Discovering Partial Periodic Pattern in Big Data Using GThreePGrowth Algorithm

In this tutorial, we will discuss two approaches to find Partial Periodic Pattern in big data using GThreePGrowth algorithm.

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

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

Step 1: Import the GThreePGrowth algorithm

In [1]:
```
from PAMI.partialPeriodicPattern.basic import GThreePGrowth  as alg
```

Step 2: Specify the following input parameters

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

periodCount=5000
relativePSCount=0.2
periodicSupportCount=1000  #Users can also specify this constraint between 0 to 1.

seperator='¥t'
```

Step 3: Execute the GThreePGrowth algorithm

In [3]:
```
obj = alg.GThreePGrowth(iFile=inputFile,periodicSupport=periodicSupportCount, perioc
obj.startMine()            #Start the mining process
```

```
375
1000 5000 0.2
Partial Periodic Patterns were generated successfully using Generalized 3PGrowth alg
orithm
```

Step 4: Storing the generated patterns

Step 4.1: Storing the generated patterns in a file

In [4]:
```
obj.savePatterns(outFile='frequentPatternsMinSupCount1000.txt')
```

Step 4.2. Storing the generated patterns in a data frame

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

Step 5: Getting the statistics

Step 5.1: Total number of discovered patterns

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

```
Total No of patterns: 384
```

Step 5.2: Runtime consumed by the mining algorithm

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

```
Runtime: 7.77413010597229
```

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): 474652672
Memory (USS): 436002816
```