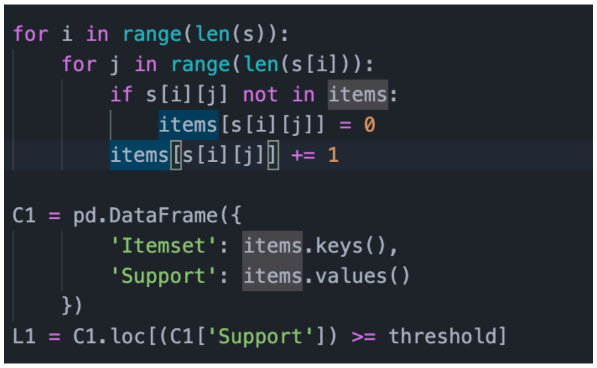
Andrew Adamah
100693132
4030 Project Part I

**** ALL TRIALS OF NETFLIX DATASET RAN FOR LONGER THAN 5H**

Apriori

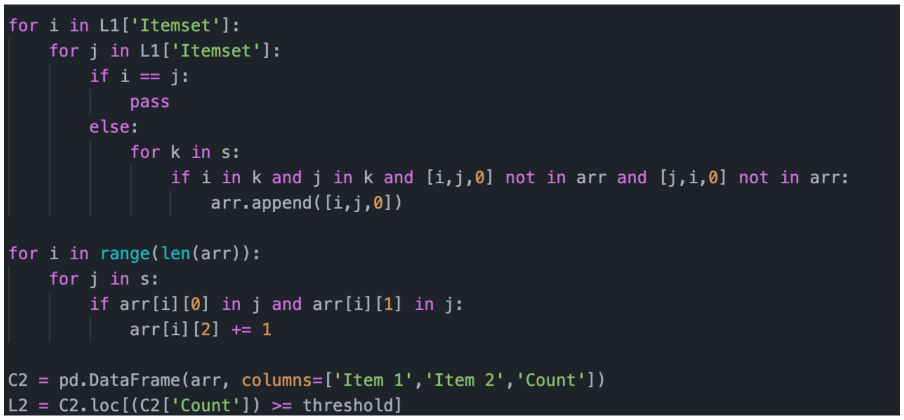
PASS 1



```
for i in range(len(s)):
    for j in range(len(s[i])):
        if s[i][j] not in items:
            items[s[i][j]] = 0
        items[s[i][j]] += 1

C1 = pd.DataFrame({
        'Itemset': items.keys(),
        'Support': items.values()
    })
L1 = C1.loc[(C1['Support']) >= threshold]
```

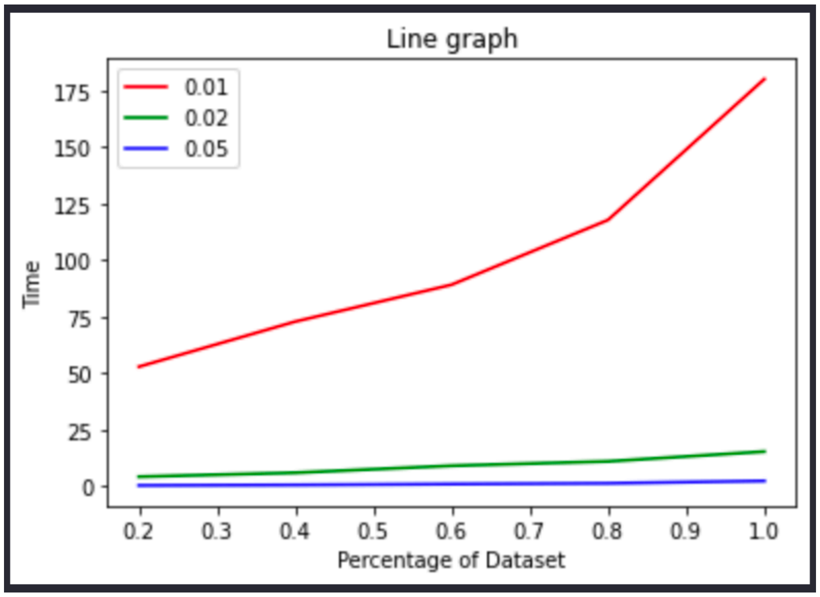
Pass 1 counts frequent singles in the dataset

PASS 2



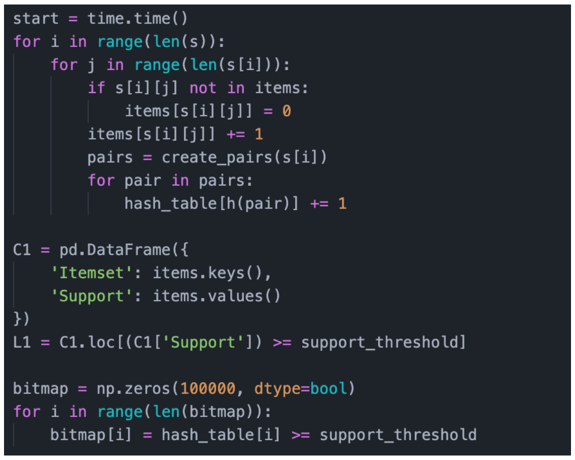
```
for i in L1['Itemset']:
    for j in L1['Itemset']:
        if i == j:
            pass
        else:
            for k in s:
                if i in k and j in k and [i,j,0] not in arr and [j,i,0] not in arr:
                    arr.append([i,j,0])

for i in range(len(arr)):
    for j in s:
        if arr[i][0] in j and arr[i][1] in j:
            arr[i][2] += 1

C2 = pd.DataFrame(arr, columns=['Item 1','Item 2','Count'])
L2 = C2.loc[(C2['Count']) >= threshold]
```

Pass 2 counts frequent pairs in the dataset



PCY

PASS 1

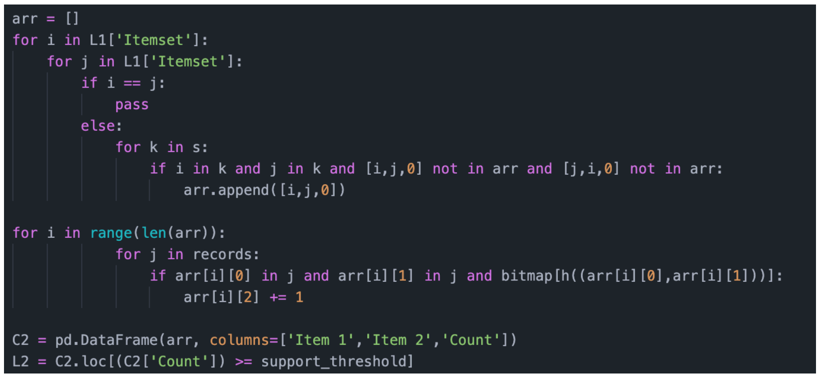


```
start = time.time()
for i in range(len(s)):
    for j in range(len(s[i])):
        if s[i][j] not in items:
            items[s[i][j]] = 0
        items[s[i][j]] += 1
        pairs = create_pairs(s[i])
        for pair in pairs:
            hash_table[h(pair)] += 1

C1 = pd.DataFrame({
    'Itemset': items.keys(),
    'Support': items.values()
})
L1 = C1.loc[(C1['Support']) >= support_threshold]

bitmap = np.zeros(100000, dtype=bool)
for i in range(len(bitmap)):
    bitmap[i] = hash_table[i] >= support_threshold
```

Pass 1 counts frequent singles, creates pairs and hashes the pairs to buckets

Bitmap is created in-between pass 1 and 2

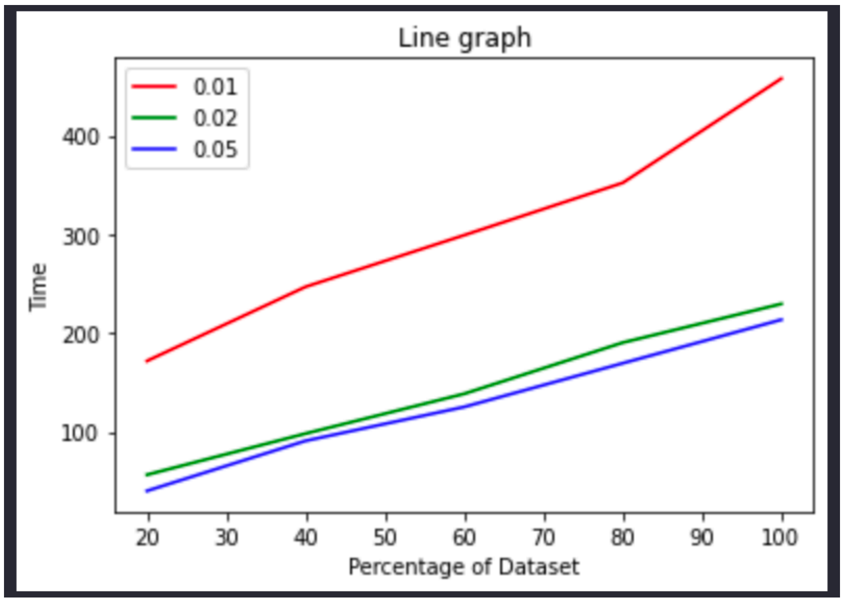
PASS 2



```
arr = []
for i in L1['Itemset']:
    for j in L1['Itemset']:
        if i == j:
            pass
        else:
            for k in s:
                if i in k and j in k and [i,j,0] not in arr and [j,i,0] not in arr:
                    arr.append([i,j,0])

for i in range(len(arr)):
            for j in records:
                if arr[i][0] in j and arr[i][1] in j and bitmap[h((arr[i][0],arr[i][1]))]:
                    arr[i][2] += 1

C2 = pd.DataFrame(arr, columns=['Item 1','Item 2','Count'])
L2 = C2.loc[(C2['Count']) >= support_threshold]
```

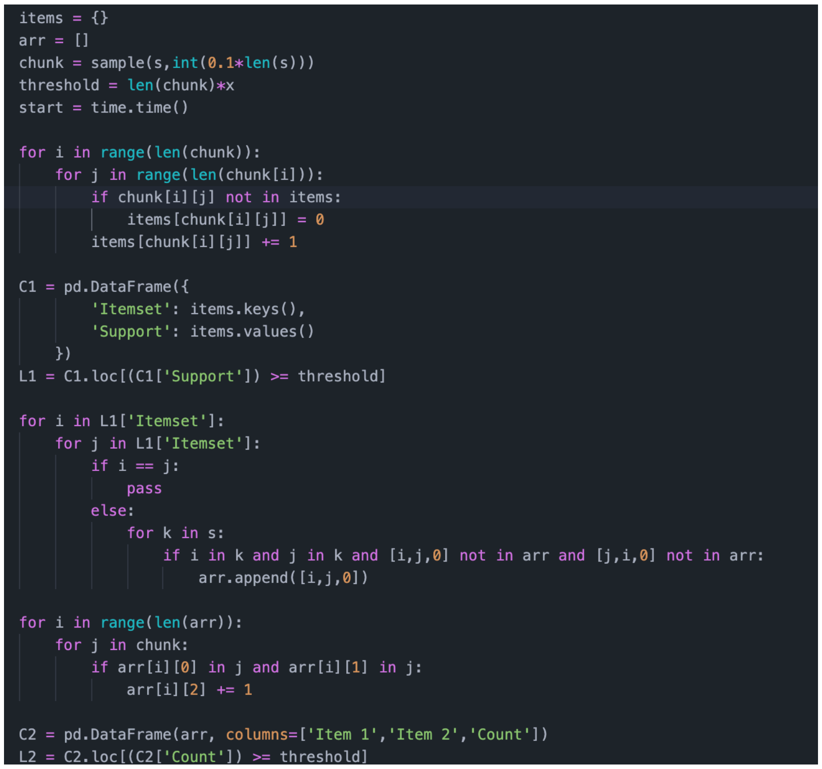
Pass 2 counts frequent pairs in baskets based on which bitmap they map to.

Only count pairs that are frequent and map to the bitmap



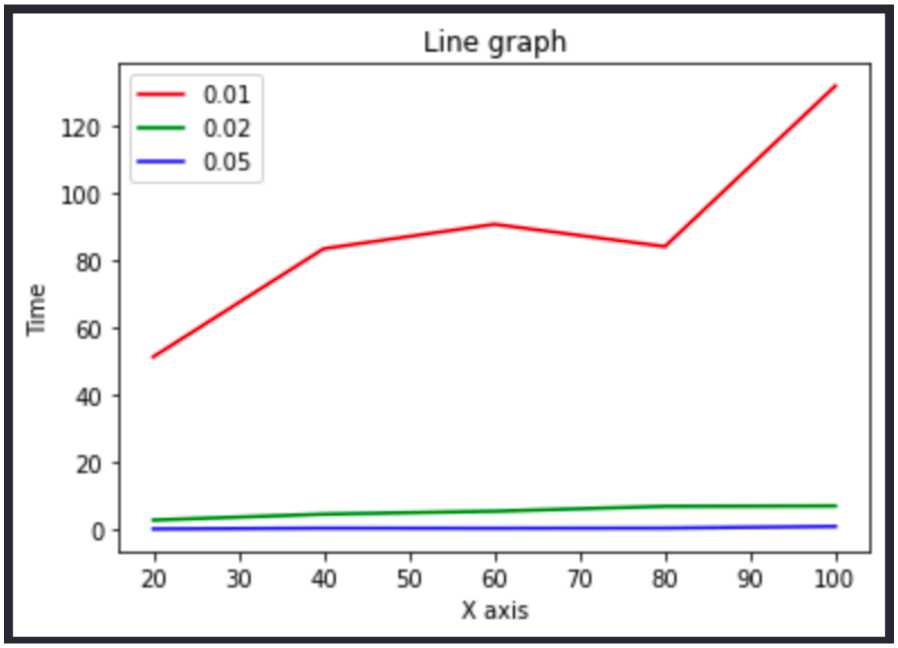
Random Sampling

PASS 1



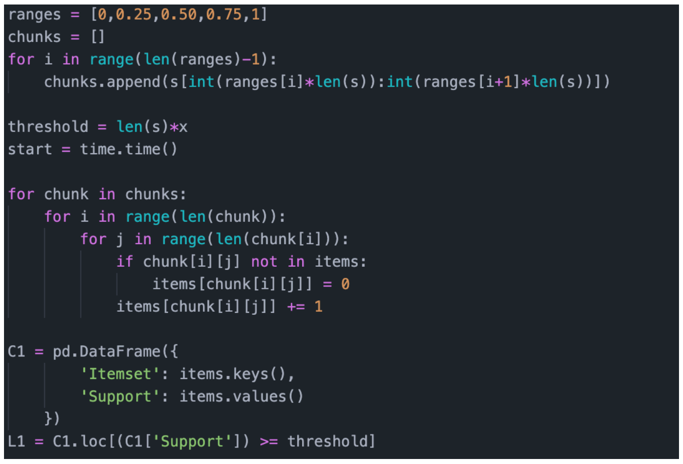
```
items = {}
arr = []
chunk = sample(s,int(0.1*len(s)))
threshold = len(chunk)*x
start = time.time()

for i in range(len(chunk)):
    for j in range(len(chunk[i])):
        if chunk[i][j] not in items:
            items[chunk[i][j]] = 0
        items[chunk[i][j]] += 1

C1 = pd.DataFrame({
        'Itemset': items.keys(),
        'Support': items.values()
    })
L1 = C1.loc[(C1['Support']) >= threshold]

for i in L1['Itemset']:
    for j in L1['Itemset']:
        if i == j:
            pass
        else:
            for k in s:
                if i in k and j in k and [i,j,0] not in arr and [j,i,0] not in arr:
                    arr.append([i,j,0])

for i in range(len(arr)):
    for j in chunk:
        if arr[i][0] in j and arr[i][1] in j:
            arr[i][2] += 1

C2 = pd.DataFrame(arr, columns=['Item 1','Item 2','Count'])
L2 = C2.loc[(C2['Count']) >= threshold]
```

Uses a random 10% chunk of the dataset and runs Apriori on this chunk of data only



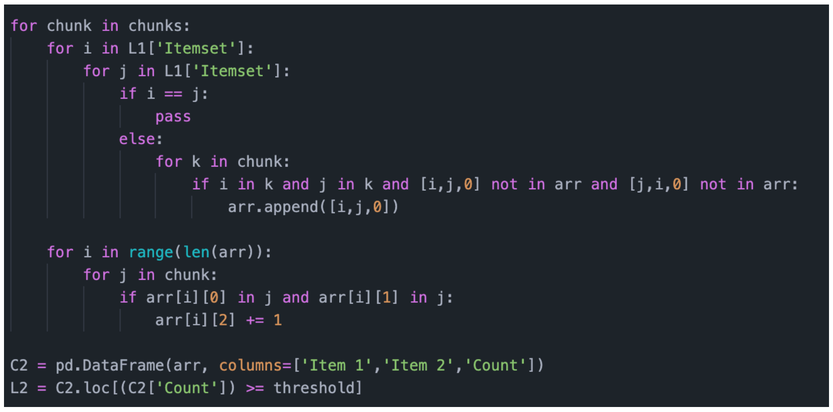
SON

PASS 1

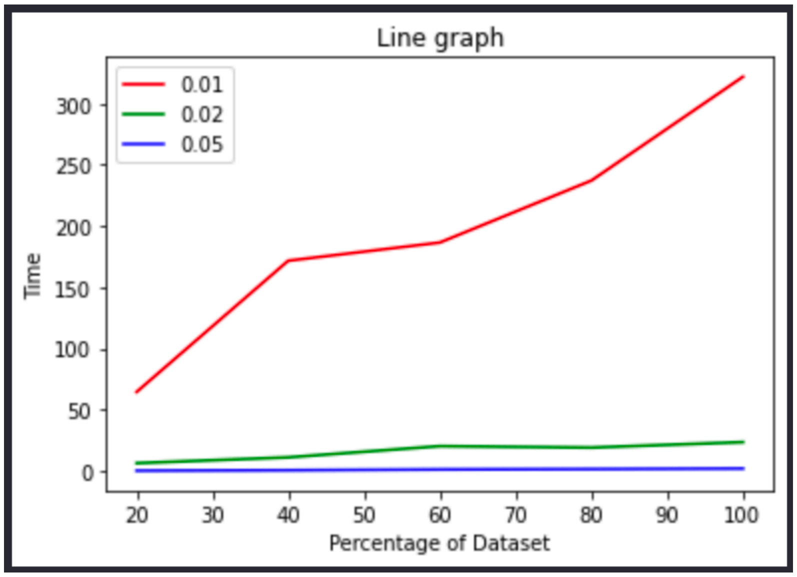


```python
ranges = [0,0.25,0.50,0.75,1]
chunks = []
for i in range(len(ranges)-1):
    chunks.append(s[int(ranges[i]*len(s)):int(ranges[i+1]*len(s))])

threshold = len(s)*x
start = time.time()

for chunk in chunks:
    for i in range(len(chunk)):
        for j in range(len(chunk[i])):
            if chunk[i][j] not in items:
                items[chunk[i][j]] = 0
            items[chunk[i][j]] += 1

C1 = pd.DataFrame({
        'Itemset': items.keys(),
        'Support': items.values()
    })
L1 = C1.loc[(C1['Support']) >= threshold]
```

Splits dataset into even chunks of data and runs priori on these chunks

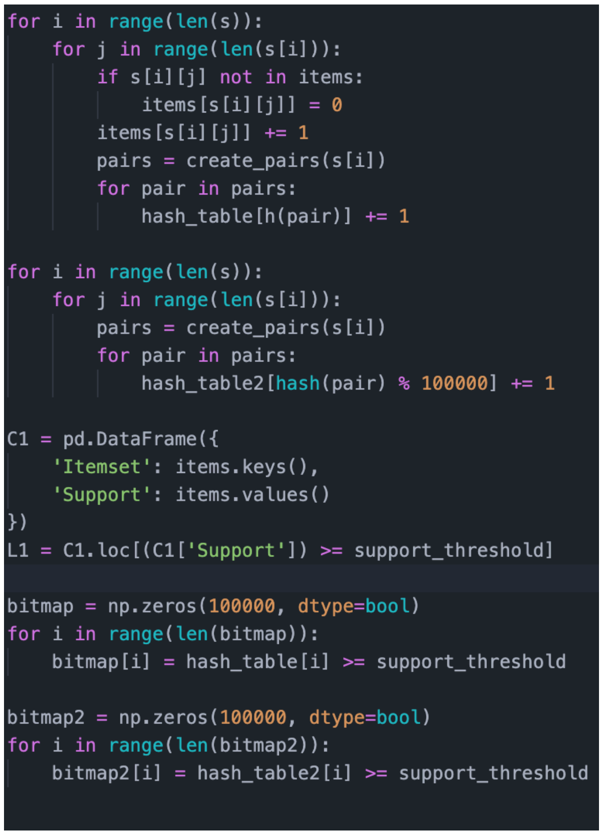


```python
for chunk in chunks:
    for i in L1['Itemset']:
        for j in L1['Itemset']:
            if i == j:
                pass
            else:
                for k in chunk:
                    if i in k and j in k and [i,j,0] not in arr and [j,i,0] not in arr:
                        arr.append([i,j,0])

    for i in range(len(arr)):
        for j in chunk:
            if arr[i][0] in j and arr[i][1] in j:
                arr[i][2] += 1

C2 = pd.DataFrame(arr, columns=['Item 1','Item 2','Count'])
L2 = C2.loc[(C2['Count']) >= threshold]
```



Multi-hash

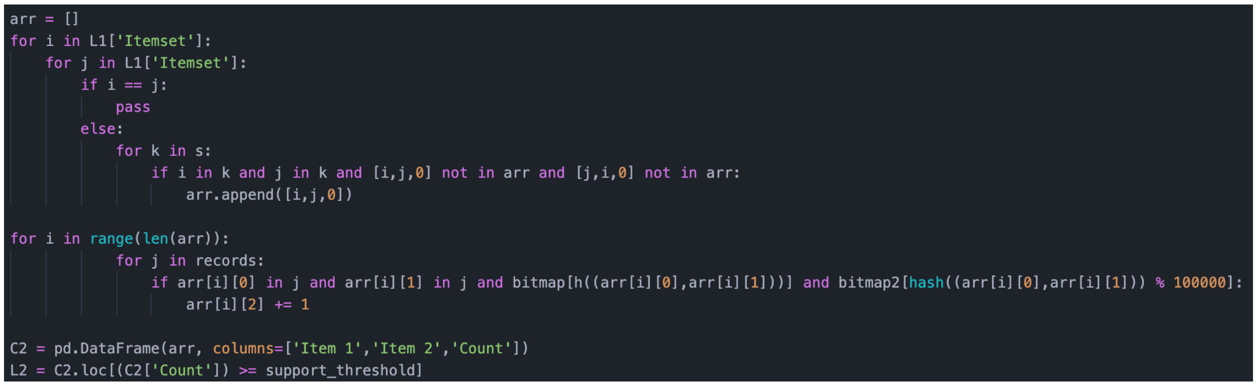
PASS 1



```python
for i in range(len(s)):
    for j in range(len(s[i])):
        if s[i][j] not in items:
            items[s[i][j]] = 0
        items[s[i][j]] += 1
        pairs = create_pairs(s[i])
        for pair in pairs:
            hash_table[h(pair)] += 1

for i in range(len(s)):
    for j in range(len(s[i])):
        pairs = create_pairs(s[i])
        for pair in pairs:
            hash_table2[hash(pair) % 100000] += 1

C1 = pd.DataFrame({
    'Itemset': items.keys(),
    'Support': items.values()
})
L1 = C1.loc[(C1['Support']) >= support_threshold]

bitmap = np.zeros(100000, dtype=bool)
for i in range(len(bitmap)):
    bitmap[i] = hash_table[i] >= support_threshold

bitmap2 = np.zeros(100000, dtype=bool)
for i in range(len(bitmap2)):
    bitmap2[i] = hash_table2[i] >= support_threshold
```

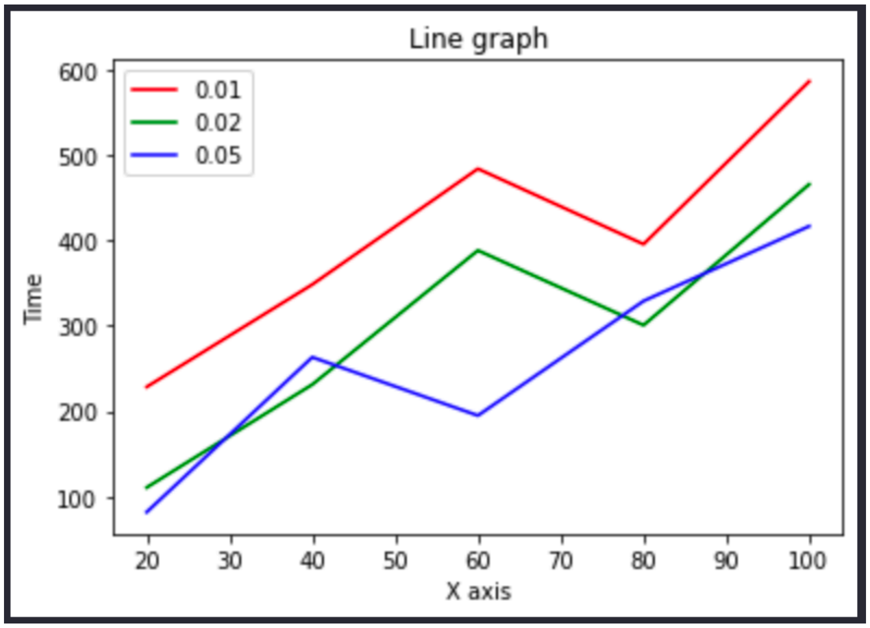
Uses 2 has functions and 2 bitmaps instead of 1 to perform the PCY algorithm

PASS 2

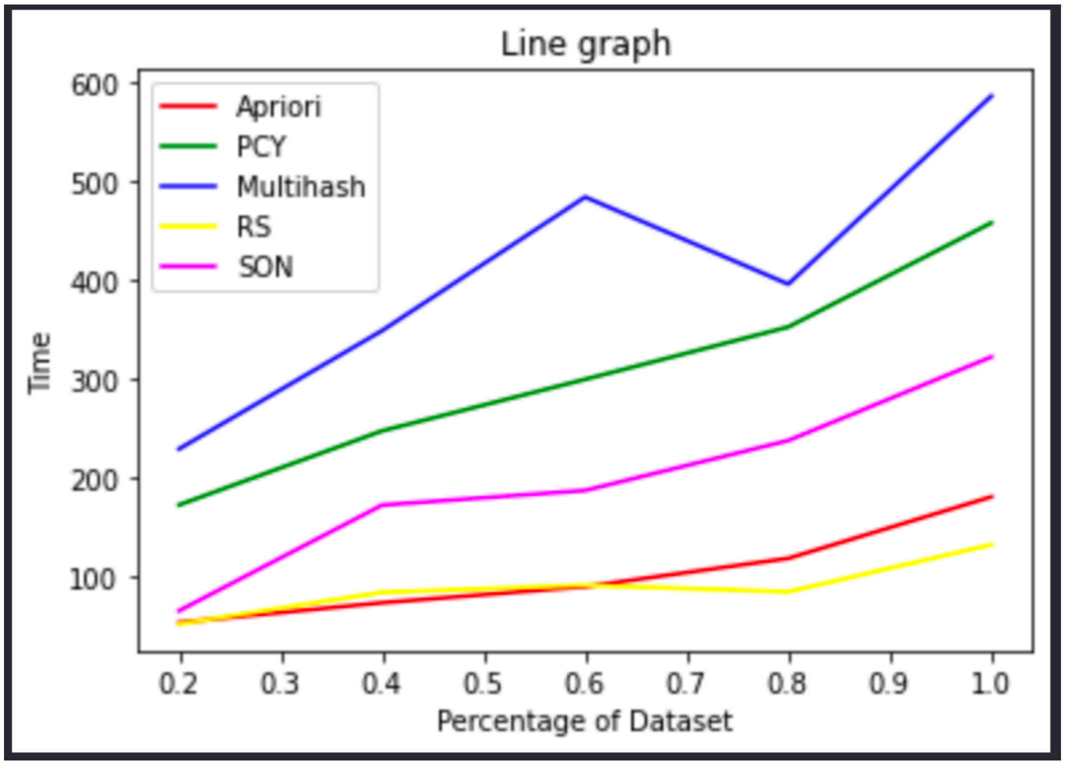


```python
arr = []
for i in L1['Itemset']:
    for j in L1['Itemset']:
        if i == j:
            pass
        else:
            for k in s:
                if i in k and j in k and [i,j,0] not in arr and [j,i,0] not in arr:
                    arr.append([i,j,0])

for i in range(len(arr)):
            for j in records:
                if arr[i][0] in j and arr[i][1] in j and bitmap[h((arr[i][0],arr[i][1]))] and bitmap2[hash((arr[i][0],arr[i][1])) % 100000]:
                    arr[i][2] += 1

C2 = pd.DataFrame(arr, columns=['Item 1','Item 2','Count'])
L2 = C2.loc[(C2['Count']) >= support_threshold]
```

Only count pairs that are frequent and map to bitmap 1 and bitmap 2



APRIORI VS. PCY VS. MULTIHASH VS. RS VS. SON



The graph above shows the times taken to compute the frequent pairs for 0.01 threshold for different percentages of the dataset. From the graph, we can see that the Aproiri algorithm outperforms the PCY algorithm. The PCY is slower than the Apriori because of the hash tables and bitmaps that it uses.The multi hash is slower than the PCY because it uses extra hash tables and bitmaps. The SON algorithm uses chunks of the dataset and runs the Apriori algorithm on these chunks. It takes significantly longer than the normal algorithm and is not as efficient in finding the pairs. The random sampling only uses 10% of the dataset and runs the Apriori algorithm on it, however the times were quite similar to the Apriori using the full dataset. In conclusion the best algorithm is the Apriori algorithm.