T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY of ENGINEERING
DEPARTMENT of COMPUTER ENGINEERING

FPGROWTH ALGORITHM IMPLEMENTATION

CSE 454 DATA MINING
ASSIGNMENT 3 REPORT

STUDENT
Şeyda Nur DEMİR
12 10 44 042

LECTURER
Burcu YILMAZ

TEACHING ASSISTANT
-

KOCAELİ, 2020

1.DESCRIPTION

1.1.Requirements :

- Read FPGrowth : A Frequent Pattern-Growth approach from the book.
- Implement the approach.
  You must use the algorithm mentioned in the book.
  You can use any programming language.
  Find a dataset to present the results.
- You can not use any code from anywhere from internet.
  You may use a library for the tree implementation.
  You have to implement the assignment by yourself.

1.2.Deadline :

- 03.01.2020 23:55
- You must upload your assignment to moodle.

1.3.Demo :

- Before 15.01.2021 (Please ask an appointment from me to attend the demo)
- You must present your code in the demo section. If you do not attend the demo section, you will get zero from the assignment, even if you upload your assignment.

2.FPGROWTH ALGORITHM IMPLEMENTATION

❖ What is FPGrowth Approach?

6.2.4 A Pattern-Growth Approach for Mining Frequent Itemsets

As we have seen, in many cases the Apriori candidate generate-and-test method significantly reduces the size of candidate sets, leading to good performance gain. However, it can suffer from two nontrivial costs:

- *It may still need to generate a huge number of candidate sets.* For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets.
- *It may need to repeatedly scan the whole database and check a large set of candidates by pattern matching.* It is costly to go over each transaction in the database to determine the support of the candidate itemsets.

"Can we design a method that mines the complete set of frequent itemsets without such a costly candidate generation process?" An interesting method in this attempt is called **frequent pattern growth,** or simply **FP-growth**, which adopts a *divide-and-conquer* strategy as follows. First, it compresses the database representing frequent items into a **frequent pattern tree,** or **FP-tree**, which retains the itemset association information. It then divides the compressed database into a set of *conditional databases* (a special kind of projected database), each associated with one frequent item or "pattern fragment," and mines each database separately. For each "pattern fragment," only its associated data sets need to be examined. Therefore, this approach may substantially reduce the size of the data sets to be searched, along with the "growth" of patterns being examined. You will see how it works in Example 6.5.

Example 6.5 **FP-growth (finding frequent itemsets without candidate generation).** We reexamine the mining of transaction database, D, of Table 6.1 in Example 6.3 using the frequent pattern growth approach.

❖ What is the algorithm of the FPGrowth mentioned our book?

Algorithm: FP_growth. Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input:

- D, a transaction database;
- *min_sup*, the minimum support count threshold.

Output: The complete set of frequent patterns.

Method:

1. The FP-tree is constructed in the following steps:
   (a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the *list* of frequent items.
   (b) Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in D do the following.
   Select and sort the frequent items in *Trans* according to the order of L. Let the sorted frequent item list in *Trans* be $[p|P]$, where p is the first element and P is the remaining list. Call insert_tree($[p|P]$, T), which is performed as follows. If T has a child N such that $N.item\text{-}name = p.item\text{-}name$, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same *item-name* via the node-link structure. If P is nonempty, call insert_tree(P, N) recursively.
2. The FP-tree is mined by calling FP_growth(*FP_tree*, *null*), which is implemented as follows.

```
procedure FP_growth(Tree, α)
    if Tree contains a single path P then
        for each combination (denoted as β) of the nodes in the path P
            generate pattern β ∪ α with support_count = minimum support count of nodes in β;
    else for each a_i in the header of Tree {
        generate pattern β = a_i ∪ α with support_count = a_i.support_count;
        construct β's conditional pattern base and then β's conditional FP_tree Tree_β;
        if Tree_β ≠ ∅ then
            call FP_growth(Tree_β, β); }
```

Figure 6.9 FP-growth algorithm for discovering frequent itemsets without candidate generation.

- ❖ **What is the dataset in our book at the Table 6.1?**

Table 6.1 Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

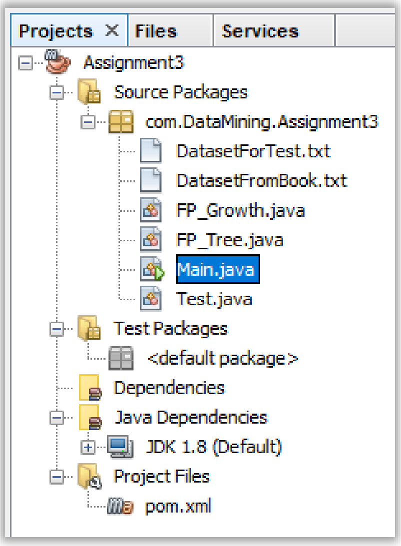
- ❖ **I used this dataset in my file :**

DatasetFromBook.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
I1,I2,I5
I2,I4
I2,I3
I1,I2,I4
I1,I3
I2,I3
I1,I3
I1,I2,I3,I5
I1,I2,I3
```

- ❖ **I used Netbeans, and I implemented code in Java programming language :**



- **My main class :**

```java
package com.DataMining.Assignment3;

/**
 *
 * @author seydanur
 * CSE 454 - Data Mining
 * Assignment 3
 * Seyda Nur DEMIR
 * 12 10 44 042
 * FP-Growth Algorithm Implementation
 * Main Class
 */
public class Main {
    public static void main(String [] args)
    {
        Test testObject = new Test();

        System.out.println("------------------------------------------------------------------------");
        System.out.println("TEST 1.1 : Dataset From Book (Table 6.1) with Minimum Support 2");
        testObject.test(2,"C:\\Users\\sefer\\Documents\\NetBeansProjects\\Assignment3\\src\\main\\java\\com\\DataMining\\Assignment3\\DatasetFromBook.txt");

        System.out.println("------------------------------------------------------------------------");
        System.out.println("TEST 1.2 : Dataset From Book (Table 6.1) with Minimum Support 3");
        testObject.test(3,"C:\\Users\\sefer\\Documents\\NetBeansProjects\\Assignment3\\src\\main\\java\\com\\DataMining\\Assignment3\\DatasetFromBook.txt");

        System.out.println("------------------------------------------------------------------------");
        System.out.println("TEST 2.1 : Dataset For Test with Minimum Support 2");
        testObject.test(2,"C:\\Users\\sefer\\Documents\\NetBeansProjects\\Assignment3\\src\\main\\java\\com\\DataMining\\Assignment3\\DatasetForTest.txt");

        System.out.println("------------------------------------------------------------------------");
        System.out.println("TEST 2.2 : Dataset For Test with Minimum Support 3");
        testObject.test(3,"C:\\Users\\sefer\\Documents\\NetBeansProjects\\Assignment3\\src\\main\\java\\com\\DataMining\\Assignment3\\DatasetForTest.txt");

        System.out.println("-------------------------------------------------------------------- END");
    }
}
```

- **Important detail for test :**

<u>You should give your own true path when you are testing algorithm.</u>

- **Test inputs :**

D : dataset (Give string filename)

min_sup : minimum support count number (Give integer number)

❖ **Test 1.1 Outputs :**

Input : Dataset From Book and Minimum Support Number 2

```
-------------------------------------------------------------------------
TEST 1.1 : Dataset From Book (Table 6.1) with Minimum Support 2
-------------------------------------------------------------------------
Dataset [D] Items [I]
-------------------------------------------------------------------------
I1 I2 I5
I2 I4
I2 I3
I1 I2 I4
I1 I3
I2 I3
I1 I3
I1 I2 I3 I5
I1 I2 I3
-------------------------------------------------------------------------
Items [F] and Frequencies [Count]
-------------------------------------------------------------------------
I1 : 6
I2 : 7
I3 : 6
I4 : 2
I5 : 2
-------------------------------------------------------------------------
Sorted List [descending order as L]
-------------------------------------------------------------------------
I2 : 7
I1 : 6
I3 : 6
I4 : 2
I5 : 2
-------------------------------------------------------------------------
Removed Items [Count >= Minimum Support Count]
-------------------------------------------------------------------------
I2 : 7
I1 : 6
I3 : 6
I4 : 2
I5 : 2
-------------------------------------------------------------------------
Mined Frequent Patterns
-------------------------------------------------------------------------
{ I2 I1 : 4 }
{ I3 I1 : 4 }
{ I3 I2 : 4 }
{ I1 I3 I2 : 2 }
{ I4 I2 : 2 }
{ I5 I1 : 2 }
{ I5 I2 : 2 }
{ I1 I5 I2 : 2 }
-------------------------------------------------------------------------
```

❖ **Test 1.2 Outputs :**

Input : Dataset From Book and Minimum Support Number 3

```
------------------------------------------------------------------------
TEST 1.2 : Dataset From Book (Table 6.1) with Minimum Support 3
------------------------------------------------------------------------
Dataset [D] Items [I]
------------------------------------------------------------------------
I1 I2 I5
I2 I4
I2 I3
I1 I2 I4
I1 I3
I2 I3
I1 I3
I1 I2 I3 I5
I1 I2 I3
------------------------------------------------------------------------
Items [F] and Frequencies [Count]
------------------------------------------------------------------------
I1 : 6
I2 : 7
I3 : 6
I4 : 2
I5 : 2
------------------------------------------------------------------------
Sorted List [descending order as L]
------------------------------------------------------------------------
I2 : 7
I1 : 6
I3 : 6
I4 : 2
I5 : 2
------------------------------------------------------------------------
Removed Items [Count >= Minimum Support Count]
------------------------------------------------------------------------
I2 : 7
I1 : 6
I3 : 6
------------------------------------------------------------------------
Mined Frequent Patterns
------------------------------------------------------------------------
{ I2 I1 : 4 }
{ I3 I1 : 4 }
{ I3 I2 : 4 }
------------------------------------------------------------------------
```

❖ **Test 2.1 Outputs :**

Input : Dataset For Test and Minimum Support Number 2

```
-------------------------------------------------------------------------
TEST 2.1 : Dataset For Test with Minimum Support 2
-------------------------------------------------------------------------
Dataset [D] Items [I]
-------------------------------------------------------------------------
I1 I2 I3
I2 I3 I4
I4 I5
I1 I2 I4
I1 I2 I3 I5
I1 I2 I3 I4
-------------------------------------------------------------------------
Items [F] and Frequencies [Count]
-------------------------------------------------------------------------
I1 : 4
I2 : 5
I3 : 4
I4 : 4
I5 : 2
-------------------------------------------------------------------------
Sorted List [descending order as L]
-------------------------------------------------------------------------
I2 : 5
I1 : 4
I3 : 4
I4 : 4
I5 : 2
-------------------------------------------------------------------------
Removed Items [Count >= Minimum Support Count]
-------------------------------------------------------------------------
I2 : 5
I1 : 4
I3 : 4
I4 : 4
I5 : 2
-------------------------------------------------------------------------
Mined Frequent Patterns
-------------------------------------------------------------------------
{ I2 I1 : 4 }
{ I3 I2 : 4 }
{ I3 I1 : 3 }
{ I1 I3 I2 : 3 }
{ I4 I2 : 3 }
{ I4 I1 : 2 }
{ I4 I3 : 2 }
{ I2 I4 I3 : 2 }
{ I1 I4 I2 : 2 }
-------------------------------------------------------------------------
```

❖ Test 2.2 Outputs :

Input : Dataset For Test and Minimum Support Number 3

```
-------------------------------------------------------------------------
TEST 2.2 : Dataset For Test with Minimum Support 3
-------------------------------------------------------------------------
Dataset [D] Items [I]
-------------------------------------------------------------------------
I1 I2 I3
I2 I3 I4
I4 I5
I1 I2 I4
I1 I2 I3 I5
I1 I2 I3 I4
-------------------------------------------------------------------------
Items [F] and Frequencies [Count]
-------------------------------------------------------------------------
I1 : 4
I2 : 5
I3 : 4
I4 : 4
I5 : 2
-------------------------------------------------------------------------
Sorted List [descending order as L]
-------------------------------------------------------------------------
I2 : 5
I1 : 4
I3 : 4
I4 : 4
I5 : 2
-------------------------------------------------------------------------
Removed Items [Count >= Minimum Support Count]
-------------------------------------------------------------------------
I2 : 5
I1 : 4
I3 : 4
I4 : 4
-------------------------------------------------------------------------
Mined Frequent Patterns
-------------------------------------------------------------------------
{ I2 I1 : 4 }
{ I3 I2 : 4 }
{ I3 I1 : 3 }
{ I1 I3 I2 : 3 }
{ I4 I2 : 3 }
- -------------------------------------------------------------------- END
```

END OF THE REPORT

UPDATED : 03.01.2021 23:30

STUDENT
Şeyda Nur DEMİR
12 10 44 042

LECTURER
Burcu YILMAZ

TEACHING ASSISTANT
-

KOCAELİ, 2020