# Data Mining Lab 2

Gibson Chikafa
chikafa@kth.se

Akhil Yerrapragada
akhily@kth.se

## 1 Assignment Goals

In this homework, we implement a program for finding frequent itemsets in Python. In the first part we implement the A-Priori algorithm with support at least **s** in a dataset of sales of transactions given in Canvas. In the second part, we develop and implement an algorithm for generating association rules between frequent itemsets discovered by using the A-Priori algorithm in a dataset of sales transactions.

## 2 Solution overview

Our solution comprises of the following steps:

1. We read the data from the file given in canvas: *T10I4D100K.dat*. Each line in the file represents a row of a set of items. This is done by the `read_buckets` function which return buckets with data structure `List[List[int]]`.

2. From the list of buckets read in step 1, for each item we get the total of its occurrences using the function `count_singletons` function. The data structure returned by this function is `Dic['string':int]`. string id the item or singleton and int is the number of occurences.

3. We find frequent singletons from the dictionary returned in step 2 using the the `support` value of 10000. Then we wrap these singletons in tuple to use the same data structure for pairs, triplets, etc later.

4. We implement the pipeline of the A-Priori algorithm. We start with finding frequent pairs. The frequent pairs become an input to find frequent triplets and frequent triplets become input to finding frequent quartets and so on until we can no longer find frequent itemssets of n-tuple. We use the same **support** of 10000 throughout.

5. Finally, we implement the association rules. Basically for each frequent n-tuple itemset we generate permutations, and starting from position i to len(k-1) in the permutation k, we calculate the confidence of elements from (0...i) and (i..len(k-1)) in the permutation. If the confidence calculated is above or equal to the *0.5* we consider this an association and add

it to our list of associations. If not we no longer proceed with this permutation, since if A,B,C→D is below confidence, so is A,B→C,D because of supp(A,B,C) ≤ supp(A,B).

# 3 How to run

You can change the following parameters:

- The "support threshold" value in variable: **support**.
- The "confidence threshold" value in variable: **confidence**, i.e the level of confidence you want in order to validate a rule.
- The "path" where you have to put the path of the dataset you want to use, which is in this case, the "T10I4D100K.dat" dataset.

# 4 Results

We obtained the following associations:

**Frequent 2-tuples**: (368, 682): 1193, (368, 829): 1194, (39, 825): 1187, (704, 825): 1102, (39, 704): 1107, (227, 390): 1049, (390, 722): 1042, (217, 346): 1336, (789, 829): 1194

**Frequent 3-tuples**: (39, 704, 825): 1035

**Frequent 4-tuples**:

**Associations**: ('704 → 39', 0.617056856187291), ('39, 825 → 704', 0.8719460825610783), ('39, 704 → 825', 0.9349593495934959), ('704 → 825', 0.6142697881828316), ('704, 825 → 39', 0.9392014519056261), ('227 → 390', 0.577007700770077), ('704 → 39, 825', 0.5769230769230769)

**Total Time**— 4.656568288803101 seconds —