# Dataset Generation


| NAME | ROLL NO. | PHONE NO. | EMAIL |
|---|---|---|---|
| Sangeeth S V | 181CO246 | 7907932994 | sangeeth.181co246@nitk.edu.in |
| Rajath C Aralikatti | 181CO241 | 7829158425 | rajath.181co241@nitk.edu.in |


## Algorithm:

For the purposes of creating and testing both algorithms used in this project, we have used the example dataset from the paper by Ya-Han Hu et. al., with the same item sequences, minimum support and time intervals list. For consequent testing and comparisons, we have used randomly generated databases made using the following algorithm.

- generate_items(n): returns a list of 'n' items randomly sampled from a predefined list of items.
- generate_time_list(n): returns a list of 'n' randomly generated timestamps. The times would start from a random timestamp and then be incremented iteratively using a number from a small addend list, say, [1,2,3,4]. So, if the starting timestamp is 1, the next timestamp would at most be 5. This limitation is imposed on the database generation algorithm because we have done all the testing with the differences in time intervals being 3.
- generate_database(itemList, timeList, minLength, maxLength, dbLength): This method takes as input, a *list of items* (generated using generate_items(n)), a *list of timestamps* (generated using generate_time_list(n)), *minLength* and *maxLength* which indicates the smallest and greatest number of items possible in each row of the database, and finally, *dbLength* which indicates the required number of rows in the database.

The generate_database method mentioned above, has been used to generate all the datasets used for the testing and analysis of our algorithms.

## Code:

```
Import random
```

```
def generate_items(n):
      items=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
      return random.sample(items, min(n,len(items)))
```

```
def generate_time_list(n):
  timeList = []
  times = [1,2,3,4,5]
  timeList.append(random.randint(1,n))
```

```
  for _ in range(n):
    timeList.append(timeList[-1] + random.choice(times))
  return timeList
```

```
def generate_database(items, timeList, minLength, maxLength, dbLength):
  db = []
  for i in range(dbLength):
    reqLength = random.randint(minLength, maxLength)
    itemSequence = []
    for j in range(reqLength):
      itemSequence.append((random.choice(items), random.choice(timeList)))
    itemSets = set()
    for i in itemSequence:
      itemSets.add(i)
    itemSequence = list(itemSets)
    itemSequence.sort(key=lambda x: x[1])
    db.append(itemSequence)
  return db

db1 = generate_database(items=['f','g','a','c'],
timeList=[1,6,2,12,25,16,19,21,17,15,8,3,5,9,10], minLength=50,
maxLength=60, dbLength=50)
for row in db1:
  print(row)
```

**Output:**

```
[('a', 1), ('g', 2), ('a', 2), ('e', 3), ('b', 6), ('e', 17), ('b', 21)]
[('b', 5), ('d', 8), ('b', 8), ('f', 9), ('b', 19), ('e', 21)]
[('g', 3), ('h', 5), ('e', 5), ('a', 17), ('h', 19), ('f', 25), ('d', 25)]
[('h', 1), ('d', 8), ('d', 8), ('c', 16), ('g', 17), ('b', 19)]
[('b', 2), ('b', 3), ('f', 8), ('e', 8), ('b', 10), ('g', 10), ('b', 19), ('h', 21)]
[('g', 1), ('d', 2), ('d', 3), ('c', 6), ('d', 8), ('e', 10), ('d', 12), ('e', 15), ('f', 15)]
[('h', 5), ('c', 10), ('c', 12), ('a', 15), ('c', 16), ('d', 16), ('b', 17), ('d', 19), ('g', 19)]
[('g', 1), ('c', 2), ('c', 3), ('f', 5), ('d', 8), ('c', 8), ('h', 9), ('a', 9), ('a', 17)]
[('a', 5), ('a', 8), ('d', 10), ('a', 19), ('b', 19), ('b', 21)]
[('a', 2), ('b', 3), ('b', 5), ('h', 6), ('h', 12), ('a', 16), ('e', 16)]
```