Advanced Tutorial on Implementing CPFPMiner Algorithm

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

Step 1: Import the CPFPMiner algorithm and pandas data frame

In [1]:
```
from PAMI.periodicFrequentPattern.closed import CPFPMiner  as alg
import pandas as pd
```

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'temporal_T10I4D100K.csv'
seperator='¥t'
maxmunPeriodCount=5000
minimumSupportCountList = [100, 150, 200, 250, 300]
#minimumSupport can also specified between 0 to 1. E.g., minSupList = [0.005, 0.006,

result = pd.DataFrame(columns=['algorithm', 'minSup', 'maxPer','patterns', 'runtime'
#initialize a data frame to store the results of CPFPMiner algorithm
```

Step 3: Execute the CPFPMiner algorithm using a for loop

In [3]:
```
algorithm = 'CPFPMiner'  #specify the algorithm name
for minSupCount in minimumSupportCountList:
    obj = alg.CPFPMiner(iFile=inputFile, minSup=minSupCount,maxPer=maxmunPeriodCoun
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, minSupCount,maxmunPeriodCount, len(obj
```

```
Closed periodic frequent patterns were generated successfully using CPFPMiner algori
thm
Closed periodic frequent patterns were generated successfully using CPFPMiner algori
thm
Closed periodic frequent patterns were generated successfully using CPFPMiner algori
thm
Closed periodic frequent patterns were generated successfully using CPFPMiner algori
thm
Closed periodic frequent patterns were generated successfully using CPFPMiner algori
thm
```

In [4]:
```
print(result)
```

```
   algorithm  minSup  maxPer  patterns    runtime     memory
0  CPFPMiner     100    5000     24711  21.105902  145326080
1  CPFPMiner     150    5000     18448  19.881866  148971520
2  CPFPMiner     200    5000     13095  18.855100  150237184
3  CPFPMiner     250    5000      7651  17.600360  151318528
4  CPFPMiner     300    5000      4509  16.433301  153374720
```

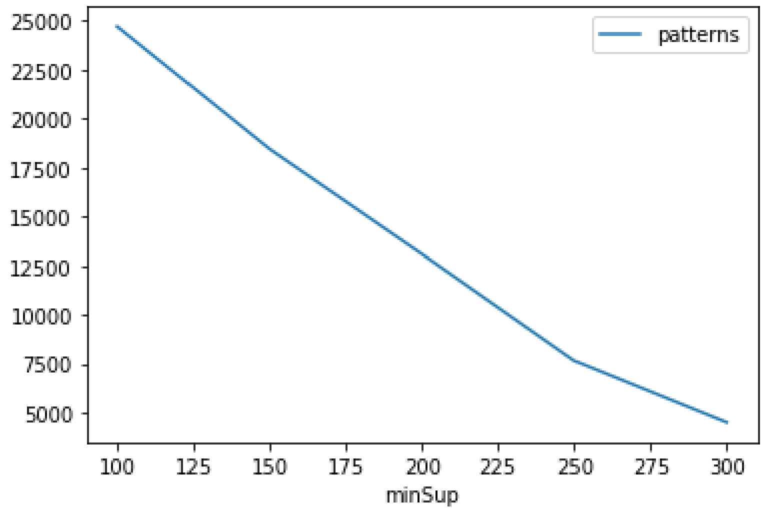
Step 5: Visualizing the results

Step 5.1 Importing the plot library

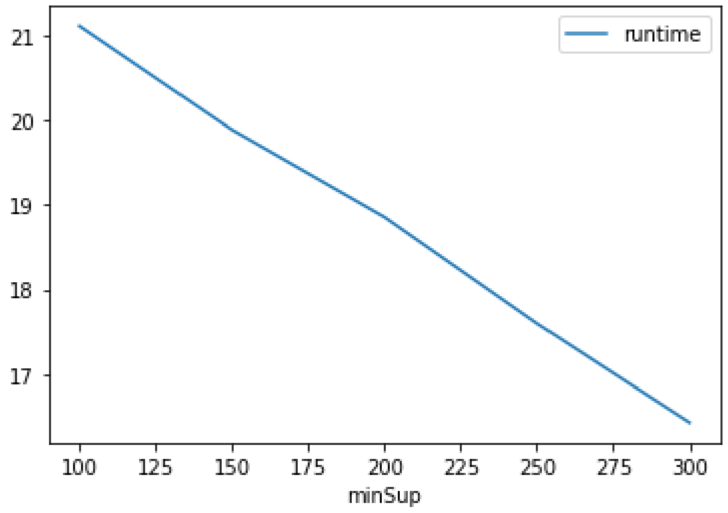
In [5]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

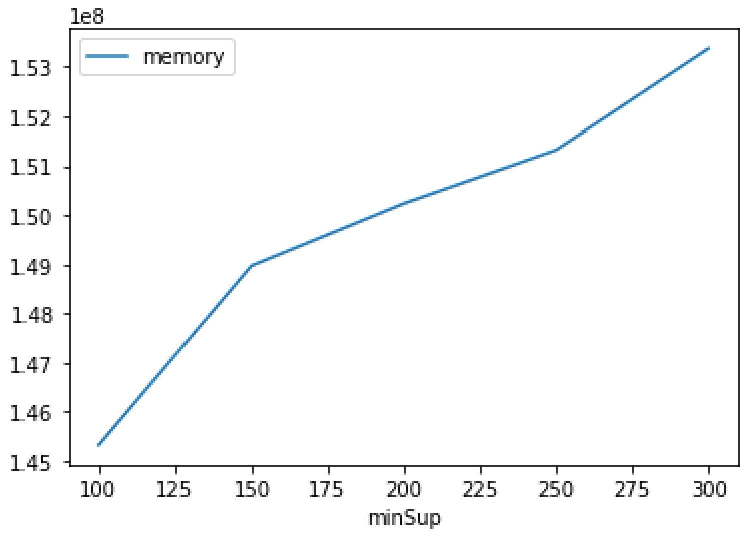
In [6]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



```
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 [7]:
```
from PAMI.extras.graph import generateLatexFileFromDataFrame as gdf
gdf.generateLatexCode(result)
```

```
Latex files generated successfully
```