DS221: Introduction to Scalable Systems – DSA Assignment

Aneesh Panchal
06-18-01-10-12-24-1-25223
Indian Institute of Science (IISc), Bangalore, IN

SEP 2024

1 Question 1

Function Signature –
void groupCustomersByHashtags(fileIterator& hashtags, fileIterator& purchases, fileIterator& prices, int k, string outputFilePath)

1.1 Time Complexity Analysis

1.1.1 Worst Case

Assume c represents the number of customers present, p represents the number of products and h represents maximum number of hashtags. k is the given parameter.

Reading and Populating the Data,
Product to Hashtags Mapping – $O(ph)$
Customer to Hashtags Mapping – $O(cph)$

Finding Clusters,
Groups to Customers Mapping – $O(c(h+k)\log(k))$

Writing to file,
Making vector for writing .csv – $O(c)$

Hence, final Asymptotic Time Complexity is given by,

$$O(ph + cph + c(h+k)\log(k) + c) \quad \text{which is equivalent to,} \quad \mathbf{O(cph + ch\log(k))} \tag{1}$$

1.1.2 Best Case

Assume c represents the number of customers present, p represents the number of products and $h = 1$ as each product is associated with only 1 hashtag represents the best case. Hence, $k \leq 1$ which implies $k = 1$ is the given parameter.

Reading and Populating the Data,
Product to Hashtags Mapping – $\Omega(p)$
Customer to Hashtags Mapping – $\Omega(c)$

Finding Clusters,
Groups to Customers Mapping – $\Omega(c)$

Writing to file,
Making vector for writing .csv – $\Omega(c)$

Hence, final Asymptotic Time Complexity is given by,

$$\Omega(p + c + c + c) \quad \text{which is equivalent to,} \quad \mathbf{\Omega(\max(c, p))} \tag{2}$$

1.2 Space Complexity Analysis

1.2.1 Worst Case

Considering the complexity inside the function only,
Product to Hashtags Mapping - $O(ph)$
Customer to Hashtags Mapping - $O(ch)$
Making vector for writing .csv - $O(c)$

Hence, final Space Complexity is given by,

$$\mathbf{O(ph + ch + c)} \tag{3}$$

1.2.2 Best Case

Best Case occurs when each customer is mapped to only 1 product and each product is associated with only 1 hashtag. Hence, considering the complexity inside the function only,
Product to Hashtags Mapping - $\Omega(p)$
Customer to Hashtags Mapping - $\Omega(\min(c, p))$
Making vector for writing .csv - $\Omega(c)$

Hence, final Space Complexity is given by,

$$\Omega(p + \min(c, p) + c) \quad \text{which is equivalent to,} \quad \mathbf{\Omega}(\max(\mathbf{c}, \mathbf{p})) \tag{4}$$

1.3 Experimental Setup

1.3.1 Worst Case

Each Customer buys all the products.
Each Product is associated with all the tags.

1. **Number of Customers:**
   500 products are considered with each having all the 50 tags with value of k as 2.
2. **Number of Products:**
   250 customers are considered with each customer having all the products and each product having same 50 tags with value of k as 2.
3. **Number of Tags:**
   250 customers and 500 products with tag values chosen from the set of number of tags. All the tags is considered for each product and each customer purchases all the product.
4. **K values:**
   1000 products and 250 customers are considered for this case with each product having all the tags and each customer buys all the product.

1.3.2 Best Case

Each Costumer buys only single product.
Each Product is associated with single tag.

1. **Number of Customers:**
   500 products are considered with each having 1 tag only with value of k as 2.
2. **Number of Products:**
   250 customers are considered with each customer having 1 product and each product having 1 tag only with value of k as 2.
3. **Number of Tags:**
   250 customers and 500 products with tag values chosen from the set of number of tags. Only 1 tag is considered for each product and each customer purchases only 1 product.
4. **K values:**
   10000 products and 250 customers are considered for this case with each product having single tag and each customer buys only single product.

1.4 Graphical Analysis & Comparisons

Number of Customers: From Fig. 1 it is clear that in best case as well as the worst case, if we consider number of products (p), number of tags (h) and k value as constant then we get **linear** dependence of Number of customers with time, which can also be seen from equation (2) and (1),

$$\text{Best Case: } \Omega(\max(c, p)) \equiv \Omega(c)$$

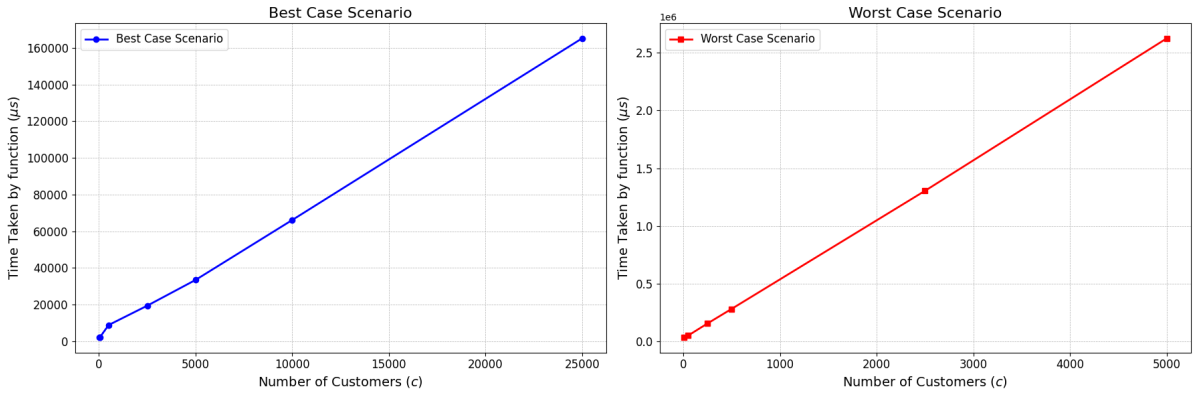

Figure 1: Number of Customers

$$\text{Worst Case: } O(cph + ch\log(k)) \equiv O(c)$$

Number of Products: From Fig. 2 it is clear that in best case as well as the worst case, if we consider number of customers (c), number of tags (h) and k value as constant then we get **linear** dependence of Number of products with time, which can also be seen from equation (2) and (1),

$$\text{Best Case: } \Omega(\max(c, p)) \equiv \Omega(p)$$

$$\text{Worst Case: } O(cph + ch\log(k)) \equiv O(p)$$

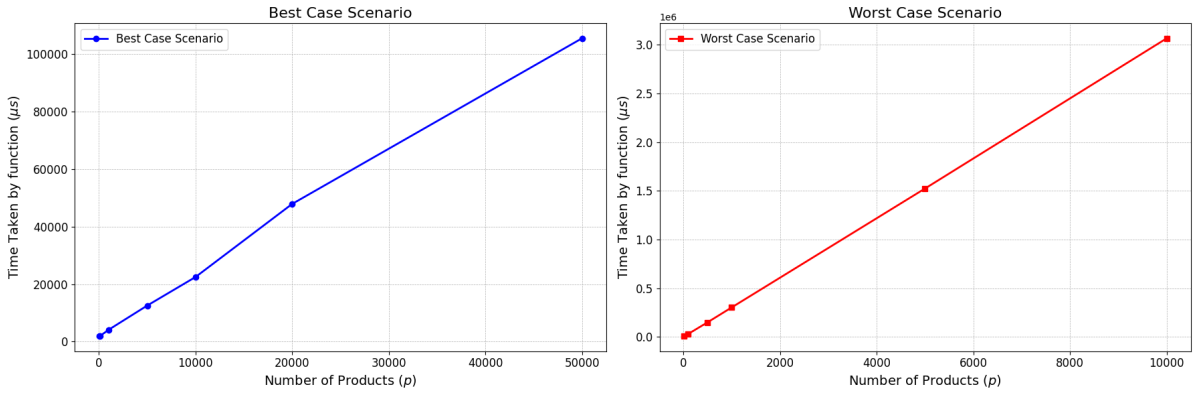

Figure 2: Number of Products

Number of Tags: From Fig. 3 it is clear that in best case time value oscillates around a small value and for the worst case **linear** dependence of Number of tags with time can be seen, if we consider number of products (p), number of customers (c) and k value as constant then we get the same results, which can also be seen from equation (2) and (1),

$$\text{Best Case: } \Omega(\max(c, p)) \equiv \Omega(1)$$

$$\text{Worst Case: } O(cph + ch\log(k)) \equiv O(h)$$

K Values: From Fig. 4 it is clear that in best case time value oscillates around a small value and for the worst case **logarithmic** dependence of k with time can be seen, if we consider number of products (p), number of customers (c) and Number of tags (h) as constant then we get the same results, which can also be seen from equation (2) and (1),

$$\text{Best Case: } \Omega(\max(c, p)) \equiv \Omega(1)$$

$$\text{Worst Case: } O(cph + ch\log(k)) \equiv O(\log(k))$$

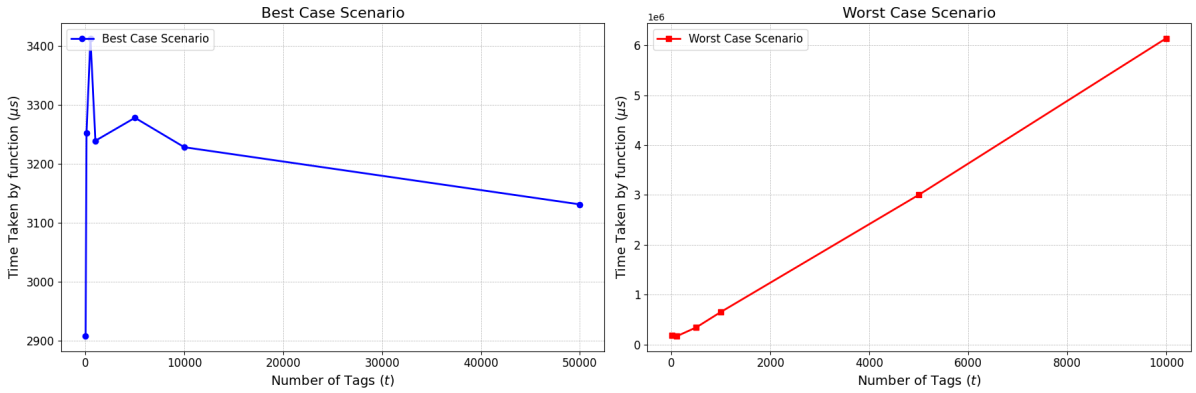

Figure 3: Number of Tags

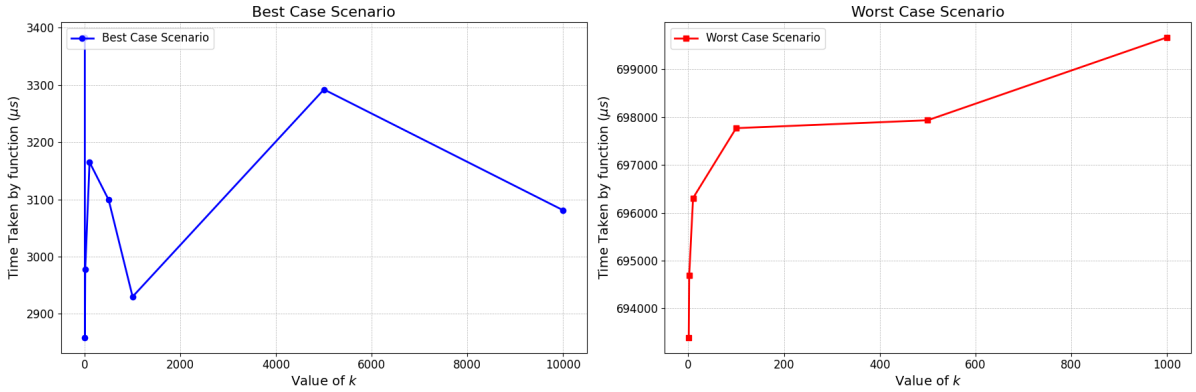

Figure 4: K Values

1.5 Memory Usage

From Fig. 5 we can see that our process when running through all the best cases takes almost negligible $MEM\%$ but CPU is utilized fully i.e. $CPU\%$ is around full, this is due to the reason that our process get priority over background events on the server.

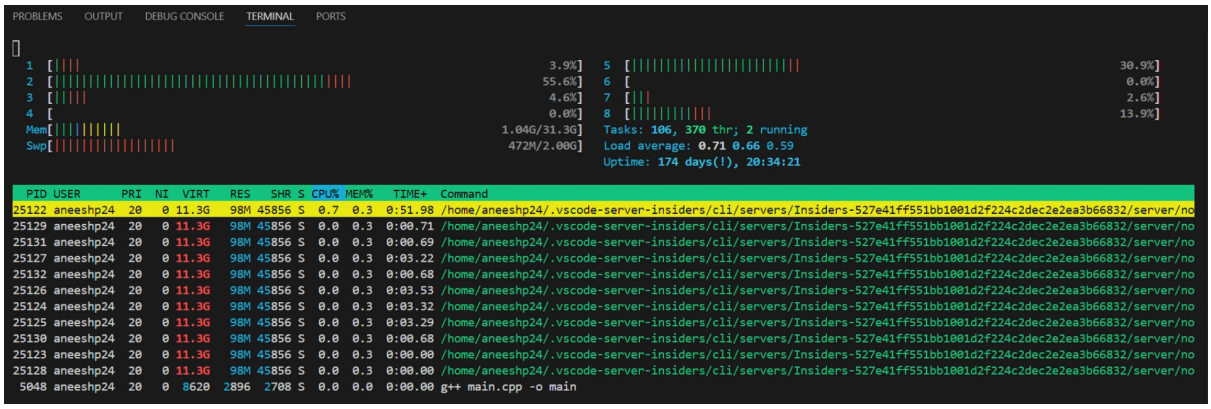

Figure 5: PID: 5048 [Best Case for Question 1]

From Fig. 6 we can see that our process when running through all the worst cases takes almost full $CPU\%$ and around 0.1% of total memory which is equivalent to,

$$0.1\% \text{ of } 31.3GB = 31.3 \times 1024 \times 0.001 = 32.05MB$$

Hence, $32.05MB$ is required for running our worst cases on the server.

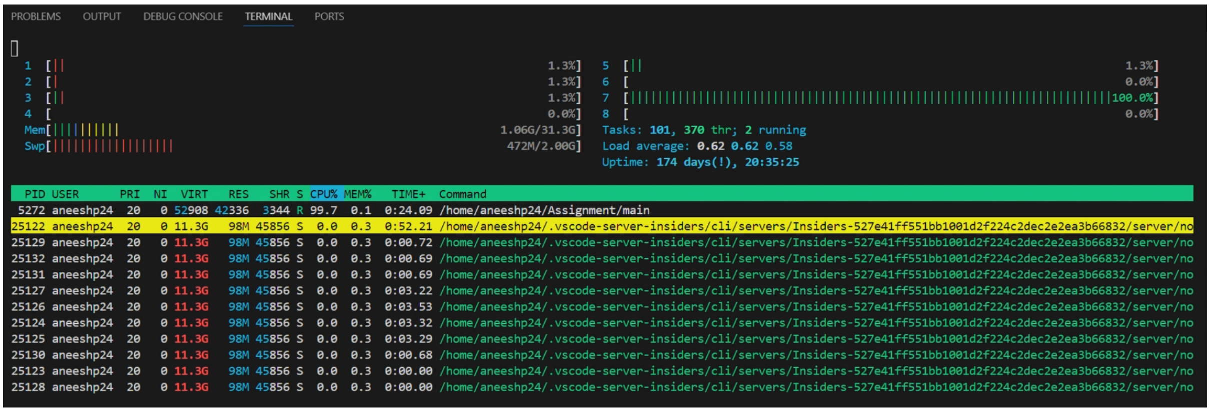

Figure 6: PID: 5272 [Worst Case for Question 1]

1.6 Edge Cases Considered

1. Customer to product mapping exist, but product not in input_product
   [**Passed** with no mention of product]

2. No hashtags for a product in input_product
   [**Passed** with no mention of product]

3. New customer with product not in input_product
   [**Passed** with no mention of customer is final output]

4. Multiple entries for same product ID in input_product
   [**Passed** with ignoring the latest ones and taking the old ones]

5. Multiple entries for same customer ID and same product ID
   [**Passed**]

6. Same multiple hashtags for same product ID
   [**Passed** using set to store unique tags.]

7. k $>$ number of tags or, k ≤ 0
   [**Passed**]

8. Include new hashtags for same product ID
   [**Passed** without using new entry]

Tools and References

1. GitHub, Inc., GitHub Copilot. Retrieved Sept 2024, from https://github.com/features/copilot

2. Van Rossum, G., & Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

3. Microsoft Corporation. (2018). Microsoft Excel. Retrieved from https://office.microsoft.com/excel

4. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms. MIT press, 2022.

5. Microsoft., Visual Studio Code. Retrieved Sept 2024, from https://code.visualstudio.com/

2 Question 3

Function Signature –
void groupCustomersByHashtagsDynamic(fileIterator& hashtags, fileIterator& purchases, fileIterator& prices, string newHastags, int k, string outputFilePath)

2.1 Assumptions

1. Hashtags are only added, not removed.
2. Purchases for customers do not change.

2.2 Change in Approach from Static to Dynamic

For the static case, we used only simple local variables for the creation of groups and other processing but due to multiple access of the same map in the dynamic case, we make the product to initial hashtags map as global variable which reduces time very significantly. As it is given in the assumption that hashtags are only added and not removed is the only reason why we can use this map as global variable.

2.3 Time Complexity Analysis

2.3.1 Worst Case

Assume c represents the number of customers present, p represents the number of products and h represents maximum number of hashtags in pre-existing file product hashtags, h_o represents the new hashtags and m represents number of products in batch to be processed ($m = p$). k is the given parameter.

Reading and Populating the Data,
Product to Hashtags Mapping $- O(p(h + h_o))$
Customer to Hashtags Mapping $- O(cp(h + h_o))$

Finding Clusters,
Groups to Customers Mapping $- O(c(h + h_o + k)\log(k))$

Writing to file,
Making vector for writing .csv $- O(c)$

Hence, final Asymptotic Time Complexity is given by,

$$O(p(h + h_o) + cp(h + h_o) + c(h + h_o + k)\log(k) + c)$$

which is equivalent to,

$$\mathbf{O(c(h + h_o)(p + \log(k)))} \tag{5}$$

2.3.2 Best Case

Assume c represents the number of customers present, p represents the number of products and $h = 1$ as each product is associated with only 1 hashtag represents the best case. Hence, $k \leq 1$ which implies $k = 1$ is the given parameter.

Reading and Populating the Data,
Product to Hashtags Mapping $- \Omega(p)$
Customer to Hashtags Mapping $- \Omega(c)$

Finding Clusters,
Groups to Customers Mapping $- \Omega(c)$

Writing to file,

Making vector for writing .csv $-\ \Omega(c)$

Hence, final Asymptotic Time Complexity is given by,

$$\Omega(p+c+c+c) \quad \text{which is equivalent to,} \quad \mathbf{\Omega}(\max(\mathbf{c},\mathbf{p})) \tag{6}$$

2.4 Space Complexity Analysis

2.4.1 Worst Case

Considering the complexity inside the function only,
Product to Hashtags Mapping - $o(p(h+h_o))$
Customer to Hashtags Mapping - $o(c(h+h_o))$
Making vector for writing .csv - $o(c)$

Global Space Complexity [remains constant over dynamic runs],

$$\mathbf{o(ph)} \tag{7}$$

Local Space Complexity,

$$\mathbf{o(ph_o + c(h+h_o) + c)} \tag{8}$$

2.4.2 Best Case

Best Case occurs when each customer is mapped to only 1 product and each product is associated with only 1 hashtag and new hashtags file is empty. Hence, considering the complexity inside the function only,
Product to Hashtags Mapping - $\Omega(p)$
Customer to Hashtags Mapping - $\Omega(\min(c,p))$
Making vector for writing .csv - $\Omega(c)$

Hence, final Space Complexity is given by,

$$\Omega(p+\min(c,p)+c) \quad \text{which is equivalent to,} \quad \mathbf{\Omega}(\max(\mathbf{c},\mathbf{p})) \tag{9}$$

2.5 Experimental Setup

As customer to product mapping is considered to be constant, hence the trend number of customers show is equivalent to the case in Question 1.

2.5.1 Worst Case

Each Customer buys all the products.
Each Product is associated with all the tags.

1. **Number of Products:**
   250 customers are considered with each customer having all the products and each product having same 50 tags with value of k as 2 and m as 5. New Hashtag list consists of all the products with 50 new hashtags
2. **Number of Tags:**
   250 customers and 500 products with tag values chosen from the set of number of tags. All the tags is considered for each product and each customer purchases all the product. Value of k is considered as 2 and m as 5. New Hashtag list consists of all the products with variable number of hashtags.
3. **K values:**
   100 products and 250 customers are considered for this case with each product having all the tags and each customer buys all the product. Along with this, New Hashtag list consisting of all the products with new 100 hashtags are considered with m value to be 5.
4. **M values:**
   100 products and 250 customers are considered for this case with each product having all the tags and each customer buys all the product. Along with this, New Hashtag list consisting of all the products with new 100 hashtags are considered with k value to be 2.

2.5.2 Best Case

Each Costumer buys only single product.
Each Product is associated with single tag.
New Hashtag list is empty, which makes it ***equivalent to Question 1***.

1. **Number of Products:**
   250 customers are considered with each customer having 1 product and each product having 1 tag only with value of k as 2 and value of m as 5.

2. **Number of Tags:**
   250 customers and 500 products with tag values chosen from the set of number of tags. Only 1 tag is considered for each product and each customer purchases only 1 product. Consider value of k as 2 and value of m as 5.

3. **K values:**
   10000 products and 250 customers are considered for this case with each product having single tag and each customer buys only single product. Consider value of m as 5.

4. **M values:**
   10000 products and 250 customers are considered for this case with each product having single tag and each customer buys only single product. Consider value of k as 2.

2.6 Graphical Analysis & Comparisons

Number of Products: From Fig. 7 it is clear that in best case as well as the worst case, if we consider number of customers (c), number of tags (h) and k value as constant then we get **linear** dependence of Number of products with time, which can also be seen from equation (6) and (5),

$$\text{Best Case: } \Omega(\max(c,p)) \equiv \Omega(p)$$

$$\text{Worst Case: } O(c(h+h_o)(p+\log(k))) \equiv O(p)$$

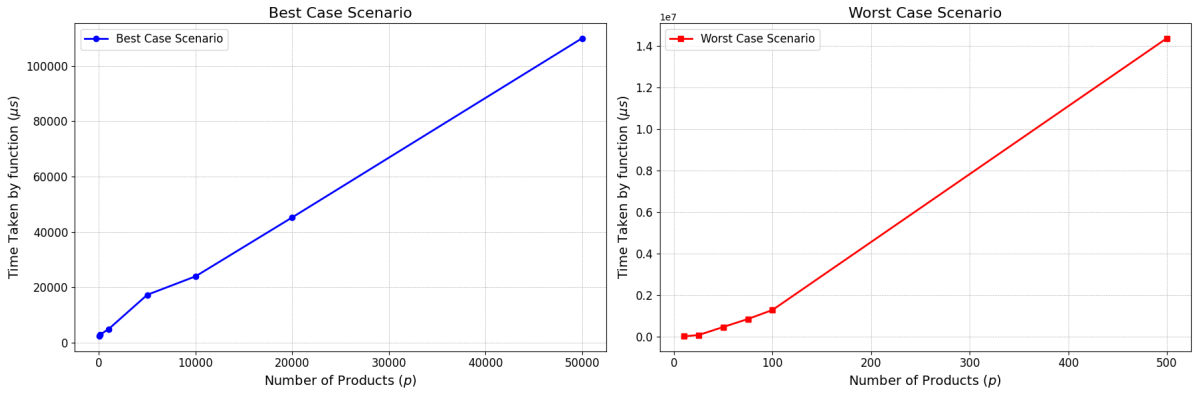

Figure 7: Number of Products

Number of Tags: From Fig. 8 it is clear that in best case time value oscillates around a small value and for the worst case **linear** dependence of Number of tags with time can be seen, if we consider number of products (p), number of customers (c) along with k and m values as constant then we get the same results, which can also be seen from equation (6) and (5),

$$\text{Best Case: } \Omega(\max(c,p)) \equiv \Omega(1)$$

$$\text{Worst Case: } O(c(h+h_o)(p+\log(k))) \equiv O(h+h_o)$$

K Values: From Fig. 9 it is clear that in best case time value oscillates around a small value and for the worst case **logarithmic** dependence of k with time can be seen, if we consider number of products (p), number of customers

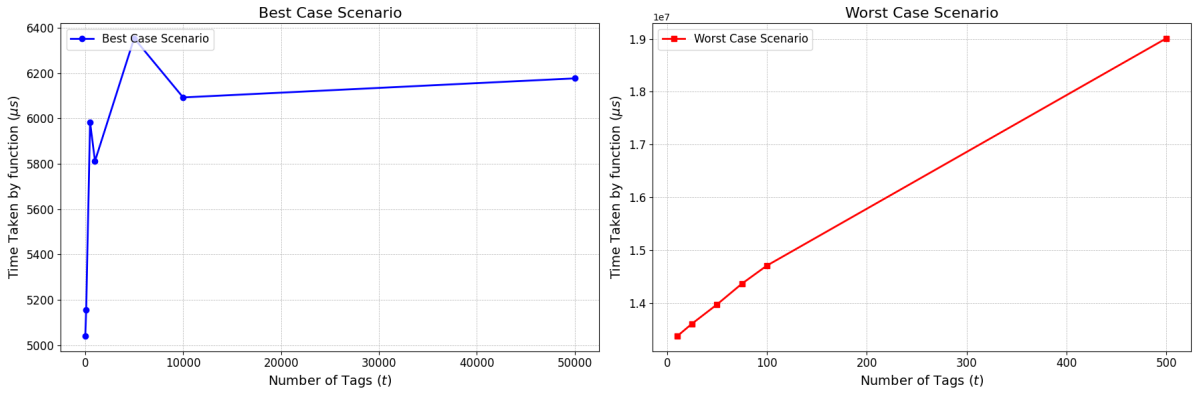

Figure 8: Number of Tags

(c) and Number of tags (h) along with value of m as constant then we get the same results, which can also be seen from equation (6) and (5),

$$\text{Best Case: } \Omega(\max(c, p)) \equiv \Omega(1)$$

$$\text{Worst Case: } O(c(h + h_o)(p + \log(k))) \equiv O(\log(k))$$

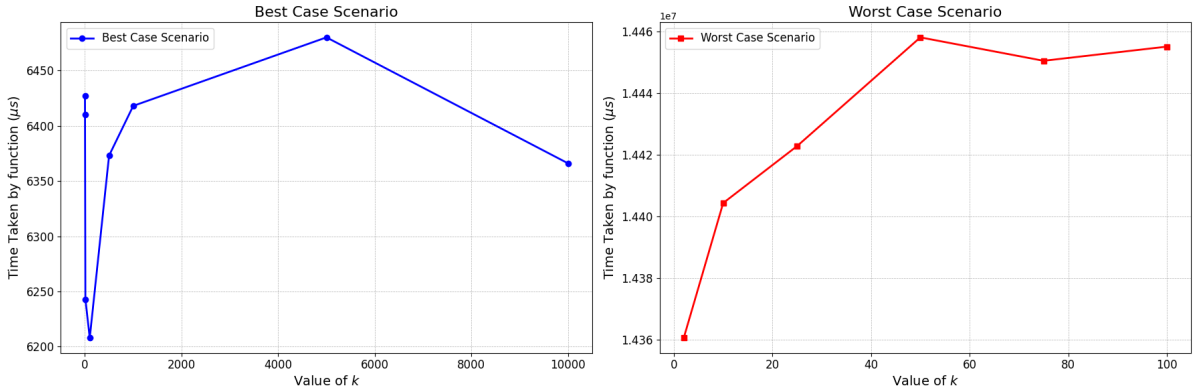

Figure 9: K Values

M Values: From Fig. 10 it is clear that in best case time value oscillates around a small value. However, for the worst case we can see that as the value of m increases the total time required get decreases because we can process more number of new hashtags in same iterations and hence, as the value of m increases the total time required gets decreases. But time taken for 1 run increases but it is very less as compare to the decrease we see. If we consider number of products (p), number of customers (c) and Number of tags (h) along with value of k as constant.

2.7 Memory Usage

From Fig. 11 we can see that our process when running through all the best cases takes almost negligible $MEM\%$ but CPU is utilized fully i.e. $CPU\%$ is around full, this is due to the reason that our process get priority over background events on the server.
From Fig. 12 we can see that our process when running through all the worst cases takes almost full $CPU\%$ and around 0.1% of total memory which is equivalent to,

$$0.1\% \text{ of } 31.3GB = 31.3 \times 1024 \times 0.001 = 32.05MB$$

Hence, $32.05MB$ is required for running our worst cases on the server.

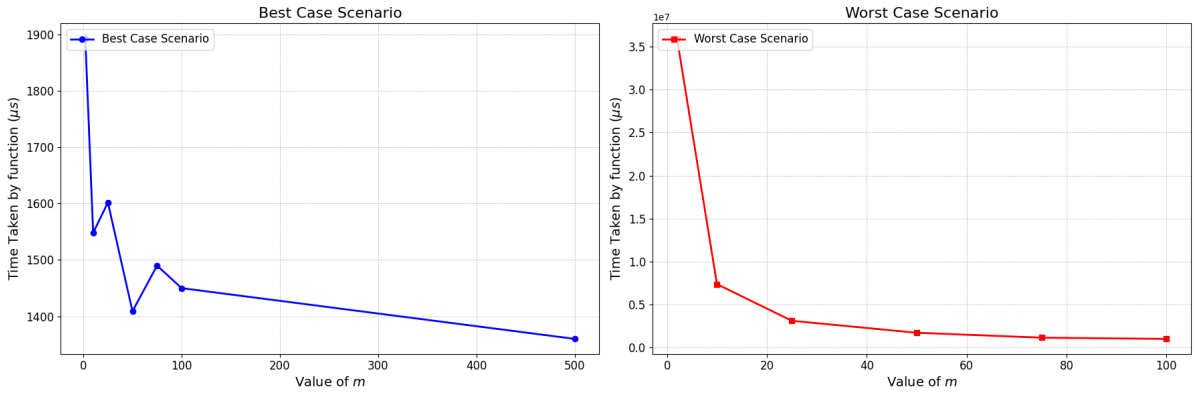

Figure 10: M Values

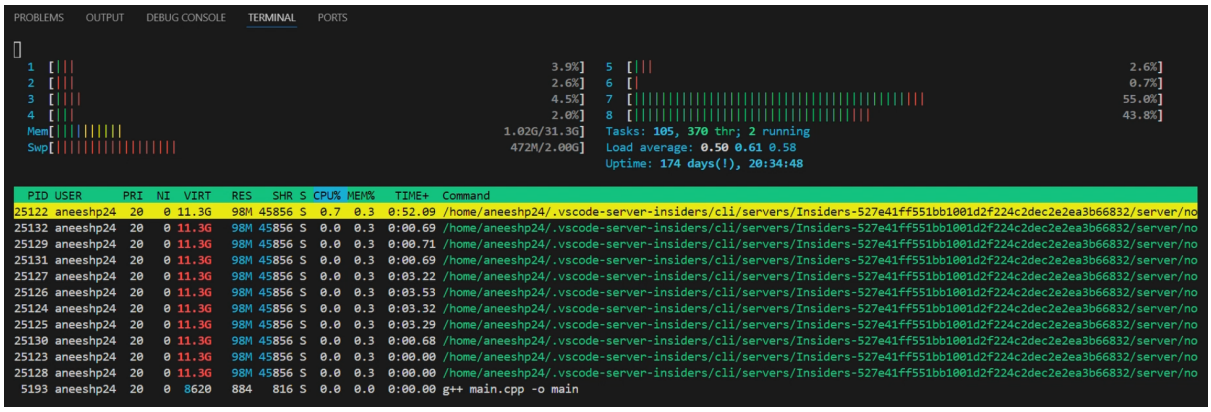

Figure 11: PID: 5193 [Best Case for Question 3]

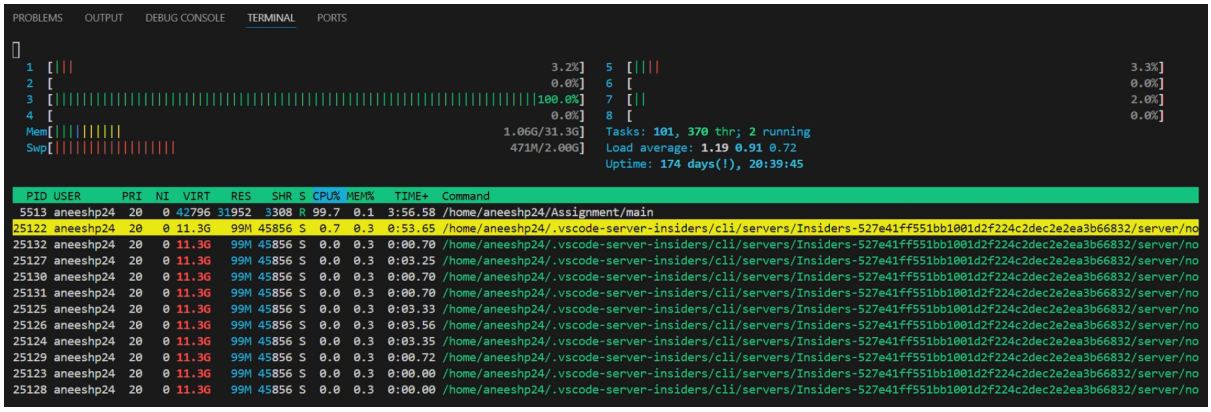

Figure 12: PID: 5513 [Worst Case for Question 3]

2.7.1 WHY BOTH (Q1 AND Q3) ARE OCCUPYING SAME SPACE ?

This is due to the reason that we are saving only customer to the memory in every run and hence, the value which were inside the function stack gets deleted and hence, the resulting occupied spaces are equal in both of the cases.

2.8 Edge Cases Considered

Most of the edge cases which arise in Question 1 are same for here as well, some of the other cases specifically for dynamic insertions are as follows,

1. New hashtags for a product not in input_product
   [**Passed** by ignoring the product]
2. Product not in initial mapping but included dynamically
   [**Passed** by including the product]
3. Same hashtags included multiple times for a product
   [**Passed** using Sets]

Tools and References

1. GitHub, Inc., GitHub Copilot. Retrieved Sept 2024, from https://github.com/features/copilot
2. Van Rossum, G., & Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.
3. Microsoft Corporation. (2018). Microsoft Excel. Retrieved from https://office.microsoft.com/excel
4. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms. MIT press, 2022.
5. Microsoft., Visual Studio Code. Retrieved Sept 2024, from https://code.visualstudio.com/