Advanced Tutorial on Implementing GThreePGrowth Algorithm

In this tutorial, we explain how the GThreePGrowth algorithm can be implemented by varying the minimum support values

Step 1: Import the GThreePGrowth algorithm and pandas data frame

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

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'temporal_T10I4D100K.csv'
seperator='¥t'
periodCount=500
periodicSupportCountList = [100, 150, 200, 250, 300]
#minimumSupport can also specified between 0 to 1. E.g., minSupList = [0.005, 0.006,
relativePSCount=0.2
result = pd.DataFrame(columns=['algorithm', 'minSup', 'period','relativePS','patterr
#initialize a data frame to store the results of GThreePGrowth algorithm
```

Step 3: Execute the GThreePGrowth algorithm using a for loop

In [3]:
```
algorithm = 'GThreePGrowth'  #specify the algorithm name
for periodicSupportCount in periodicSupportCountList:
    obj = alg.GThreePGrowth('temporal_T10I4D100K.csv', periodicSupport=periodicSuppo
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, periodicSupportCount,periodCount,relat
    print(result)
```

```
760
100 500 0.2
Partial Periodic Patterns were generated successfully using Generalized 3PGrowth alg
orithm
       algorithm  minSup  period  relativePS  patterns    runtime     memory
0  GThreePGrowth     100     500         0.2      8454  12.124357  574042112
733
150 500 0.2
Partial Periodic Patterns were generated successfully using Generalized 3PGrowth alg
orithm
       algorithm  minSup  period  relativePS  patterns    runtime     memory
0  GThreePGrowth     100     500         0.2      8454  12.124357  574042112
1  GThreePGrowth     150     500         0.2      6369  10.808095  571535360
707
200 500 0.2
Partial Periodic Patterns were generated successfully using Generalized 3PGrowth alg
orithm
       algorithm  minSup  period  relativePS  patterns    runtime     memory
0  GThreePGrowth     100     500         0.2      8454  12.124357  574042112
1  GThreePGrowth     150     500         0.2      6369  10.808095  571535360
2  GThreePGrowth     200     500         0.2      4672  10.902376  567865344
682
250 500 0.2
Partial Periodic Patterns were generated successfully using Generalized 3PGrowth alg
orithm
       algorithm  minSup  period  relativePS  patterns    runtime     memory
0  GThreePGrowth     100     500         0.2      8454  12.124357  574042112
1  GThreePGrowth     150     500         0.2      6369  10.808095  571535360
2  GThreePGrowth     200     500         0.2      4672  10.902376  567865344
3  GThreePGrowth     250     500         0.2      2789   9.689085  564477952
654
300 500 0.2
Partial Periodic Patterns were generated successfully using Generalized 3PGrowth alg
orithm
       algorithm  minSup  period  relativePS  patterns    runtime     memory
0  GThreePGrowth     100     500         0.2      8454  12.124357  574042112
1  GThreePGrowth     150     500         0.2      6369  10.808095  571535360
2  GThreePGrowth     200     500         0.2      4672  10.902376  567865344
3  GThreePGrowth     250     500         0.2      2789   9.689085  564477952
4  GThreePGrowth     300     500         0.2      1996   9.762537  559456256
```

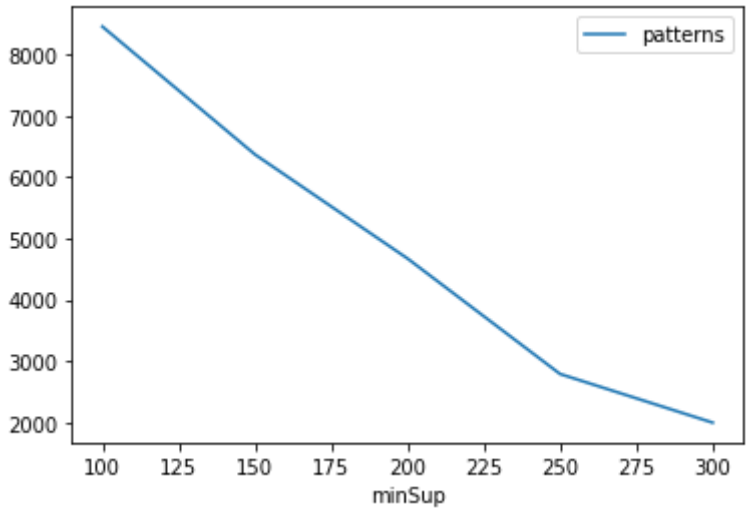
Step 5: Visualizing the results

Step 5.1 Importing the plot library

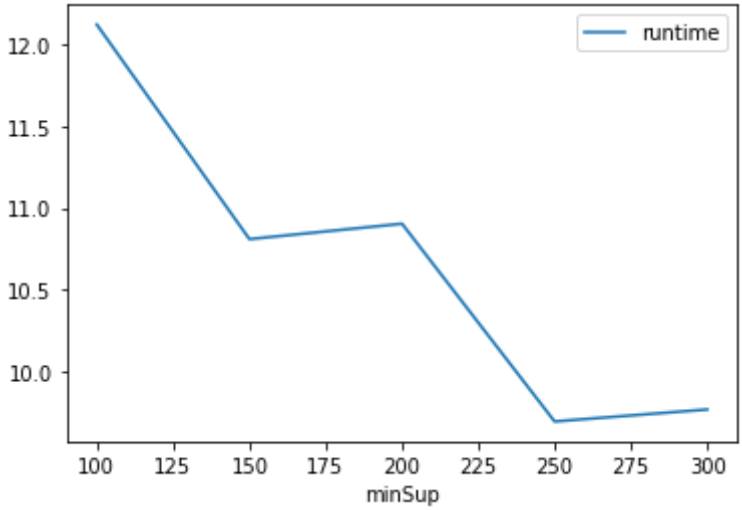
In [4]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

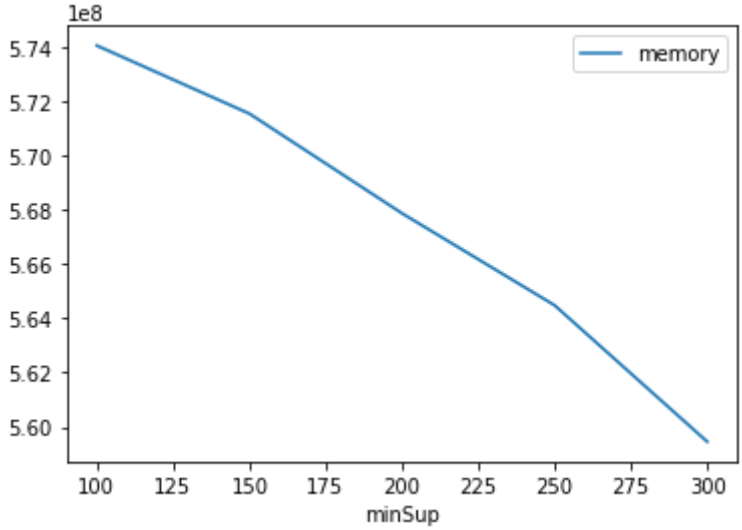
In [5]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```




```
Graph for No Of Patterns is successfully generated!
```




```
Graph for Runtime taken is successfully generated!
```




```
Graph for memory consumption is successfully generated!
```

Step 6: Saving the results as latex files

In [6]:
```
from PAMI.extras.graph import generateLatexFileFromDataFrame as gdf
gdf.generateLatexCode(result)
```

```
Latex files generated successfully
```