Lab5: Frede Emnetu (100704524)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import time
import pandas as pd
from itertools import combinations
```

useful functions

In [2]:
```
def createCk(Lk, k):
    cand_list = []
    len_Lk = len(Lk)

    for i in range(len_Lk):
        for j in range(i + 1, len_Lk):
            L1 = list(Lk[i])[:k - 2]
            L2 = list(Lk[j])[:k - 2]
            L1.sort()
            L2.sort()
            if L1 == L2:
                cand_list.append(Lk[i] | Lk[j])
    return cand_list
```

In [3]:
```
# hash functions
def hash1(i, j, numBuckets):
    return (i*j) % numBuckets


def hash2(i, j,numBuckets):
    return (i+j) % numBuckets
```

In [4]:
```
def output(pairs):
    for x in pairs:
        list_ = list(x)
        print('%d --> %d' % (list_[0],list_[1]))
```

In [5]:

```
def apyori(itemsets, min_support, numBuckets):
    start = time.time()
    C1 = []
    for basket in itemsets:
        for item in basket:
            if not [item] in C1:
                C1.append([item])
    C1 = [set(x) for x in C1]

    count = {}
    freq_items = []
    L1 = []
    countBuckets1 = [0] * numBuckets
    countBuckets2 = [0] * numBuckets

    for basket in itemsets:
        for item in C1:
            if item.issubset(basket):
                candidate = frozenset(item)
                if candidate not in count:
                    count[candidate] = 1
                else:
                    count[candidate] += 1
        pairs = list(combinations(basket, 2))
        for pair in pairs:
            listPair = list(pair)
            numbers = [int(x) for x in listPair]
            countBuckets1[hash1(numbers[0], numbers[1], numBuckets)] += 1

    for i in range(0, len(countBuckets1) - 1):
        if countBuckets1[i] / len(itemsets) >= min_support:
            countBuckets1[i] = 1
        else:
            countBuckets1[i] = 0

    for key in count:
        support = count[key] / len(itemsets)
        if support >= min_support:
            freq_items.insert(0, key)
            freq_items.insert(1, support)
            L1.insert(0, key)

#     print("Frequent Items: ", freq_items)

    freq_pairs = []
    L2 = []
    C2 = []
    final = []
    count = {}
    test = [list(x) for x in L1]

    for i in test:
        final.append(int(i[0]))
    # construct second hashmap
    for basket in itemsets:
        pairs = list(combinations(basket, 2))
```

```
        for pair in pairs:
            listPair = list(pair)
            numbers = [int(x) for x in listPair]
            # if the pair hashes to frequent bucket in first hashmap and bot
            # hash the pair to the second hashmap
            if countBuckets1[hash1(numbers[0], numbers[1],numBuckets)] == 1
                countBuckets2[hash2(numbers[0], numbers[1],numBuckets)] += 1

    for i in range(0, len(countBuckets2) - 1):
        if countBuckets2[i] / len(itemsets) >= min_support:
            countBuckets2[i] = 1
        else:
            countBuckets2[i] = 0

    for basket in itemsets:
        pairs = list(combinations(basket, 2))
        for pair in pairs:
            listPair = list(pair)
            numbers = [int(x) for x in listPair]
            # if pair hashes to frequent buckets in both hashmaps and values
            if countBuckets1[hash1(numbers[0], numbers[1],numBuckets)] == 1
                C2.append(pair)

    for basket in itemsets:
        for item in C2:
            item = frozenset(item)
            if item.issubset(basket):
                candidate = frozenset(item)
                if candidate not in count:
                    count[candidate] = 1
                else:
                    count[candidate] += 1

    for key in count:
        support = count[key] / len(itemsets)
        if support >= min_support:
            freq_pairs.insert(0, key)
            freq_pairs.insert(1, support)
            L2.insert(0, key)
    end = time.time()
    return [end - start, L2]

#     print("Frequent Pairs: ", freq_pairs)
```

Retail DataSet

In [6]:
```
retail = pd.read_csv("http://fimi.uantwerpen.be/data/retail.dat",delimiter="
itemsets = retail.values.tolist()
```

Remove Nan Values

In [7]:
```
retail_NN = []
for x in itemsets:
    retail_NN.append([i for i in x if str(i) != 'nan'])
```

Create Sections

In [8]:
```
sections = [2,5,10,15, 20]
associationR = []
for i, x in enumerate(sections):
    new_list = retail_NN[0:int(len(retail_NN)*(sections[i]/100))]
    associationR.append(new_list)
```

Use implemented function

In [9]:
```
times = []
pairslen = []
actualpairs = []
for x in range(5):
    print(x)
    results = apyori(associationR[x],0.01,2000)
    times.append(results[0])
    pairslen.append(len(results[1]))
    actualpairs.append(results[1])
```

```
0
1
2
3
4
```

Pairs

In [10]:
```
for x in range(len(sections)):
    print()
    print()
    print('with %g percent of the dataset I got %d pairs.' % (sections[x], p
    print()
    print(output(actualpairs[x]))
```

```
with 2 percent of the dataset I got 84 pairs.

41 --> 1327
38 --> 1327
32 --> 65
32 --> 271
749 --> 39
340 --> 39
381 --> 38
48 --> 301
41 --> 101
48 --> 1327
1327 --> 39
41 --> 271
41 --> 338
32 --> 110
589 --> 39
48 --> 740
740 --> 39
664 --> 39
38 --> 271
48 --> 110
604 --> 39
41 --> 255
48 --> 264
48 --> 310
41 --> 225
41 --> 170
48 --> 475
475 --> 39
48 --> 438
438 --> 39
89 --> 41
65 --> 41
48 --> 371
48 --> 60
371 --> 39
60 --> 39
371 --> 38
310 --> 39
38 --> 286
105 --> 38
48 --> 237
271 --> 39
270 --> 39
264 --> 39
225 --> 39
```

```
48 --> 255
255 --> 39
48 --> 65
237 --> 39
65 --> 39
179 --> 39
32 --> 179
32 --> 38
32 --> 48
32 --> 39
48 --> 170
170 --> 39
170 --> 38
48 --> 147
147 --> 39
41 --> 110
110 --> 39
110 --> 38
89 --> 38
48 --> 101
48 --> 95
48 --> 89
101 --> 39
89 --> 39
48 --> 79
48 --> 41
48 --> 36
79 --> 39
32 --> 41
48 --> 49
48 --> 39
48 --> 38
41 --> 39
41 --> 38
38 --> 39
37 --> 38
41 --> 36
36 --> 39
36 --> 38
None
```

with 5 percent of the dataset I got 126 pairs.

```
225 --> 60
89 --> 60
522 --> 1715
2990 --> 3735
1715 --> 604
604 --> 3966
60 --> 189
48 --> 3735
48 --> 2990
3735 --> 39
48 --> 1121
1715 --> 39
2238 --> 39
```

```
32 --> 286
48 --> 2238
2238 --> 38
32 --> 101
352 --> 1859
9 --> 310
41 --> 1327
32 --> 65
32 --> 271
48 --> 1715
1198 --> 39
48 --> 749
749 --> 39
32 --> 749
48 --> 301
41 --> 604
41 --> 310
41 --> 301
32 --> 310
41 --> 101
48 --> 1327
1327 --> 39
48 --> 271
48 --> 677
32 --> 110
208 --> 41
1121 --> 39
352 --> 39
41 --> 117
348 --> 38
522 --> 39
48 --> 824
824 --> 39
48 --> 740
41 --> 740
740 --> 39
677 --> 39
48 --> 201
38 --> 271
48 --> 604
48 --> 110
604 --> 39
48 --> 592
592 --> 39
48 --> 522
48 --> 310
48 --> 270
48 --> 225
41 --> 225
41 --> 170
189 --> 38
48 --> 475
475 --> 39
48 --> 438
438 --> 39
89 --> 41
65 --> 41
```

```
48 --> 60
60 --> 39
48 --> 352
48 --> 338
48 --> 161
310 --> 39
38 --> 286
48 --> 237
271 --> 39
270 --> 39
225 --> 39
48 --> 255
255 --> 39
48 --> 65
237 --> 39
36 --> 237
65 --> 39
48 --> 189
189 --> 39
48 --> 179
179 --> 39
32 --> 179
32 --> 38
32 --> 48
32 --> 39
48 --> 170
170 --> 39
170 --> 38
48 --> 147
147 --> 39
41 --> 110
110 --> 39
110 --> 38
89 --> 38
48 --> 101
48 --> 94
48 --> 89
101 --> 39
94 --> 39
89 --> 39
48 --> 79
48 --> 41
48 --> 36
79 --> 39
32 --> 41
48 --> 49
49 --> 38
48 --> 39
48 --> 38
41 --> 39
41 --> 38
38 --> 39
37 --> 38
41 --> 36
36 --> 39
36 --> 38
None
```

```
with 10 percent of the dataset I got 124 pairs.

225 --> 60
89 --> 260
89 --> 60
1715 --> 604
604 --> 3966
60 --> 189
476 --> 38
48 --> 1121
1715 --> 39
2238 --> 39
32 --> 286
2238 --> 38
855 --> 39
32 --> 101
352 --> 1859
9 --> 310
41 --> 1327
48 --> 1859
1859 --> 39
32 --> 271
48 --> 1715
1198 --> 39
48 --> 749
749 --> 39
65 --> 9
48 --> 301
41 --> 604
41 --> 310
41 --> 301
32 --> 310
41 --> 101
48 --> 1327
1327 --> 39
48 --> 271
65 --> 201
48 --> 1146
1146 --> 39
32 --> 110
208 --> 41
1121 --> 39
352 --> 41
352 --> 39
249 --> 237
41 --> 117
32 --> 117
522 --> 39
48 --> 9
48 --> 846
824 --> 39
48 --> 740
19 --> 38
533 --> 39
48 --> 201
```

```
38 --> 271
48 --> 604
48 --> 110
41 --> 475
604 --> 39
48 --> 533
48 --> 522
48 --> 310
48 --> 270
48 --> 225
41 --> 225
41 --> 170
189 --> 38
41 --> 237
48 --> 475
475 --> 39
48 --> 438
438 --> 39
89 --> 41
65 --> 41
48 --> 60
60 --> 39
48 --> 352
48 --> 161
310 --> 39
38 --> 286
48 --> 237
271 --> 39
270 --> 39
225 --> 39
48 --> 255
255 --> 39
48 --> 65
237 --> 39
36 --> 237
65 --> 39
48 --> 179
179 --> 39
32 --> 179
32 --> 38
32 --> 48
32 --> 39
48 --> 170
170 --> 39
170 --> 38
48 --> 147
147 --> 39
41 --> 110
110 --> 39
110 --> 38
89 --> 38
48 --> 101
48 --> 94
48 --> 89
101 --> 39
94 --> 39
89 --> 39
```

```
48 --> 79
48 --> 41
48 --> 36
79 --> 39
32 --> 41
48 --> 39
48 --> 38
41 --> 39
41 --> 38
38 --> 39
37 --> 38
41 --> 36
36 --> 39
36 --> 38
None
```

with 15 percent of the dataset I got 123 pairs.

```
8978 --> 39
225 --> 60
89 --> 60
1715 --> 604
604 --> 3966
48 --> 1121
65 --> 1327
1715 --> 39
65 --> 19
2238 --> 39
32 --> 286
32 --> 156
156 --> 39
2238 --> 38
249 --> 36
855 --> 39
32 --> 101
352 --> 1859
9 --> 310
41 --> 1327
32 --> 271
48 --> 1715
1198 --> 39
749 --> 39
65 --> 9
48 --> 301
41 --> 310
41 --> 301
32 --> 310
41 --> 101
48 --> 1344
1344 --> 39
48 --> 1327
1327 --> 39
48 --> 271
65 --> 201
48 --> 1146
41 --> 1146
```

```
1146 --> 39
32 --> 110
208 --> 41
1121 --> 39
352 --> 39
249 --> 237
976 --> 41
41 --> 117
32 --> 117
522 --> 39
48 --> 9
48 --> 846
48 --> 740
19 --> 38
533 --> 39
48 --> 201
38 --> 271
48 --> 604
48 --> 110
41 --> 475
604 --> 39
48 --> 522
48 --> 310
48 --> 270
48 --> 225
41 --> 225
41 --> 170
41 --> 237
48 --> 475
475 --> 39
48 --> 438
438 --> 39
89 --> 41
65 --> 41
48 --> 60
60 --> 39
48 --> 352
41 --> 147
48 --> 161
310 --> 39
38 --> 286
48 --> 237
271 --> 39
270 --> 39
225 --> 39
48 --> 255
255 --> 39
48 --> 65
237 --> 39
36 --> 237
65 --> 39
179 --> 39
32 --> 179
32 --> 38
32 --> 48
32 --> 39
48 --> 170
```

```
170 --> 39
170 --> 38
48 --> 147
41 --> 110
110 --> 39
110 --> 38
89 --> 38
48 --> 101
48 --> 94
48 --> 89
101 --> 39
94 --> 39
89 --> 39
48 --> 79
41 --> 79
48 --> 41
48 --> 36
79 --> 39
32 --> 41
48 --> 39
48 --> 38
41 --> 39
41 --> 38
38 --> 39
37 --> 38
41 --> 36
36 --> 39
36 --> 38
None


with 20 percent of the dataset I got 117 pairs.

8978 --> 39
225 --> 60
89 --> 60
1715 --> 604
48 --> 1121
65 --> 1327
2238 --> 39
32 --> 156
156 --> 39
2238 --> 38
249 --> 36
855 --> 39
352 --> 1859
9 --> 310
89 --> 1327
41 --> 1327
32 --> 271
1198 --> 39
749 --> 39
65 --> 9
48 --> 301
41 --> 310
41 --> 301
32 --> 310
```

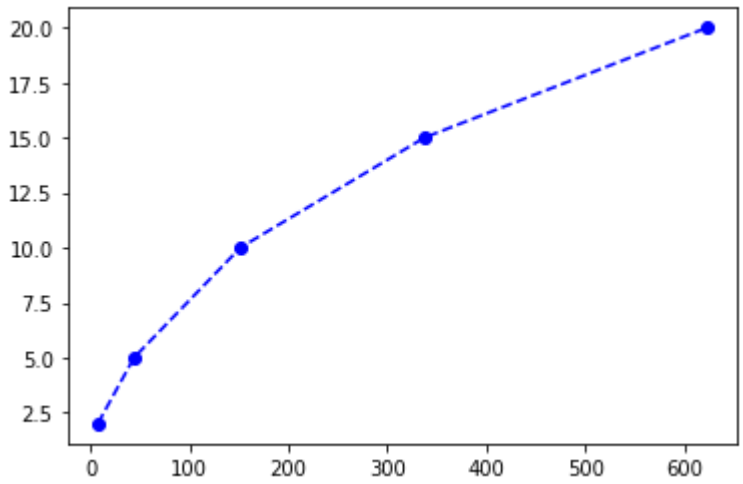
```
41 --> 101
48 --> 1344
1344 --> 39
48 --> 1327
1327 --> 39
48 --> 271
48 --> 117
65 --> 201
48 --> 1146
41 --> 1146
1146 --> 39
32 --> 110
208 --> 41
1121 --> 39
352 --> 39
249 --> 237
976 --> 41
41 --> 117
32 --> 117
41 --> 161
522 --> 39
48 --> 846
824 --> 39
769 --> 647
48 --> 740
533 --> 39
48 --> 201
38 --> 271
48 --> 604
48 --> 110
41 --> 475
604 --> 39
48 --> 310
48 --> 225
41 --> 225
41 --> 170
32 --> 170
41 --> 237
48 --> 475
475 --> 39
48 --> 438
438 --> 39
89 --> 41
65 --> 41
48 --> 60
60 --> 39
48 --> 352
41 --> 147
48 --> 161
310 --> 39
38 --> 286
48 --> 237
271 --> 39
270 --> 39
225 --> 39
48 --> 255
255 --> 39
```

```
48 --> 65
237 --> 39
36 --> 237
65 --> 39
179 --> 39
32 --> 179
32 --> 38
32 --> 48
32 --> 39
48 --> 170
170 --> 39
170 --> 38
48 --> 147
147 --> 39
41 --> 110
110 --> 39
110 --> 38
89 --> 38
48 --> 101
48 --> 89
101 --> 39
89 --> 39
48 --> 79
48 --> 41
48 --> 36
79 --> 39
32 --> 41
48 --> 39
48 --> 38
41 --> 39
41 --> 38
38 --> 39
37 --> 38
41 --> 36
36 --> 39
36 --> 38
None
```

Graph

In [11]:
```
1 plt.plot(times, [x for x in sections], 'bo--')
```

Out[11]: [<matplotlib.lines.Line2D at 0x7f9130dd1640>]



Netflix

In [ ]:
```
1 netflix = pd.read_csv('netflix.data', delimiter=" ",engine='python', on_bad_
2 netflix_L = netflix.values.tolist()
```

In [ ]:
```
1 netflix_NN = []
2 for x in netflix_L:
3     netflix_NN.append([i for i in x if str(i) != 'nan'])
```

In [ ]:
```
1 sections = [2,5,10,15, 20]
2 associationN = []
3 for i, x in enumerate(sections):
4     new_list = netflix_NN[0:int(len(netflix_NN)*(sections[i]/100))]
5     associationN.append(new_list)
```

In [ ]:
```
1 times = []
2 pairslen = []
3 actualpairs = []
4 for x in range(5):
5     print(x)
6     results = apyori(associationN[x],0.01,2000)
7     times.append(results[0])
8     pairslen.append(len(results[1]))
9     actualpairs.append(results[1])
```

In [ ]:
```
1 for x in range(len(sections)):
2     print()
3     print()
4     print('with %g percent of the dataset I got %d pairs.' % (sections[x], p
5     print()
6     print(output(actualpairs[x]))
```

In [ ]:
```
1 pl.plot(times, [x for x in sections], 'bo--')
```

Conclusions

This is was the Multistage implementation of the Apyori algorithm. I feel that this was faster than the library algorithm as well as the normal apyori algorithm, reason being that I was able to run a max of 20% of the dataset and was not taking as long as the other algorithms. As for the pairs I also believed I maintained a consistent number of pairs throughout all my subsections. Unfortunately I am still not able to run my netflix data set. I've added a picture that you can see the contains the "kernel has died" error.