# Mining Recurring Patterns in Temporal Databases

## What is recurrent pattern mining?

Recurrent pattern mining aims to discover all interesting patterns in a temporal database that have **periodic support** no less than the user-specified **minimum periodic support** (**minPS**) constraint, **period** no greater than the user-specified **maximum Interval Time** (**maxIAT**) constraint and **Recurrence** no less than the user-specified **minimum recurrence** (***minRec***). The **minSup** controls the minimum number of transactions that a pattern must appear in a database. The **maxIAT** controls the maximum interval time the pattern must reappear. The **minRec** controls the number of periodic intervals of a pattern.

Reference: S. Lorpunmanee and S. Kamonsantiroj, "Efficient Mining Recurring Patterns of Inter-Transaction in Time Series," J. Adv. Comput. Intell. Intell. Inform., Vol.23, No.3, pp. 402-413, 2019, DOI: 10.20965/jaciii.2019.p0402

## What is a temporal database?

A temporal database is an unordered collection of transactions. A temporal represents a pair constituting of temporal-timestamp and a set of items.
A hypothetical temporal database containing the items ***a, b, c, d, e, f, and g*** and its timestamp is shown below

| TS | Transactions |
|---|---|
| 1 | a c d f g |
| 2 | a b c d |
| 3 | a c d f g |
| 4 | a b c d e f g |
| 5 | b c e f g |
| 7 | c d e f |
| 8 | a b c d e f g |
| 9 | c d e f g |
| 10 | a b c d e f g |
| 12 | a c d e f g |
| 13 | a c d f g |
| 14 | a c e g |
| 15 | a d f g |
| 16 | a b c |

**Note:** Duplicate items must not exist within a transaction.

# What is the acceptable format of a temporal database in PAMI?

Each row in a temporal database must contain timestamp and items. A sample transactional database, say sampleInputFile.txt, is provided below.

In [2]:
```
!cat recurringSample.txt
```

```
1 a c d f g
2 a b c d
3 a c d f g
4 a b c d e f g
5 b c e f g
7 c d e f
8 a b c d e f g
9 c d e f g
10 a b c d e f g
12 a c d e f g
13 a c d f g
14 a c e g
15 a d f g
16 a b c
```

# Understanding the statistics of a temporal database

The performance of a pattern mining algorithm primarily depends on the satistical nature of a database. Thus it is important to know the following details of a database:

- Total number of transactions (Database size)
- Total number of unique items in database
- Minimum lenth of transaction that exists in database
- Average length of all transactions that exists in database
- Maximum length of transaction that exists in database
- Minimum periodicity that exists in database
- Average periodicity hat exists in database
- Maximum periodicity that exists in database
- Standard deviation of transaction length
- Variance in transaction length
- Sparsity of database

The below sample code prints the statistical details of a database.

In [ ]:
```
import PAMI.extras.dbStats.temporalDatabaseStats as stats
obj = stats.temporalDatabaseStats('sampleInputFile.txt', ' ')
obj.run()
obj.printStats()
```

# What are the input parameters?

The input parameters to a periodic frequent pattern mining algorithm are:

- **Temporal database**
  Acceptable formats:

  > - String : E.g., 'temporalDatabase.txt'
  > - URL : E.g., https://u-aizu.ac.jp/~udayrage/datasets/transactionalDatabases/transactional_T10
  > - DataFrame with the header titled 'TS' and 'Transactions'

- **minSup**
  specified in

  > - **count (beween 0 to length of a database)** or
  > - [0, 1]

- **maxPer**
  specified in

  > - **count (beween 0 to length of a database)** or
  > - [0, 1]

- **minRec**
  specified in

  > - [0, 1]

- **seperator**
  default seperator is '\t' (tab space)

## How to store the output of a recurring pattern mining algorithm?

The patterns discovered by a recurring pattern mining algorithm can be saved into a file or a data frame.

## How to run the recurrent pattern mining algorithms in a terminal?

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into recurring pattern folder.
- Enter into recurringPattern folder
- Enter into specific folder execute the following command on terminal.

**syntax:** python3 algorithmName.py `<path to the input file>` `<path to the output file>` `<minSup>` `<maxPer>` `<minRec>` `<seperator>`

**Example:** python3 `RPGrowth` `inputFile.txt` `outputFile.txt` `4` `3` `2` `' '`

# How to execute a recurring pattern mining algorithm in a Jupyter Notebook?

- Install the PAMI package from the PYPI repository by executing the following command: **pip3 install PAMI**
- Run the below sample code by making necessary changes

In [ ]:
```
import PAMI.recurringPattern.RPGrowth as alg

iFile = 'sampleInputFile.txt'  #specify the input temporal database <br>
minSup = 4  #specify the minSup value <br>
maxPer = 4   #specify the maxPer value <br>
minRec = 1.5    #specify the maxRec Value <br>
seperator = ' ' #specify the seperator. Default seperator is tab space. <
oFile = 'recurringPatterns.txt'   #specify the output file name<br>

obj = alg.RPGrowth(iFile, minSup, maxPer, minRec, seperator) #initialize
obj.startMine()                         #start the mining process <br>
obj.savePatterns(oFile)                 #store the patterns in file <br>
df = obj.getPatternsAsDataFrame()       #Get the patterns discovered into a
obj.printStats()                        #Print the statistics of mining pro
```

The recurringPatterns.txt file contains the following patterns (*format:* pattern:support:lability):!cat periodicCorrelatedPatterns.txt

In [4]:
```
!cat recurringPatterns.txt
```

```
a :11:2:{{[1, 4] : 4}{[8, 16] : 7}}
a f :8:2:{{[1, 4] : 3}{[8, 15] : 5}}
a f c :7:2:{{[1, 4] : 3}{[8, 13] : 4}}
a f c d :7:2:{{[1, 4] : 3}{[8, 13] : 4}}
a f c d g :7:2:{{[1, 4] : 3}{[8, 13] : 4}}
a f c g :7:2:{{[1, 4] : 3}{[8, 13] : 4}}
a f d :8:2:{{[1, 4] : 3}{[8, 15] : 5}}
a f d g :8:2:{{[1, 4] : 3}{[8, 15] : 5}}
a f g :8:2:{{[1, 4] : 3}{[8, 15] : 5}}
a d :9:2:{{[1, 4] : 4}{[8, 15] : 5}}
a d g :8:2:{{[1, 4] : 3}{[8, 15] : 5}}
a d g c :7:2:{{[1, 4] : 3}{[8, 13] : 4}}
a d c :8:2:{{[1, 4] : 4}{[8, 13] : 4}}
a g :9:2:{{[1, 4] : 3}{[8, 15] : 6}}
a g c :8:2:{{[1, 4] : 3}{[8, 14] : 5}}
a c :10:2:{{[1, 4] : 4}{[8, 16] : 6}}
d g :9:2:{{[1, 4] : 3}{[8, 15] : 6}}
d g c :8:2:{{[1, 4] : 3}{[8, 13] : 5}}
d g c f :8:2:{{[1, 4] : 3}{[8, 13] : 5}}
d g f :9:2:{{[1, 4] : 3}{[8, 15] : 6}}
```

The dataframe containing the patterns is shown below:

In [3]:
```
df
```

Out[3]:

| | Patterns | Support | Recurrance | intervals |
|---|---|---|---|---|
| **0** | a | 11 | 2 | {[1, 4] : 4}{[8, 16] : 7} |
| **1** | a f | 8 | 2 | {[1, 4] : 3}{[8, 15] : 5} |
| **2** | a f c | 7 | 2 | {[1, 4] : 3}{[8, 13] : 4} |
| **3** | a f c d | 7 | 2 | {[1, 4] : 3}{[8, 13] : 4} |
| **4** | a f c d g | 7 | 2 | {[1, 4] : 3}{[8, 13] : 4} |
| **5** | a f c g | 7 | 2 | {[1, 4] : 3}{[8, 13] : 4} |
| **6** | a f d | 8 | 2 | {[1, 4] : 3}{[8, 15] : 5} |
| **7** | a f d g | 8 | 2 | {[1, 4] : 3}{[8, 15] : 5} |
| **8** | a f g | 8 | 2 | {[1, 4] : 3}{[8, 15] : 5} |
| **9** | a d | 9 | 2 | {[1, 4] : 4}{[8, 15] : 5} |
| **10** | a d g | 8 | 2 | {[1, 4] : 3}{[8, 15] : 5} |
| **11** | a d g c | 7 | 2 | {[1, 4] : 3}{[8, 13] : 4} |
| **12** | a d c | 8 | 2 | {[1, 4] : 4}{[8, 13] : 4} |
| **13** | a g | 9 | 2 | {[1, 4] : 3}{[8, 15] : 6} |
| **14** | a g c | 8 | 2 | {[1, 4] : 3}{[8, 14] : 5} |
| **15** | a c | 10 | 2 | {[1, 4] : 4}{[8, 16] : 6} |
| **16** | d g | 9 | 2 | {[1, 4] : 3}{[8, 15] : 6} |
| **17** | d g c | 8 | 2 | {[1, 4] : 3}{[8, 13] : 5} |
| **18** | d g c f | 8 | 2 | {[1, 4] : 3}{[8, 13] : 5} |
| **19** | d g f | 9 | 2 | {[1, 4] : 3}{[8, 15] : 6} |