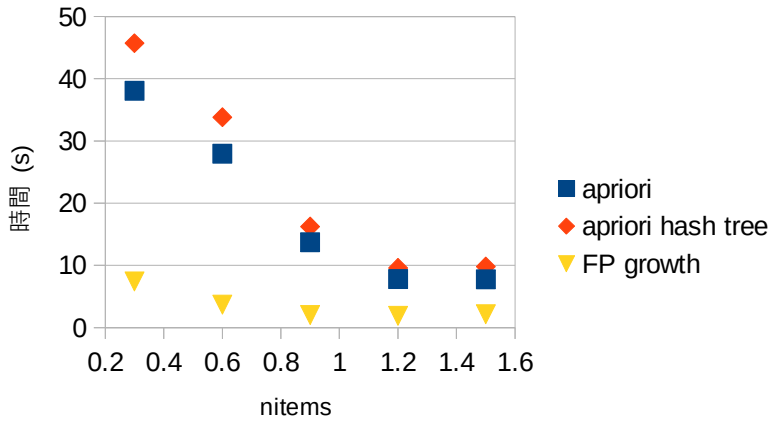
4. 完成後利用 diff 檢查，3 個檔案完全相同

五、 結果

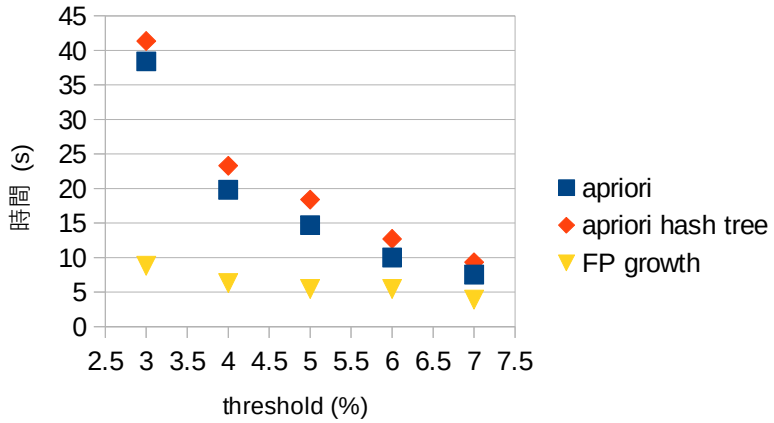
1. 測試不同 transaction 數量的執行時間，資料位於 data/generator/ntrans



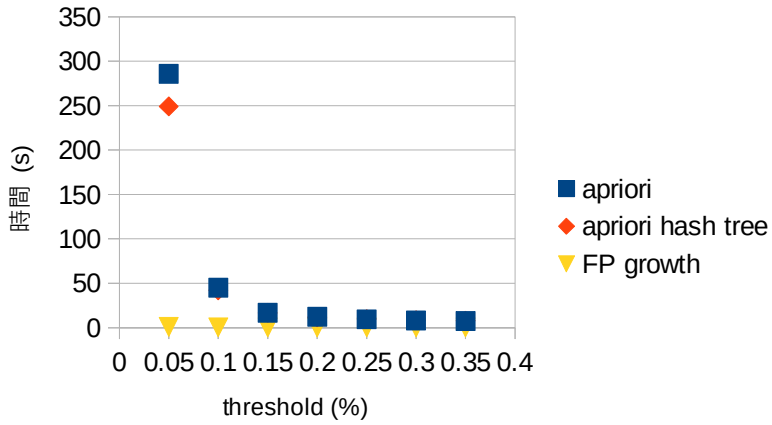
2. 測試不同 item 數量的執行時間，資料位於 data/generator/nitems



3. 測試不同 threshold 的執行時間，資料位於 data/generator/threshold



4. 測試不同 threshold 的執行時間，這邊用 kaggle 的資料，資料位於 data/kaggle/



六、 結論

1. FP growth 比 Apriori 快很多
2. 雖然 apriori 用 hash tree 可以增加比對 candidate 的效率，但是使用 generator 產生的資料來測試，使用 hash tree 有時反而會增加執行的時間