Discovering Periodic-Frequent patterns in Big Data Using PFPMC Algorithm

In this tutorial, we will discuss two approaches to find Periodic-Frequent patterns in big data using top algorithm.

1. **Basic approach:** Here, we present the steps to discover Periodic-Frequent patterns using a single minimum support value
2. **Advanced approach:** Here, we generalize the basic approach by presenting the steps to discover Periodic-Frequent patterns using multiple minimum support values.

Basic approach: Executing PFPMC on a single dataset at a particular minimum support value

Step 1: Import the PFPMC algorithm

In [1]:
```
from PAMI.periodicFrequentPattern.basic import PFPMC  as alg
```

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'temporal_T10I4D100K.csv'

minimumSupportCount=100  #Users can also specify this constraint between 0 to 1.
maxmunPeriodCount=500
seperator='¥t'
```

Step 3: Execute the PFPMC algorithm

In [3]:
```
obj = alg.PFPMC(iFile=inputFile, minSup=minimumSupportCount,maxPer=maxmunPeriodCount
obj.startMine()            #Start the mining process
```

```
Periodic-Frequent patterns were generated successfully using PFPDiffset ECLAT algori
thm
```

Step 4: Storing the generated patterns

Step 4.1: Storing the generated patterns in a file

In [4]:
```
obj.savePatterns(outFile='periodicFrequentPatternsMinSupCount100.txt')
```

Step 4.2. Storing the generated patterns in a data frame

In [5]:
```
periodicFrequentPatternsDF= obj.getPatternsAsDataFrame()
```

Step 5: Getting the statistics

Step 5.1: Total number of discovered patterns

In [6]:
```
print('Total No of patterns: ' + str(len(periodicFrequentPatternsDF)))
```

```
Total No of patterns: 385
```

Step 5.2: Runtime consumed by the mining algorithm

In [7]:
```
print('Runtime: ' + str(obj.getRuntime()))
```

```
Runtime: 1611.1489055156708
```

In [8]:
```
##### Step 5.3: Total Memory consumed by the mining algorithm
```

In [9]:
```
print('Memory (RSS): ' + str(obj.getMemoryRSS()))
print('Memory (USS): ' + str(obj.getMemoryUSS()))
```

```
Memory (RSS): 1815703552
Memory (USS): 1776992256
```