

Advanced Tutorial on Implementing PFPGrowth Algorithm

In this tutorial, we explain how the Periodic Frequent Puttern Growth (PFPGrowth) algorithm can be implemented by varying the minimum support values

Step 1: Import the PFPGrowth algorithm and pandas data frame

```
In [1]: from PAMI.periodicFrequentPattern.basic import PFPGrowth as alg
import pandas as pd
```

Step 2: Specify the following input parameters

```
In [2]: inputFile = 'temporal_T10I4D100K.csv'
separator='¥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 PFPGrowth algorithm
```

Step 3: Execute the PFPGrowth algorithm using a for loop

```
In [3]: algorithm = 'PFPGrowth' #specify the algorithm name
for minSupCount in minimumSupportCountList:
    obj = alg.PFPGrowth('temporal_T10I4D100K.csv', minSup=minSupCount, maxPer=maxmunPeriodCount)
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, minSupCount, maxmunPeriodCount, len(obj.patterns), obj.runtime, obj.memory]
```

Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
 Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
 Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
 Periodic Frequent patterns were generated successfully using PFPGrowth algorithm
 Periodic Frequent patterns were generated successfully using PFPGrowth algorithm

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

	algorithm	minSup	maxPer	patterns	runtime	memory
0	PFPGrowth	100	5000	25462	13.866913	579379200
1	PFPGrowth	150	5000	18982	12.462541	576831488
2	PFPGrowth	200	5000	13251	12.453089	573923328
3	PFPGrowth	250	5000	7702	10.481392	570593280
4	PFPGrowth	300	5000	4552	10.867716	566153216

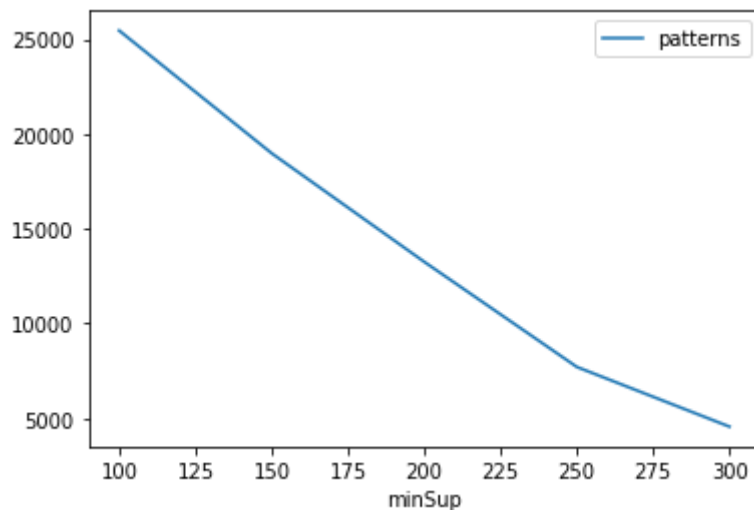
Step 5: Visualizing the results

Step 5.1 Importing the plot library

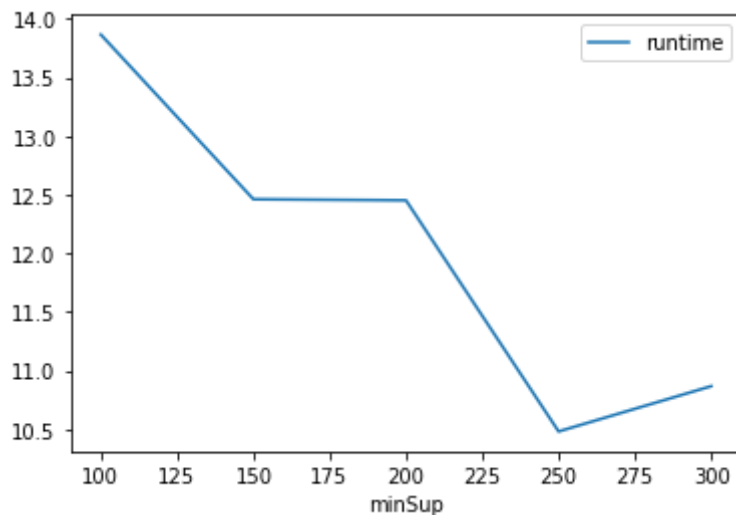
```
In [5]: from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

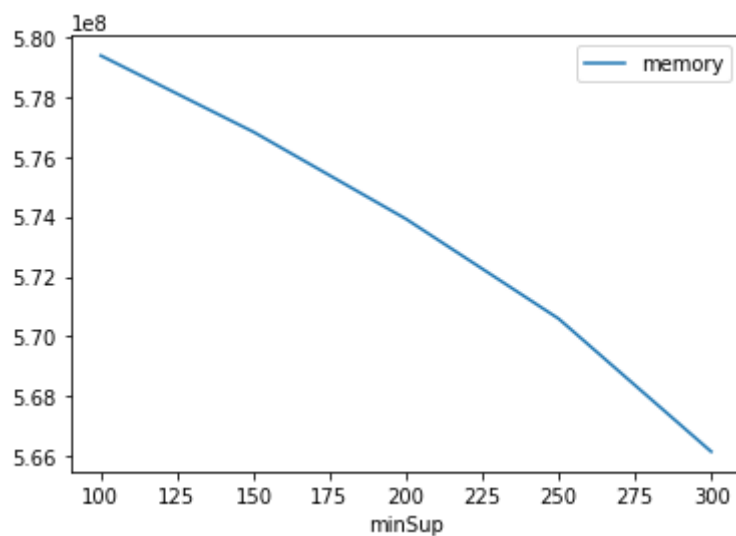
```
In [6]: ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



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