Lab Assignment 1 Report

Section 1 – Step II

1. How to run the code (StepII)

Task 1: Mining all Frequent Itemset

Task 2: Mining all Frequent Closed Itemset

a. Turn on Terminal and input: (direct to my .py folder)

cd "/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1"

b. Take turns run datasetA, datasetB and datasetC. Put the command line in Terminal For datasetA:

python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.3 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1'

- -f is the file path of datasetA
- -p is the path of output file
- For datasetA, B, C, minimal support [0.3, 0.2, 0.5] are all seem too large. So I set minimal support = 0.01 for all of them

c. The frequent itemset and statistics file are created. (For my task: step2_task1_datasetA_0.3_result.txt)

ps. The executing time is recorded in the **statistics.txt** file.

2. The modifications I made for Step II

1. **Changed OptionParser to argparse:** Because OptionParser cannot be used after Python 3.2, switch to “argparse” to set the parameters. See Figure.1
2. Added **memory_usage** to measure the efficency. See Figure1

```python
import os
import argparse
from itertools import chain, combinations
from collections import defaultdict
import time
from memory_profiler import memory_usage
from tqdm import tqdm

def parse_args():
    parser = argparse.ArgumentParser(description="Run Apriori Algorithm")
    parser.add_argument("-f", "--inputFile", type=str, required=True, help="Dataset file path")
    parser.add_argument("-p", "--outputPath", type=str, required=True, help="Output file path")
    parser.add_argument("-s", "--minSupport", type=float, required=True, help="Minimum support value")
    return parser.parse_args()
```

Figure.1

3. **Add write_result_files & write_closed_result_file**. See Figure2, Figure3

```python
def write_result_files(items, candidates_stats, dataset_name, min_support, output_path, total_time_task1, mem_usage_task1):
    result_filename_1 = f"{output_path}/step2_task1_{dataset_name}_{min_support}_result.txt"
    with open(result_filename_1, 'w') as f:
        for item, support in sorted(items, key=lambda x: x[1], reverse=True):
            itemset_str = "{" + ", ".join(item) + "}"
            f.write(f"{round(support * 100, 1)}%\t{itemset_str}\n")
    print(f"Frequent itemset results written to {result_filename_1}")

    result_filename_2 = f"{output_path}/step2_task1_{dataset_name}_{min_support}_stats.txt"
    with open(result_filename_2, 'w') as f:
        f.write(f"Total number of frequent itemsets: {len(items)}\n")

        for k, before, after in candidates_stats:
            f.write(f"{k}\t{before}\t{after}\n")
        f.write(f"\nTotal execution time for Task 1: {total_time_task1:.2f} seconds\n")
        f.write(f"Memory Usage for Task 1: {max(mem_usage_task1) - min(mem_usage_task1):.2f} MiB\n")

    print(f"Frequent itemset statistics written to {result_filename_2}")
```

Figure.2

```python
def write_closed_result_file(closedItems, dataset_name, min_support, output_path, total_time_task2, ratio):
    result_filename = f"{output_path}/step2_task2_{dataset_name}_{min_support}_closed_result.txt"
    with open(result_filename, 'w') as f:
        f.write(f"Total number of frequent closed itemsets: {len(closedItems)}\n")
        for item, support in sorted(closedItems, key=lambda x: x[1], reverse=True):
            itemset_str = "{" + ", ".join(item) + "}"
            f.write(f"{round(support * 100, 1)}%\t{itemset_str}\n")
        f.write(f"\nRatio of computation time compared to Task 1: {ratio:.2f}%\n")

    print(f"Closed itemset results written to {result_filename}")
```

Figure. 3

4. **Use itemset_str = "{" + ", ".join(item) + "}" as medium brackets (for the requirement of file format).** See Figure4

```python
def write_result_files(items, candidates_stats, dataset_name, min_support,
    result_filename_1 = f"{output_path}/step2_task1_{dataset_name}_{min_su
    with open(result_filename_1, 'w') as f:
        for item, support in sorted(items, key=lambda x: x[1], reverse=Tru
            itemset_str = "{" + ", ".join(item) + "}"
            f.write(f"{round(support * 100, 1)}%\t{itemset_str}\n")
    print(f"Frequent itemset results written to {result_filename_1}")
```

Figure. 4

5. **Add candidates_before, candidates_after to record the candidates when pruning and write to file 2.** See Figure5

```
candidates_before = len(currentLSet)
currentLSet = joinSet(currentLSet, k)
currentCSet = returnItemsWithMinSupport(currentLSet, transactionList, minSupport, freqSet)

candidates_after = len(currentCSet)
candidates_stats.append((k - 1, candidates_before, candidates_after))

currentLSet = currentCSet
k += 1
```

Figure.5

6. **Set runTask1 function to run in the main program.** See Figure6

```
def runTask1(input_file, min_support, output_path, dataset_name):
    start_time_task1 = time.time()

    def task1_func():
        inFile = dataFromFile(input_file)
        items, candidates_stats, freqSet = runApriori(inFile, min_support)
        return items, candidates_stats, freqSet

    mem_usage_task1, (items, candidates_stats, freqSet) = memory_usage(task1_func, retval=True)
```

Figure. 6

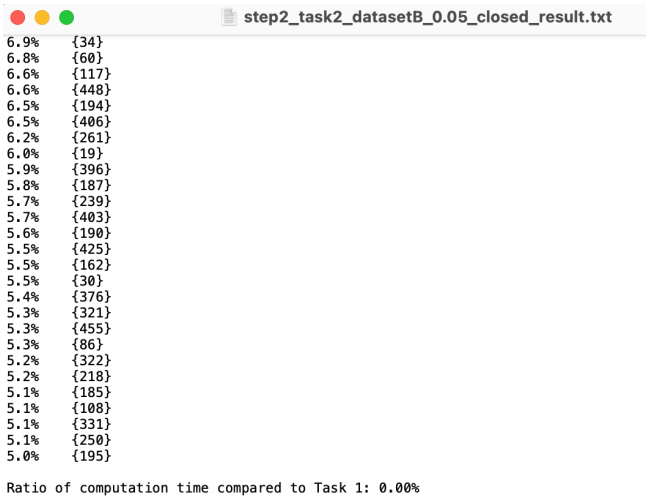
7. **Pass freqSet from runTask1 to findClosedItemsets in Task2:**
   - freqSet stores the frequent itemsets and their support values
   - In Task2, use **findClosedItemsets(items, freqSet)** to find the frequent closed itemset

```
def findClosedItemsets(freqItems, freqSet):
    closedItemsets = []
    # itemsets_by_length
    itemsets_by_length = defaultdict(list)
    for item, support in freqItems:
        itemsets_by_length[len(item)].append((item, support))

    for item, support in freqItems:
        isClosed = True
        # Check if there is a superset with the same support
        larger_len = len(item) + 1
        if larger_len in itemsets_by_length:
            for otherItem, otherSupport in itemsets_by_length[larger_len]:
                if item.issubset(otherItem) and support == otherSupport:
                    isClosed = False
                    break

        if isClosed:
            closedItemsets.append((item, support))

    return closedItemsets
```

Figure. 7

8. **Merge Task2 in main program (including time ratio calculation)**:

- Make sure **Task2 is executed after Task1**, so it is included in the main program
- Add the time computation of both Task1 and Task2. The ratio of computed time is stored in the **result file of Task2** (step2_task2_dataset_closed_result) **But the** ratio of computation time compared to Task1 is always 0.00% (refer to the picture), then I checked every min support value in each dataset and found the execution time for task 2 are all 0.00 seconds.



-> Check if **freqItems and freqSet** are empty in findClosedItemsets function and **runTask1 function**, so added the following:

```
if not freqItems:
    print("Error: freqItems is empty")
    return closedItemsets
if not freqSet:
    print("Error: freqSet is empty")
    return closedItemsets
```

3. Screenshot of the execution time & memory usage for Task1 & Task2. The ratio of computation time of task1 and task2 were included

min support I tested:

datasetA - 0.3 / 0.1 / 0.05 / 0.01

datasetB – 0.2 / 0.1 / 0.05 / 0.01

datasetC – 0.5 / 0.1 / 0.05 / 0.01

- datasetA – min support 0.3

The ratio of computation time compared to that of Task 1 = 0 / 1.12 = 0%

```
(base) Annie@Annies-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.3 -p '/Users/Annie/Desktop/Data_Mining_202
4/Lab HW_1'
Running Task 1 with min_support = 0.3
Execution time for Task 1 (min_support 0.3): 1.12 seconds
Memory Usage for Task 1: 0.86 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_tas
k1_datasetA_0.3_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_
task1_datasetA_0.3_stats.txt
Running Task 2 with min_support = 0.3
Execution time for Task 2 (min_support 0.3): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2
_datasetA_0.3_closed_result.txt
Memory Usage for task1: 0.859375 MiB
```

- datasetA – min support 0.1

The ratio of computation time compared to that of Task 1 = 0 / 1.08 = 0%

```
(base) Annie@Annies-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.1 -p '/Users/Annie/Desktop/Data_Mining_202
4/Lab HW_1'
Running Task 1 with min_support = 0.1
Execution time for Task 1 (min_support 0.1): 1.08 seconds
Memory Usage for Task 1: 0.84 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_tas
k1_datasetA_0.1_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_
task1_datasetA_0.1_stats.txt
Running Task 2 with min_support = 0.1
Execution time for Task 2 (min_support 0.1): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2
_datasetA_0.1_closed_result.txt
Memory Usage for task1: 0.84375 MiB
```

- datasetA – min support 0.05

The ratio of computation time compared to that of Task 1 = 0 / 1.19 = 0%

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_20
24/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.05 -p '/Users/Annie/Desktop/Data_Mining_2024/L
ab HW_1'
Running Task 1 with min_support = 0.05
Execution time for Task 1 (min_support 0.05): 1.19 seconds
Memory Usage for Task 1: 1.12 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1_dat
asetA_0.05_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1_
datasetA_0.05_stats.txt
Running Task 2 with min_support = 0.05
Execution time for Task 2 (min_support 0.05): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2_datas
etA_0.05_closed_result.txt
Memory Usage for task1: 1.125 MiB
```

- datasetA – min support 0.01

The ratio of computation time compared to that of Task 1 = 0.06 / 4.80 = 1.25%

```
(base) Annie@Annies-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.01 -p '/Users/Annie/Desktop/Data_Mining_20
24/Lab HW_1'
Running Task 1 with min_support = 0.01
Execution time for Task 1 (min_support 0.01): 4.80 seconds
Memory Usage for Task 1: 37.53 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_tas
k1_datasetA_0.01_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_
task1_datasetA_0.01_stats.txt
Running Task 2 with min_support = 0.01
Execution time for Task 2 (min_support 0.01): 0.06 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2
_datasetA_0.01_closed_result.txt
Memory Usage for task1: 37.53125 MiB
```

- datasetB – min support 0.2

  The ratio of computation time compared to that of Task 1 = 0 / 7.56 = 0%

```
(base) Annie@Annies-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.2 -p '/Users/Annie/Desktop/Data_Mining_202
4/Lab HW_1'
Running Task 1 with min_support = 0.2
Execution time for Task 1 (min_support 0.2): 7.56 seconds
Memory Usage for Task 1: 142.30 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_tas
k1_datasetB_0.2_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_
task1_datasetB_0.2_stats.txt
Running Task 2 with min_support = 0.2
Execution time for Task 2 (min_support 0.2): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2
_datasetB_0.2_closed_result.txt
Memory Usage for task1: 142.296875 MiB
```

- datasetB – min support 0.1

  The ratio of computation time compared to that of Task 1 = 0 / 1392.47 = 0%

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab
HW_1/IBMGenerator-master/datasetB.data' -s 0.1 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1'
Running Task 1 with min_support = 0.1
Before filtering: 100001 candidates
After filtering: 10 frequent itemsets retained
Before filtering: 45 candidates
After filtering: 0 frequent itemsets retained
Execution time for Task 1 (min_support 0.1): 1392.47 seconds
Memory Usage for Task 1: 184.33 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1_datasetB_0
.1_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1_dataset
B_0.1_stats.txt
Running Task 2 with min_support = 0.1
Execution time for Task 2 (min_support 0.1): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2_datasetB_0.1
_closed_result.txt
Memory Usage for task1: 184.33203125 MiB
```

- datasetB – min support 0.05
- **The ratio of computation time compared to that of Task 1 = 0 / 1447.61 = 0%**

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2
024/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.05 -p '/Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1'
Running Task 1 with min_support = 0.05
Before filtering: 100001 candidates
After filtering: 56 frequent itemsets retained
Before filtering: 1540 candidates
After filtering: 0 frequent itemsets retained
Execution time for Task 1 (min_support 0.05): 1447.61 seconds
Memory Usage for Task 1: 187.07 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1_da
tasetB_0.05_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1
_datasetB_0.05_stats.txt
Running Task 2 with min_support = 0.05
Execution time for Task 2 (min_support 0.05): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2_data
setB_0.05_closed_result.txt
Memory Usage for task1: 187.06640625 MiB
```

- datasetB – min support 0.01

  The ratio of computation time compared to that of Task 1 = 0 / 2440.18 = 0%

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mini
ng_2024/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.01 -p '/Users/Annie/Desktop/Data_Min
ing_2024/Lab HW_1'
Running Task 1 with min_support = 0.01
Processing itemsets for minSupport: 100%|██████████| 100001/100001 [24:02<00:00, 69.32it/s]
Processing itemsets for minSupport: 100%|██████████| 51360/51360 [12:55<00:00, 66.22it/s]
Processing itemsets for minSupport: 100%|██████████| 1286/1286 [00:19<00:00, 64.47it/s]
Execution time for Task 1 (min_support 0.01): 2240.18 seconds
Memory Usage for Task 1: 202.84 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task
1_datasetB_0.01_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_t
ask1_datasetB_0.01_stats.txt
Running Task 2 with min_support = 0.01
Execution time for Task 2 (min_support 0.01): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2_
datasetB_0.01_closed_result.txt
Memory Usage for task1: 202.84375 MiB
```

- datasetC – min support 0.5

  The ratio of computation time compared to that of Task 1 = 0 / 39.38 = 0%

```
(base) Annie@Annies-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1/IBMGenerator-master/datasetC.data' -s 0.5 -p '/Users/Annie/Desktop/Data_Mining_202
4/Lab HW_1'
Running Task 1 with min_support = 0.5
Execution time for Task 1 (min_support 0.5): 39.38 seconds
Memory Usage for Task 1: 694.65 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_tas
k1_datasetC_0.5_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_
task1_datasetC_0.5_stats.txt
Running Task 2 with min_support = 0.5
Execution time for Task 2 (min_support 0.5): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2
_datasetC_0.5_closed_result.txt
Memory Usage for task1: 694.6484375 MiB
```

- datasetC – min support 0.1

  The ratio of computation time compared to that of Task 1 = 0 / 38.86 = 0%

```
(base) Annie@Annies-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1/IBMGenerator-master/datasetC.data' -s 0.1 -p '/Users/Annie/Desktop/Data_Mining_202
4/Lab HW_1'
Running Task 1 with min_support = 0.1
Execution time for Task 1 (min_support 0.1): 38.86 seconds
Memory Usage for Task 1: 691.56 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_tas
k1_datasetC_0.1_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_
task1_datasetC_0.1_stats.txt
Running Task 2 with min_support = 0.1
Execution time for Task 2 (min_support 0.1): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2
_datasetC_0.1_closed_result.txt
Memory Usage for task1: 691.5625 MiB
```

- datasetC – min support 0.05

  The ratio of computation time compared to that of Task 1 = 0 / 41218.76 = 0%

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step2.py -f '/Users/Annie/Desktop/Data_Mining_2024/
Lab HW_1/IBMGenerator-master/datasetC.data' -s 0.05 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_
1'
Running Task 1 with min_support = 0.05
Processing itemsets for minSupport: 100%| | 500001/500001 [11:24:39<00:00, 12.17it/s]
Processing itemsets for minSupport: 100%| | 1485/1485 [02:10<00:00, 11.36it/s]
Execution time for Task 1 (min_support 0.05): 41218.76 seconds
Memory Usage for Task 1: 933.69 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1_datase
tC_0.05_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task1_dat
asetC_0.05_stats.txt
Running Task 2 with min_support = 0.05
Execution time for Task 2 (min_support 0.05): 0.00 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step2_task2_datasetC
_0.05_closed_result.txt
Memory Usage for task1: 933.6875 MiB
```

- datasetC – min support 0.01

  The ratio of computation time compared to that of Task 1 = 0.01 / 55095.22 = 0%

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step2.py -f '/Users/A
nnie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetC.dat
a' -s 0.01 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1'
Running Task 1 with min_support = 0.01
Processing itemsets for minSupport: 100%| | 500001/500001 [14:13:11<00:
Processing itemsets for minSupport: 100%| | 51360/51360 [1:03:14<00:00,
Processing itemsets for minSupport: 100%| | 1251/1251 [01:40<00:00, 12.
Execution time for Task 1 (min_support 0.01): 55095.22 seconds
Memory Usage for Task 1: 914.23 MiB
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_20
24/Lab HW_1/step2_task1_datasetC_0.01_result.txt
Frequent itemset statistics written to /Users/Annie/Desktop/Data_Mining
_2024/Lab HW_1/step2_task1_datasetC_0.01_stats.txt
Running Task 2 with min_support = 0.01
Execution time for Task 2 (min_support 0.01): 0.01 seconds
Closed itemset results written to /Users/Annie/Desktop/Data_Mining_2024
/Lab HW_1/step2_task2_datasetC_0.01_closed_result.txt
Memory Usage for task1: 914.234375 MiB
```

4. The restrictions of StepII

1. This algorithm can be slow for **large datasets** due to its combinatorial nature.
   - For datasetC, the program was executing about 11 hrs.

5. Problems encountered in mining

1. It seems that def findClosedItemsets(freqItems, freqSet): return the same output as frequent itemsets > revised code as below, but the output still the same:
   - Store frequent itemsets by length in itemsets_by_length
   - Reduce the number of comparisons when checking for closure, as it only needs to check itemsets that are one length longer than the current itemset.
   - See Figure

```
def findClosedItemsets(freqItems, freqSet):
    closedItemsets = []
    # itemsets_by_length
    itemsets_by_length = defaultdict(list)
    for item, support in freqItems:
        itemsets_by_length[len(item)].append((item, support))

    for item, support in freqItems:
        isClosed = True
        # Check if there is a superset with the same support
        larger_len = len(item) + 1
        if larger_len in itemsets_by_length:
            for otherItem, otherSupport in itemsets_by_length[larger_len]:
                if item.issubset(otherItem) and support == otherSupport:
                    isClosed = False
                    break

        if isClosed:
            closedItemsets.append((item, support))
```

6. Any observations/discoveries

1. Results from def findClosedItemsets(freqItems, freqSet): is totally the same as result file1. My assumption is that it

2. When the number of transactions far exceeds the number of items (e.g., in Dataset C, where there are 500,000 transactions and 500 items), the probability of frequent itemsets occurring becomes sparse as the number of transactions increases. If the minimum support is set too high, it may overly restrict the frequent itemsets, resulting in no itemsets being mined that meet the conditions. In datasetB and C, as the number of transactions increases, minimum supports of 0.2 and 0.5 become more stringent, making the

occurrence of frequent itemsets almost impossible.

3. To see what exactly are the support values in the transactions, I counted in returnItemsWithMinSupport function, the log was shown as below. So reduce the min support may be better for mining.

```
Item: frozenset({'149'}), Count: 6, Support: 0.006
Item: frozenset({'678'}), Count: 1, Support: 0.001
Item: frozenset({'548'}), Count: 5, Support: 0.005
Item: frozenset({'813'}), Count: 1, Support: 0.001
Item: frozenset({'355'}), Count: 8, Support: 0.008
Item: frozenset({'656'}), Count: 1, Support: 0.001
Item: frozenset({'288'}), Count: 42, Support: 0.042
Item: frozenset({'423'}), Count: 15, Support: 0.015
Item: frozenset({'245'}), Count: 30, Support: 0.03
Item: frozenset({'350'}), Count: 9, Support: 0.009
```

Section 2 – Step III

1. Description of this FP growth algorithm

a. The algorithm used here: **FP Growth**

Source code: https://github.com/evandempsey/fp-growth/blob/master/pyfpgrowth/pyfpgrowth.py

b. **Program flow:**

(1) Command-Line Argument Parsing:

- The program parses command-line arguments to get the input file path, minimum support value, output path, and maximum tree depth.

(2) Run and Monitor:

- The `run_and_monitor` function is called.
- Inside `run_and_monitor`, the `run_fpgrowth` function is defined and called.
- `run_fpgrowth` calls `generate_patterns_rules` to generate patterns and rules.

(3) Generate Patterns and Rules:

- `generate_patterns_rules` calls `open_data` to read transactions from the input file.
- Then calls `find_frequent_patterns` to create an FP-Tree and mine patterns.

(4) Find Frequent Patterns:

- `find_frequent_patterns` creates an instance of FPTree.
- The FPTree constructor (`__init__`) initializes the tree, finds frequent items, builds the header table, constructs the FP-Tree, and calculates metadata.
- Then calls `mine_patterns` to mine frequent patterns from the tree.

(5) Mine Patterns:

- mine_patterns checks if the tree has a single path.
- If it does, it calls generate_pattern_list.
- If it doesn't, it calls mine_sub_trees to mine patterns from sub-trees.

(6) Write Results and Statistics:

- The main block calculates the total execution time.
- It prints the total execution time and memory usage.

2. How to run the code (StepIII)

a. Turn on Terminal and input: (direct to my .py folder)

```
cd "/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1"
```

b. Take turns run datasetA, datasetB and datasetC. Put the command line in Terminal For datasetA:

```
python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.01 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
```

- I set the max depth as 20

3. The modifications & improvement I made for Step III

1. **Changed OptionParser to argparse** and **memory_usage** to measure efficiency

```
import time
import itertools
import argparse
from memory_profiler import memory_usage
```

2. **Add __slots__ and def remove_references(self) to improve efficiency:**

- Originally, the memory usage didn't perform well, then I added slots to improve the usage.
- When the mining of a subtree is completed and the structure of that subtree is no longer needed, these unnecessary data (parent, link) can be released.
- The improvement of time and memory usage will be shown in the later section.

```python
class FPNode(object):
    __slots__ = ['value', 'count', 'parent', 'link', 'children']
    def __init__(self, value, count, parent):
        self.value = value
        self.count = count
        self.parent = parent
        self.link = None
        self.children = []

    def has_child(self, value):
        for node in self.children:
            if node.value == value:
                return True
        return False

    def get_child(self, value):
        for node in self.children:
            if node.value == value:
                return node
        return None

    def add_child(self, value):
        child = FPNode(value, 1, self)
        self.children.append(child)
        return child

    def remove_references(self):
        """Remove references to parent and link to free memory."""
        self.parent = None
        self.link = None
```

3. **Add tree construction time, tree depth, node count into FP tree:**

```python
class FPTree(object):
    __slots__ = ['transactions', 'frequent', 'headers', 'root', 'tree_construction_time', 'tree_depth', 'node_count']

    def __init__(self, transactions, min_support, root_value=None, root_count=None, max_depth=None):
        start_time = time.time()
        transaction_count = len(transactions)
        self.frequent = self.find_frequent_items(transactions, min_support, transaction_count)
        self.headers = self.build_header_table(self.frequent)
        self.root = self.build_fptree(transactions, root_value, root_count, self.frequent, self.headers, max_depth)
        self.tree_construction_time = time.time() - start_time
        self.tree_depth = self.get_tree_depth(self.root)
        self.node_count = self.get_node_count(self.root)
        self.remove_unused_references(self.root)
```

4. **Revised to return {item: count for item, count in items.items() if count / transaction_count >= min_support}**

   - Directly generate a dictionary so that it won't repeatedly deletes non-matching keys -> improve efficiency !

```python
def find_frequent_items(self, transactions, min_support, transaction_count):
    items = {}
    for transaction in transactions:
        for item in transaction:
            if item in items:
                items[item] += 1
            else:
                items[item] = 1
    return {item: count for item, count in items.items() if count / transaction_count >= min_support}
```

5. **Add max_depth parameter to control the depth of tree**:

```python
def insert_tree(self, items, node, headers, max_depth, depth=0):
    if max_depth is not None and depth >= max_depth:
        return
    if items:
        first, *rest = items
        child = node.get_child(first)
        if child:
            child.count += 1
        else:
            child = node.add_child(first)
            if headers[first] is None:
                headers[first] = child
            else:
                current = headers[first]
                while current.link:
                    current = current.link
                current.link = child
        self.insert_tree(rest, child, headers, max_depth, depth + 1)
```

6. **Add `seen_patterns = set()` and for pattern in `subtree_patterns.keys()` in def `mine_sub_trees()` to avoid recalculating patterns that have already been processed** -> improve efficiency !

```python
def mine_sub_trees(self, min_support):
    patterns = {}
    seen_patterns = set()  # 記錄已經計算的模式
    mining_order = sorted(self.frequent.keys(), key=lambda x: self.frequent[x])

    for item in mining_order:
        suffixes = []
        conditional_tree_input = []
        node = self.headers[item]

        while node is not None:
            suffixes.append(node)
            node = node.link

        for suffix in suffixes:
            frequency = suffix.count
            path = []
            parent = suffix.parent

            while parent is not None and parent.parent is not None:
                path.append(parent.value)
                parent = parent.parent

            for i in range(frequency):
                conditional_tree_input.append(path)

        subtree = FPTree(conditional_tree_input, min_support, item, self.frequent[item])
        subtree_patterns = subtree.mine_patterns(min_support)

        for pattern in subtree_patterns.keys():
            if pattern in seen_patterns:
                continue  # Skip already calculated patterns
            seen_patterns.add(pattern)

            if pattern in patterns:
                patterns[pattern] += subtree_patterns[pattern]
            else:
                patterns[pattern] = subtree_patterns[pattern]
    return patterns
```

7. **Add `run_and_monitor` to package FP growth and record memory usage**

```
def run_and_monitor():
    def run_fpgrowth():
        patterns, tree, transaction_count = generate_patterns_rules(input_file, min_support, max_depth)
        total_itemsets = len(patterns)
        write_result_file(patterns, output_path, dataset_name, min_support, transaction_count)
        return patterns, tree, total_itemsets

    mem_usage, (patterns, tree, total_itemsets) = memory_usage(
        (run_fpgrowth, ()), retval=True
    )

    return patterns, tree, total_itemsets, mem_usage
```

8. **Use if __name__ == "__main__" to run the script**

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run FP-Growth Algorithm")
    parser.add_argument("-f", "--inputFile", type=str, required=True, help="Dataset file path")
    parser.add_argument("-s", "--minSupport", type=float, required=True, help="Minimum support value")
    parser.add_argument("-p", "--outputPath", type=str, required=True, help="Output file path")
    parser.add_argument("-d", "--maxDepth", type=int, required=False, help="Maximum tree depth", default=None)

    args = parser.parse_args()
    input_file = args.inputFile
    min_support = args.minSupport
    output_path = args.outputPath
    max_depth = args.maxDepth
    dataset_name = input_file.split("/")[-1].split(".")[0]

    start_time = time.time()
    patterns, tree, total_itemsets, mem_usage = run_and_monitor()

    total_execution_time = time.time() - start_time

    write_statistics_file(
        tree,
        total_itemsets,
        output_path,
        total_execution_time,
        dataset_name,
        min_support,
        mem_usage,
    )
```

9. **Set the max-depth of FP tree as 20:** Because it took so long to run the fp growth without limiting the max-depth.

4. Computation time compared to Step II (task1)

a. datasetA – min support 0.3

   [(1.12 – 1.42) / 1.12]*100 = -26.7%

b. datasetA – min support 0.1

   [(1.08 – 1.43) / 1.08]*100 = -32.4%

c. datasetA – min support 0.05

   [(1.19 - 1.46) / 1.19]*100 = -22.6%

d. datasetA – min support 0.01

   [(4.80 – 1.07) / 4.80]*100 = 91.4%

e. datasetB – min support 0.2

   [(7.56 - 1.11) / 7.56]*100 = 85.3%

f. datasetB – min support 0.1

   [(1392.47 – 1.32) / 1392.47]*100 = 99.99%

g. datasetB – min support 0.05

   [(1447.61 - 18.45) / 1447.61]*100 = 98.72%

h. datasetB – min support 0.01

   [(2440.18 - 199.05) / 2440.18]*100 = 91.84%

i. datasetC – min support 0.5

   [(39.38 – 3.92) /]*100 = 90.04%

j. datasetC – min support 0.1

   [(38.86 – 4.38) / 38.86]*100 = 88.72%

k. datasetC – min support 0.05

   [(41218.76 – 523.58) / 41218.76]*100 = 99.21%

l. datasetC – min support 0.01

   [(55095.22 - 5159.88) / 55095.22]*100 =

Screenshot of the execution time & memory usage for Step3 (only Task1)

- datasetA – min support 0.3

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_20
24/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.3 -p '/Users/Annie/Desktop/Data_Mining_2024/La
b HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.3_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.3_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.3_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetA_0.3_stat
s.txt
Total execution time: 1.43 seconds
Memory Usage: 0.3671875 MiB
```

- datasetA – min support 0.1

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_20
24/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.1 -p '/Users/Annie/Desktop/Data_Mining_2024/La
b HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.1_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.1_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.1_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetA_0.1_stat
s.txt
Total execution time: 1.44 seconds
Memory Usage: 0.6875 MiB
```

Because it did not seem work a lot efficiently than StepII, then I added **__slots__** and **def remove_references(self)**, it accordingly improved a little:

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/De
sktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.1 -p '
/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab
HW_1/step3_task1_datasetA_0.1_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab
HW_1/step3_task1_datasetA_0.1_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab
HW_1/step3_task1_datasetA_0.1_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_tas
k1_datasetA_0.1_stats.txt
Total execution time: 1.43 seconds
Memory Usage: 0.40234375 MiB
```

- datasetA – min support 0.05

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_20
24/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.05 -p '/Users/Annie/Desktop/Data_Mining_2024/L
ab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.05_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.05_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetA_0.05_sta
ts.txt
Total execution time: 1.09 seconds
Memory Usage: 1.1484375 MiB
```

Because it did not seem work a lot efficiently than StepII, then I added **__slots__** and **def remove_references(self)**, it accordingly improved memory usage a lot but time didn't :

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/De
sktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.05 -p
'/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab
HW_1/step3_task1_datasetA_0.05_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab
HW_1/step3_task1_datasetA_0.05_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab
HW_1/step3_task1_datasetA_0.05_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_tas
k1_datasetA_0.05_stats.txt
Total execution time: 1.46 seconds
Memory Usage: 0.46484375 MiB
```

- datasetA – min support 0.01

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_20
24/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.01 -p '/Users/Annie/Desktop/Data_Mining_2024/L
ab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_dat
asetA_0.01_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetA_0.01_sta
ts.txt
Total execution time: 144.95 seconds
Memory Usage: 2020.02734375 MiB
```

The memory usage was originally too much, after adding **__slots__** and **def remove_references(self)**, it improved both memory usage and time a lot:

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetA.data' -s 0.01 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetA_0.01_results.txt
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetA_0.01_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetA_0.01_stats.txt
Total execution time: 1.07 seconds
Memory Usage: 2.64453125 MiB
```

- datasetB – min support 0.2

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.2 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.2_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.2_stats.txt
Total execution time: 1.11 seconds
Memory Usage: 80.52734375 MiB
```

- datasetB – min support 0.1

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.1 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.1_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.1_stats.txt
Total execution time: 1.32 seconds
Memory Usage: 110.71484375 MiB
```

- datasetB – min support 0.05

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.05 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.05_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.05_stats.txt
Total execution time: 45.83 seconds
Memory Usage: 355.15625 MiB
```

After adding **__slots__** and **def remove_references(self)**, it improved both memory usage and time a lot:

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.05 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.05_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.05_stats.txt
Total execution time: 18.45 seconds
Memory Usage: 124.8828125 MiB
```

- datasetB – min support 0.01

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetB.data' -s 0.01 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.01_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetB_0.01_stats.txt
Total execution time: 199.05 seconds
Memory Usage: 211.984375 MiB
```

- datasetC – min support 0.5

```
(base) Annie@Annies-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetC.data' -s 0.5 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1'
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.5_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.5_stats.txt
Total execution time: 3.92 seconds
Memory Usage: 382.01953125 MiB
```

- datasetC – min support 0.1

```
(base) Annie@Annies-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetC.data' -s 0.1 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1'
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.1_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.1_stats.txt
Total execution time: 4.38 seconds
Memory Usage: 385.36328125 MiB
```

- datasetC – min support 0.05

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetC.data' -s 0.05 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.05_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.05_stats.txt
Total execution time: 523.58 seconds
Memory Usage: 540.04296875 MiB
```

- datasetC – min support 0.01

```
(base) Annie@Annies-MacBook-Air Lab HW_1 % python Step3.py -f '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/IBMGenerator-master/datasetC.data' -s 0.01 -p '/Users/Annie/Desktop/Data_Mining_2024/Lab HW_1' -d 20
Frequent itemset results written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.01_results.txt
Statistics written to /Users/Annie/Desktop/Data_Mining_2024/Lab HW_1/step3_task1_datasetC_0.01_stats.txt
Total execution time: 5159.88 seconds
Memory Usage: 951.73828125 MiB
```

5. The scalability of my FP growth in terms of datasize

1. **For Dataset A**: For smaller datasets (like A) with higher minimum support(0.3, 0.1, 0.05), the additional steps in FP-tree construction might add overhead, making it slightly slower. However, with lower support, FP-Growth's structure

becomes beneficial.

2. **For Dataset B:** As the minimum support decreases from 0.2 to 0.01, FP growth shows a big increase in time savings, ranging from 85.3% to 99.99% faster. This shows the effectiveness of the FP-Growth algorithm for mid-sized datasets, especially as the minimum support threshold lowers. As a result, FP growth scales well for larger datasets and increasingly low support values, making it ideal for tasks requiring comprehensive pattern mining in mid-sized datasets.
3. **For Dataset C:** For the largest datasetC, FP growth shows consistent time savings at various support levels, ranging from around 88.72% (0.1 support) to an impressive 99.21% at 0.05 support. These results show that FP growth scales efficiently with dataset size, handling large transaction counts effectively, even with lower support thresholds. This trend confirms that the FP-Growth can manage expansive data sizes while maintaining computational efficiency.
4. In summary, FP-Growth in my setp3 improves with both increasing dataset size and decreasing minimum support levels. As datasets become larger, FP-Growth's tree-based pattern mining outpaces the candidate generation method in Apriori, handling exponentially growing pattern counts without excessive computation.

Step IV. Verify the correctness of the output format and mining results

python ItemsetVerifier.py -r T10I6001Ms1_freq_itemsets.txt -s step3_task1_datasetC_0.01_results.txt