DM ASSIGNMENT – 1 REPORT

Name: Rahul Varma

Roll No: S20200010212

Memory Comparison of Apriori and PCY:

Two data taken for Experiment:

1) apriori.csv (given in assignment)

2)cla.csv (example taken in class)

Apriori Algorithm:

Command Line Argument:

python {code_file} {data_file} {min_support} {min_confidence}

Eg: python S20200010212_apriori.py apriori.csv 0.4 0.7

Install some useful dependencies:

1)pip install numpy

2)pip install matplotlib (For Graph)

Sample csv is taken which is given in the class room

The data taken is:

apriori.csv:

	A	B	C	D
1	I4	I3	I2	
2	I4			
3	I1	I4	I3	
4	I2	I1	I5	
5	I3			
6	I4	I3		
7	I4	I1	I3	I5
8	I4	I3		
9	I4			
10	I3	I5		

- Using sys command line data is taken and converted the minimum support from percentage to normal value.
- If we get decimal value they simply consider the ceil value.
- Store the data into some list and compute the logic for apriori.
- Using strip the data to converted into nice format (by using the csv which is directly downloaded has some extra spaces, so I used strip and removed the unwanted spaces)
- Taking the minimum support 0.4 and minimum confidence 0.7 for the above data and running the apriori using command line argument:
- python apriori.py apriori.csv 0.4 0.7
- While printing the output we should check the minimum confidence given in the input and print only the data which satisfy the minimum confidence value.

The output is: (No spaces in the format).

```
PS D:\5th sem\DM\Final_Ass> python S20200010212_apriori.py apriori.csv 0.4 0.7
{I4}{I3}[0.5,0.7143]
{I3}{I4}[0.5,0.7143]
PS D:\5th sem\DM\Final_Ass>
```

The format is taken as mentioned in the class room (for output).

Sample data 2: (named as cla.csv)

cla.csv:

```
I1,I2,I5
I2,I4
I2,I3
I1,I2,I4
I1,I3
I2,I3
I1,I3
I1,I2,I3,I5
I1,I2,I3
```

Output for the data 2:

For showing the three item-sets:

```
PS D:\5th sem\DM\Final_Ass> python S20200010212_apriori.py cla.csv 0.22 0.4
{I1}{I2}[0.44,0.6667]
{I2}{I1}[0.44,0.5714]
{I5}{I1}[0.22,1.0]
{I1}{I3}[0.44,0.6667]
{I3}{I1}[0.44,0.6667]
{I5}{I2}[0.22,1.0]
{I4}{I2}[0.22,1.0]
{I2}{I3}[0.44,0.5714]
{I3}{I2}[0.44,0.6667]
{I5}{I1,I2}[0.22,1.0]
{I1,I2}{I5}[0.22,0.5]
{I1,I5}{I2}[0.22,1.0]
{I2,I5}{I1}[0.22,1.0]
{I1,I2}{I3}[0.22,0.5]
{I1,I3}{I2}[0.22,0.5]
{I2,I3}{I1}[0.22,0.5]
PS D:\5th sem\DM\Final_Ass>
```

PCY Algorithm: (Same command line argument is taken as apriori).

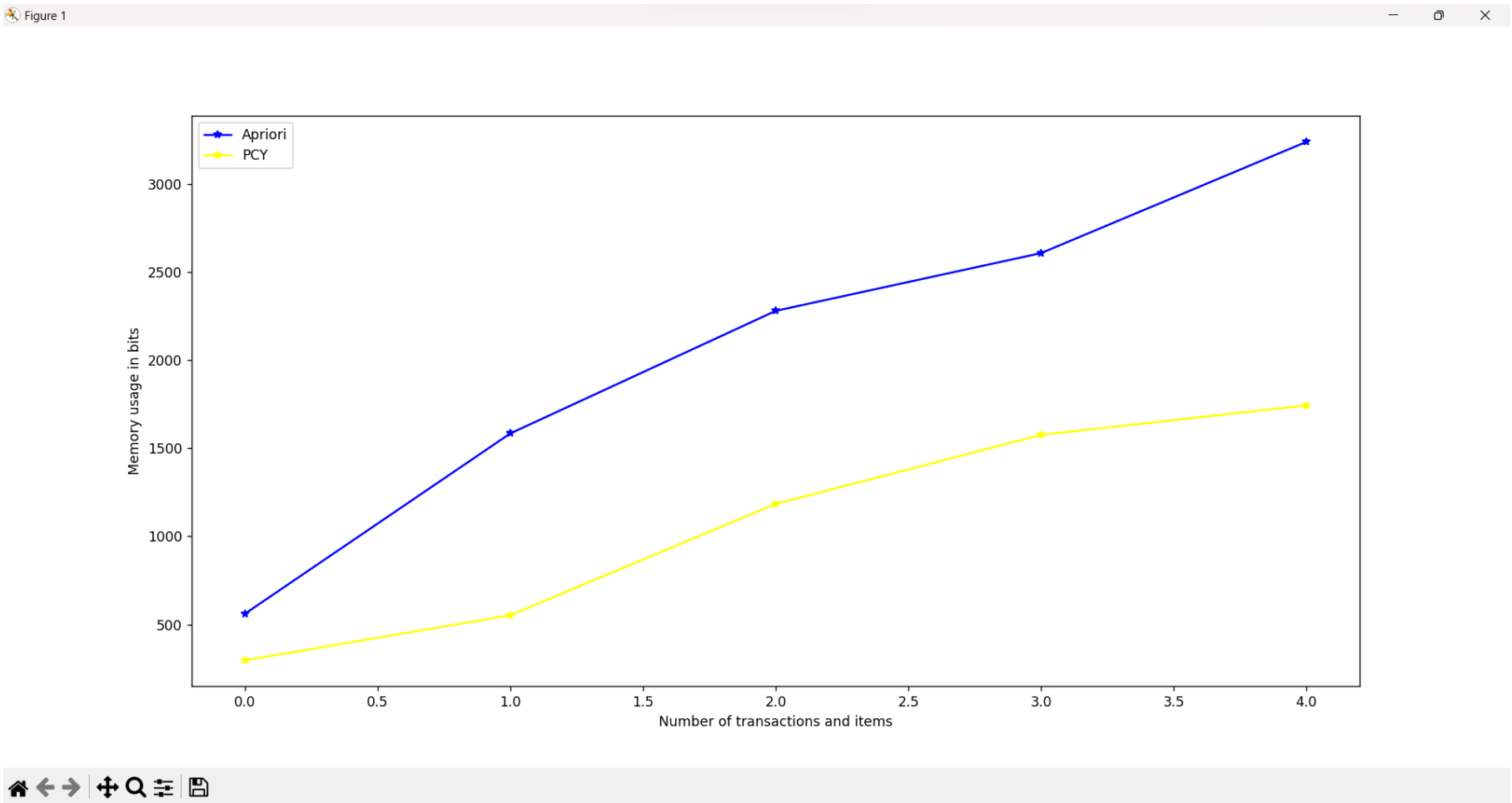
Eg: python S20200010212_pcy.py apriori.csv 0.4 0.7

- Modified the apriori algorithm to include a hash table to eliminate the memory requirement while counting 2-itemsets.
- Hash Function taken is:

$$h(x, y) = \big((order\ of\ x) * 10 + order\ of\ y\big) mod\ 7$$

- In S20200010212_pcy.py file I implemented both apriori and pcy (2-itemsets is implemented using hash table)
- and stored the memory used by both algo using tracemalloc in different scenarios and plotted the graph between memory usage of apriori and pcy algo.
- Two lines in the Graph are of memory usage using apriori and pcy.

The Graph by using the same data given in assignment (apriori.csv):



By varying the number of transactions in the database and the number of items the graph is plotted using matplotlib.

The output for the pcy of memory value :

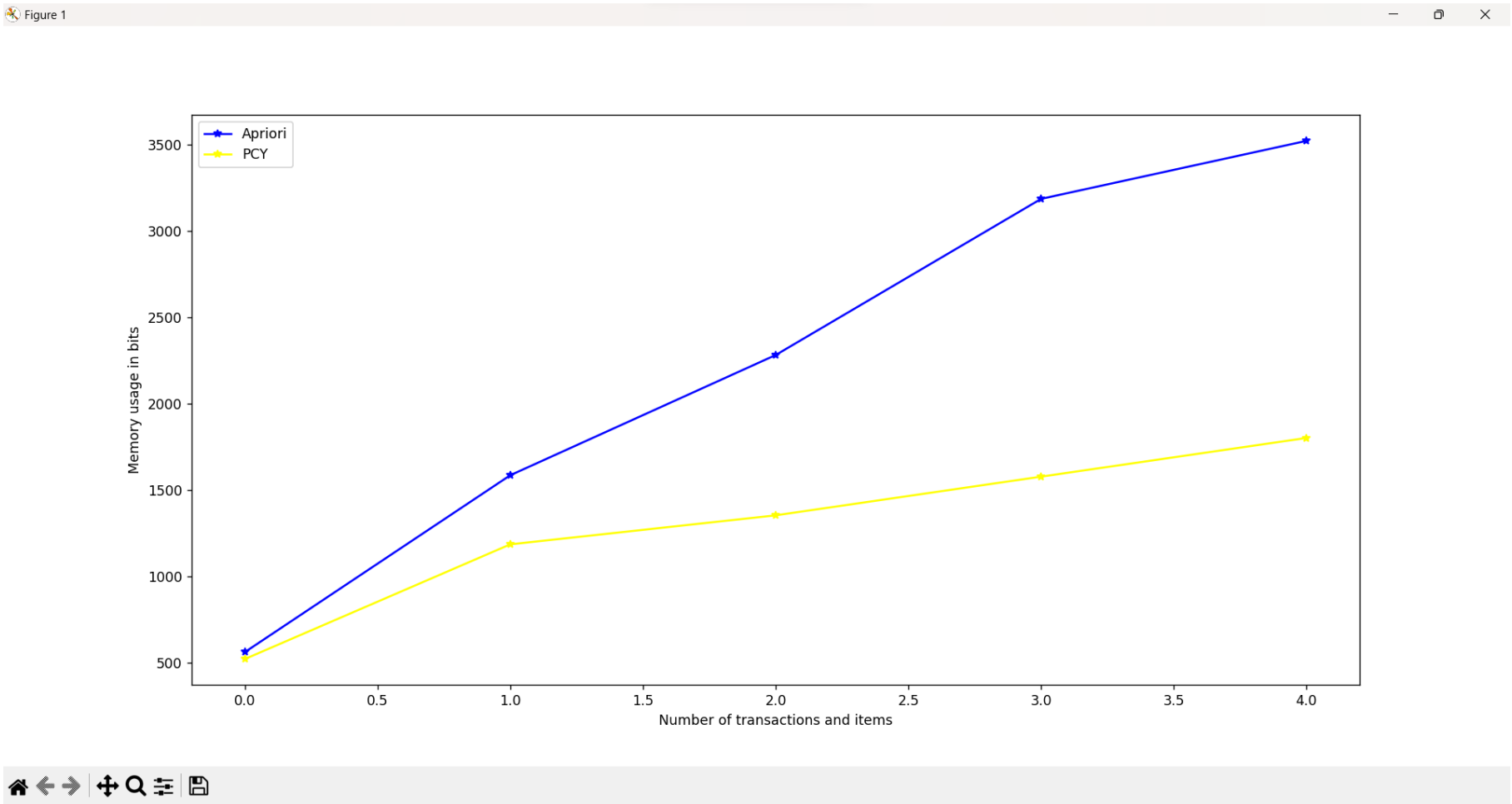
```
PS D:\5th sem\DM\Final_Ass> python S20200010212_pcy.py apriori.csv 0.4 0.7
{I3}{I4}[0.5,0.7143]
{I4}{I3}[0.5,0.7143]

The data of memory after varying the number of transactions in the database and the number of items:
Memory for Apriori:  [560, 1584, 2280, 2608, 3240]
Memory for PCY:  [296, 552, 1184, 1576, 1744]
PS D:\5th sem\DM\Final_Ass>
```

While submitting code file the data memory part is commented.

For the second data: (cla.csv)

```
I1,I2,I5
I2,I4
I2,I3
I1,I2,I4
I1,I3
I2,I3
I1,I3
I1,I2,I3,I5
I1,I2,I3
```

Graph for the second data: (memory value for pcy is changed which are also present in the next page)



Output:

```
PS D:\5th sem\DM\Final_Ass> python S20200010212_pcy.py cla.csv 0.22 0.4
{I5}{I1,I2}[0.22,1.0]
{I1,I2}{I5}[0.22,0.5]
{I1,I5}{I2}[0.22,1.0]
{I2,I5}{I1}[0.22,1.0]
{I1,I2}{I3}[0.22,0.5]
{I1,I3}{I2}[0.22,0.5]
{I2,I3}{I1}[0.22,0.5]
{I1}{I2}[0.44,0.6667]
{I2}{I1}[0.44,0.5714]
{I5}{I1}[0.22,1.0]
{I1}{I3}[0.44,0.6667]
{I3}{I1}[0.44,0.6667]
{I5}{I2}[0.22,1.0]
{I4}{I2}[0.22,1.0]
{I2}{I3}[0.44,0.5714]
{I3}{I2}[0.44,0.6667]

The data of memory after varying the number of transactions in the database and the number of items:
Memory for Apriori:  [560, 1584, 2280, 3184, 3520]
Memory for PCY:  [520, 1184, 1352, 1576, 1800]
PS D:\5th sem\DM\Final_Ass>
```

Conclusion:

By seeing the graph one can easily say that apriori is taking more memory as the number of transitions and items are increased.

And the memory usage of pcy algo in 2-itemsets is very low and compared to apriori as the number of transitions and items are increased.

So, Using PCY in apriori decreases the usage of memory and code can run in more efficient way.

The Hash Function Used is:

$$h(x, y) = \big((order\ of\ x) * 10 + order\ of\ y\big) mod\ 7$$

Therefore the Final Conclusion is:

PCY algorithm improves the memory efficiency of the apriori algorithm.

THANK YOU