Advanced Tutorial on Implementing CSPGrowth Algorithm

In this tutorial, we explain how the Correlated Pattern GrowthPlus (CSPGrowth) algorithm can be implemented by varying the minimum support values

Step 1: Import the CSPGrowth algorithm and pandas data frame

In [1]:
```
from PAMI.correlatedSpatialPattern.basic import CSPGrowth  as alg
import pandas as pd
```

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'transactional_T10I4D100K.csv'
seperator='¥t'
minAllConfCount=0.1
minimumSupportCountList = [100, 150, 200, 250, 300]
#minimumSupport can also specified between 0 to 1. E.g., minSupList = [0.005, 0.006,
neghberFile='T10_utility_neighbour.txt'
result = pd.DataFrame(columns=['algorithm', 'minSup',"minAllConf" , 'patterns', 'rur
#initialize a data frame to store the results of CSPGrowth algorithm
```

Step 3: Execute the CSPGrowth algorithm using a for loop

In [3]:
```
algorithm = 'CSPGrowth'  #specify the algorithm name
for minSupCount in minimumSupportCountList:
    obj = alg.CSPGrowth('transactional_T10I4D100K.csv', minSup=minSupCount,nFile=neg
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, minSupCount,minAllConfCount, len(obj.g
```

```
Correlated Spatial Frequent Patterns were generated successfully using CSPGrowth alg
orithm
Correlated Spatial Frequent Patterns were generated successfully using CSPGrowth alg
orithm
Correlated Spatial Frequent Patterns were generated successfully using CSPGrowth alg
orithm
Correlated Spatial Frequent Patterns were generated successfully using CSPGrowth alg
orithm
Correlated Spatial Frequent Patterns were generated successfully using CSPGrowth alg
orithm
```

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

```
   algorithm  minSup  minAllConf  patterns    runtime     memory
0  CSPGrowth     100         0.1      1200  63.358973  406347776
1  CSPGrowth     150         0.1      1620  59.510943  452620288
2  CSPGrowth     200         0.1      1870  58.002539  453492736
3  CSPGrowth     250         0.1      2067  56.828973  453505024
4  CSPGrowth     300         0.1      2285  55.200268  454152192
```

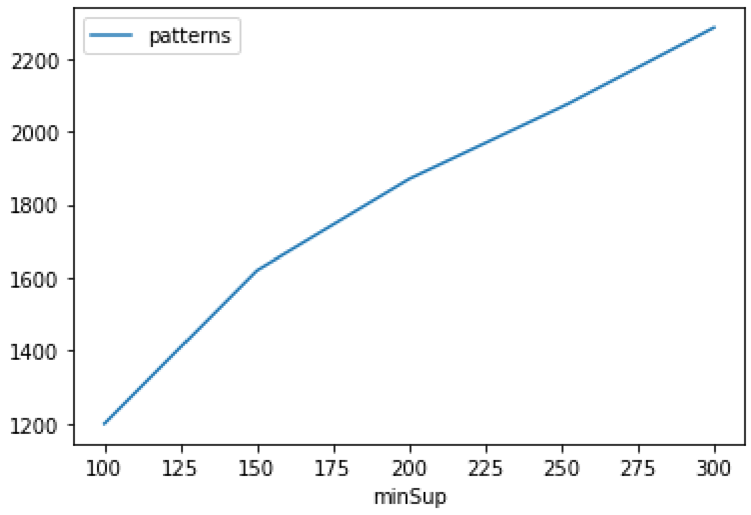
Step 5: Visualizing the results

Step 5.1 Importing the plot library

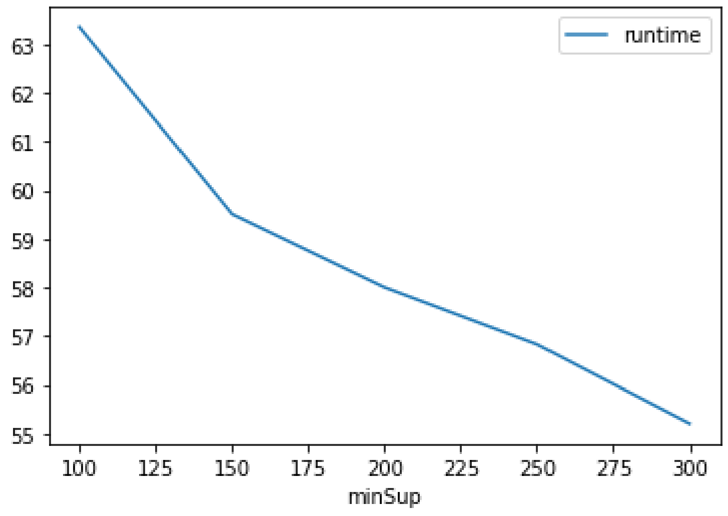
In [5]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

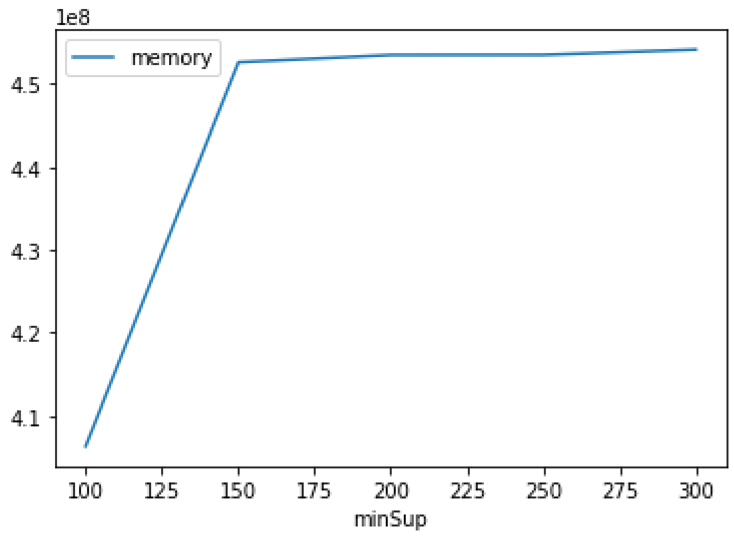
In [6]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



```
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
```

In [ ]: