Discovering Correlated Spatial Pattern in Big Data Using CSPGrowth Algorithm

In this tutorial, we will discuss two approaches to find correlated spatial pattern in big data using CSPGrowth algorithm.

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

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

Step 1: Import the CSPGrowth algorithm

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

Step 2: Specify the following input parameters

In [2]:
```
inputFile = 'transactional_T10I4D100K.csv'
minAllConfCount=0.1
minimumSupportCount=100  #Users can also specify this constraint between 0 to 1.
neghberFile='T10_utility_neighbour.txt'
seperator='¥t'
```

Step 3: Execute the CSPGrowth algorithm

In [3]:
```
obj = alg.CSPGrowth(iFile=inputFile, minSup=minimumSupportCount, nFile=neghberFile,
obj.startMine()            #Start the mining process
```

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

Step 4: Storing the generated patterns

Step 4.1: Storing the generated patterns in a file

In [4]:
```
obj.savePatterns(outFile='frequentPatternsMinSupCount100.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: 1200
```

Step 5.2: Runtime consumed by the mining algorithm

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

```
Runtime: 64.01227521896362
```

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): 405430272
Memory (USS): 366821376
```