Distributed Operating System Principles Project 4
Twitter Clone Part II

Date: 15th Dec, 2022

Team members:

Hrushyang Adloori, UFID: 86776220, adloorih@ufl.edu

Goutham Kumar Mekala, UFID: 29386297, mekala.g@ufl.edu

Implementation Details:

- Initially both server and client files have to be compiled. The compiled .beam files have to be generated in their respective directories or nodes.
  >c(twitter_socket_engine).
  >c(twitter_client).
- Run the server to establish the main twitter engine.
  >twitter_socket_engine:initiate().
- Once the twitter engine is successfully started, we run clients on different nodes.
  >twitter_client:initiate().
- All functionalities can be achieved using the command line instructions.

Implementation of Functionalities:

(We have to type the command string exactly, or else the code prompts an invalid command exception and asks to retry again.)

- **Sign up:** Nothing can be done without registering an account. The code handles the error by instructing the user to register the account first. After opting to sign up, give a unique username which has not been used before. If used, the code prompts you to give another username to retry.

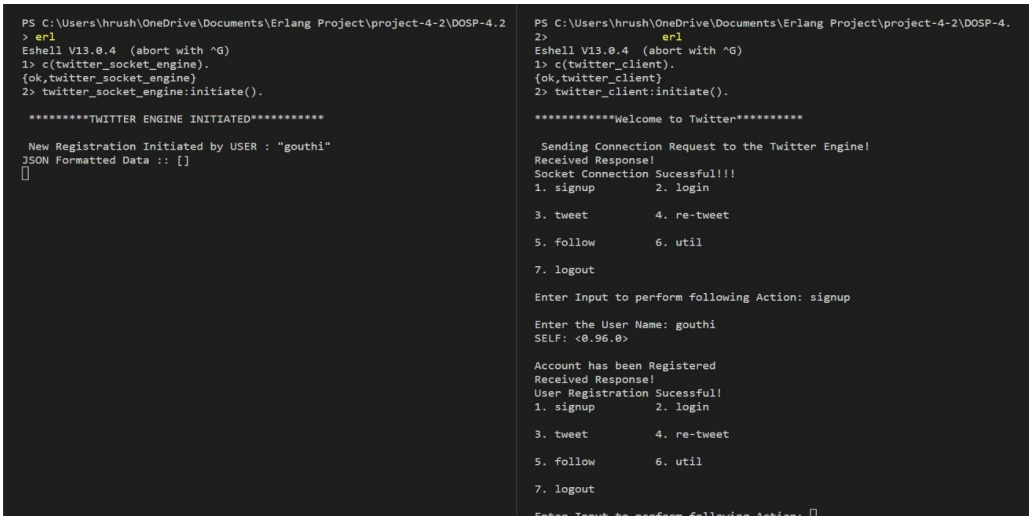


Fig.1: Implementation screenshot of Signup and Login

- **login :** After initial sign up, you're logged in by default. But once logged out, we can login to previously registered accounts using this command. We have to enter the exact username used before. Then we continue to achieve further functionalities.
- **tweet :** After logging in to desired account, we can now tweet anything we want that can include message, mentions (containing "@") and hashtags (containing "#"). This tweet is concatenated to the JSON file and sent to the server. This file contains key-value storage, that stores tweets in whole, mentions, hashtags and followers.

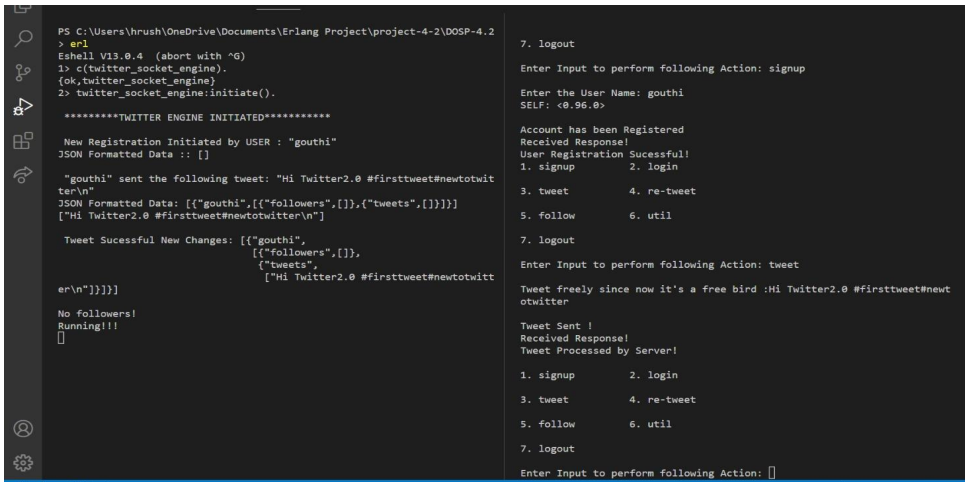


Fig.2: Implementation screenshot of tweet

- **follow :** This functionality allows the user to subscribe to any existing user. We need to enter the exact username to subscribe when prompted. By doing this, the user gets live updates of tweets tweeted by the subscribed user.

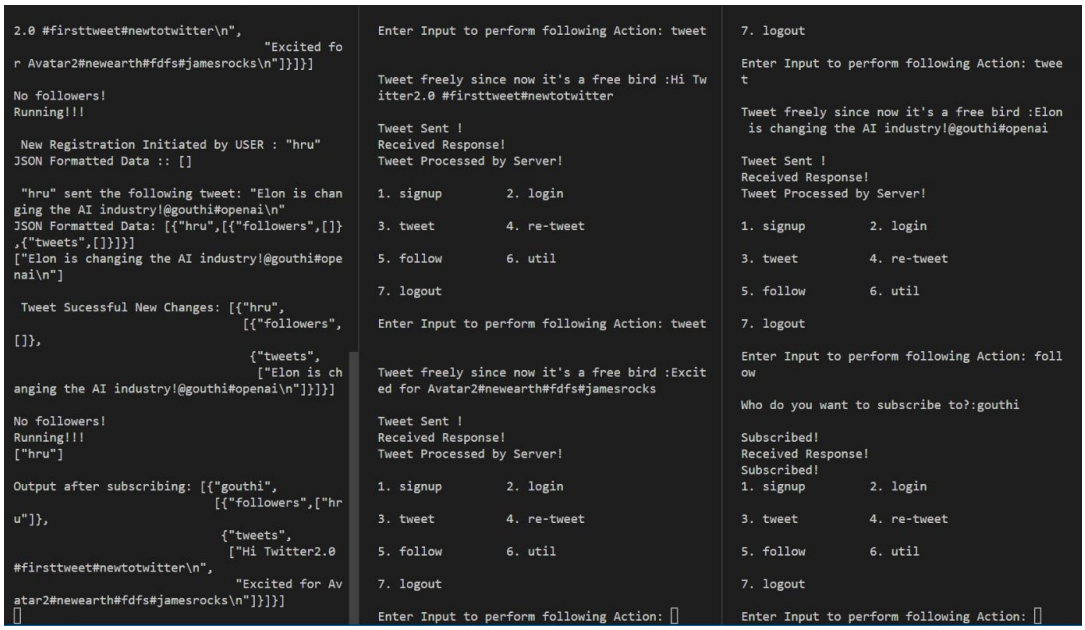


Fig.3: Implementation screenshot of Subscribe

- **re-tweet :** We can use this functionality to retweet an existing tweet. For re-tweeting, we need to enter the exact username who has tweeted it previously. Then we need to give input of the tweet he has previously tweeted. This counts as a retweet for the current user.

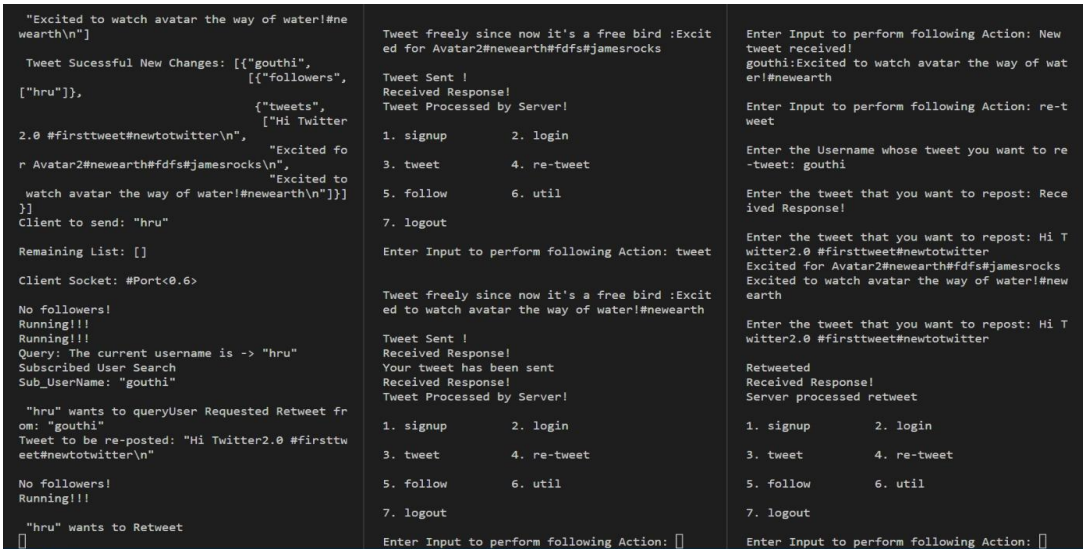


Fig.4: Implementation screenshot of Re-tweet

- **logout :** Once we are done, we can log out of this account which suspends the sustained socket connection. While the records are not lost until the server is shutdown.
- **util :** The util functionality contains another sub-menu that contains commands which can be used to query tweets. By opting this command. We are introduced to new set of functionalities that are described below:
  (Enter input as integers denoted to the command respectively.)
  - ❖ **@MyMentions :** A user can mention another user in their tweet using the symbol “@” followed by the username of the user to be tagged with no spaces. This tags user, when opted this command, the user can see all tweets they are mentioned in.

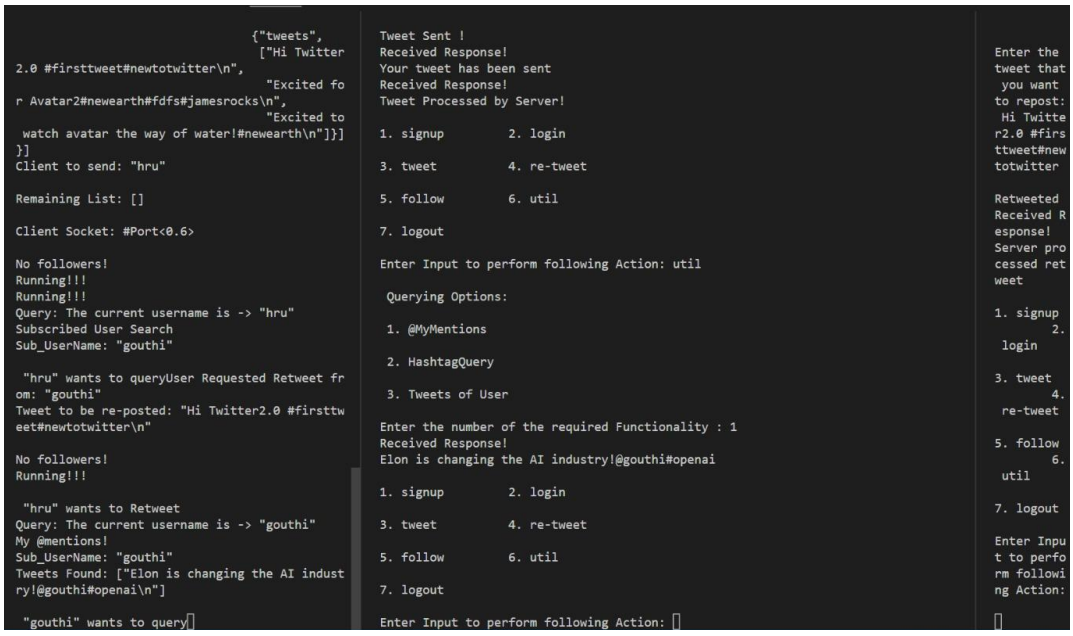


Fig.5: Implementation screenshot of MyMentions

  - ❖ **HashtagQuery :** The user will send a request to the twitter engine to search all tweets that contain the hashtag given as input.

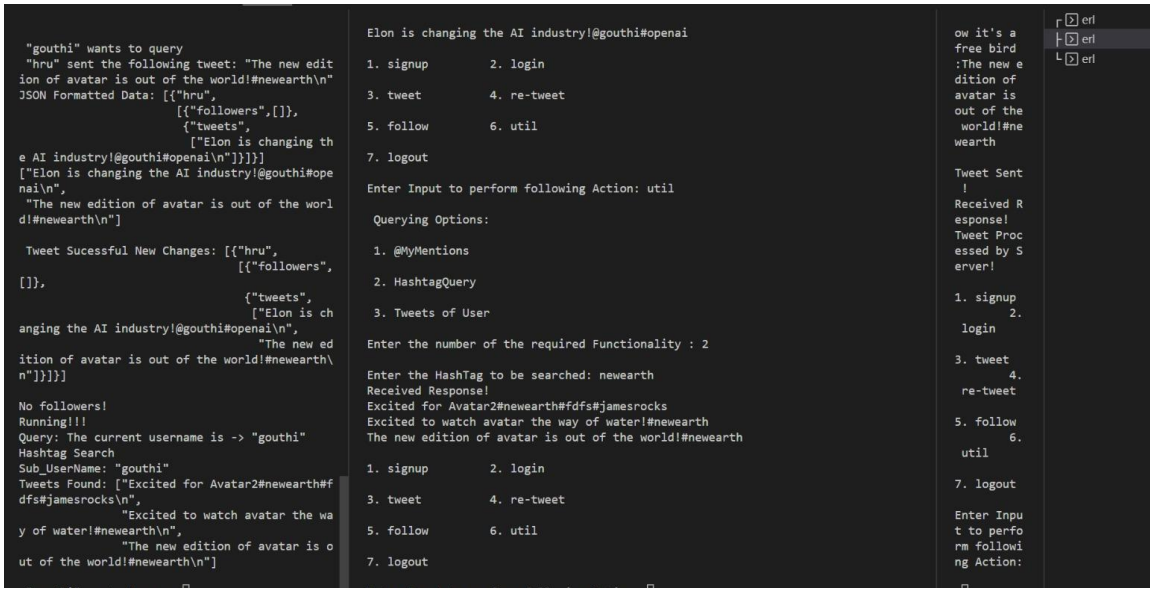


Fig.5: Implementation screenshot of HashtagQuery newearth

- **Tweets of User :** Upon giving input of the username, the function previews all the tweets tweeted by that particular user on the client console.

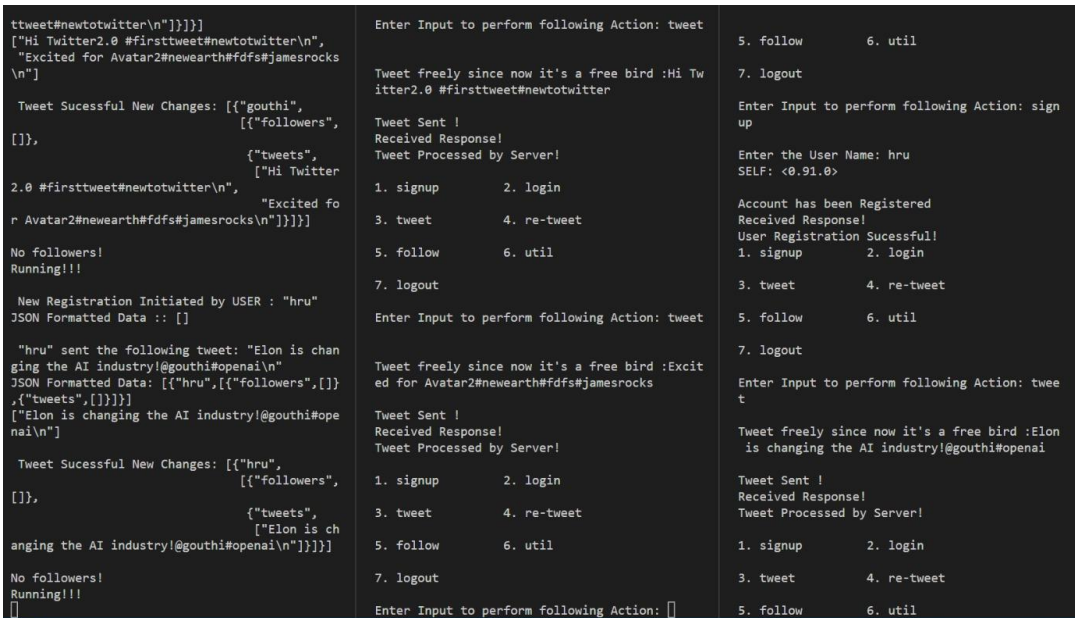


Fig.6: Implementation screenshot of Mentions

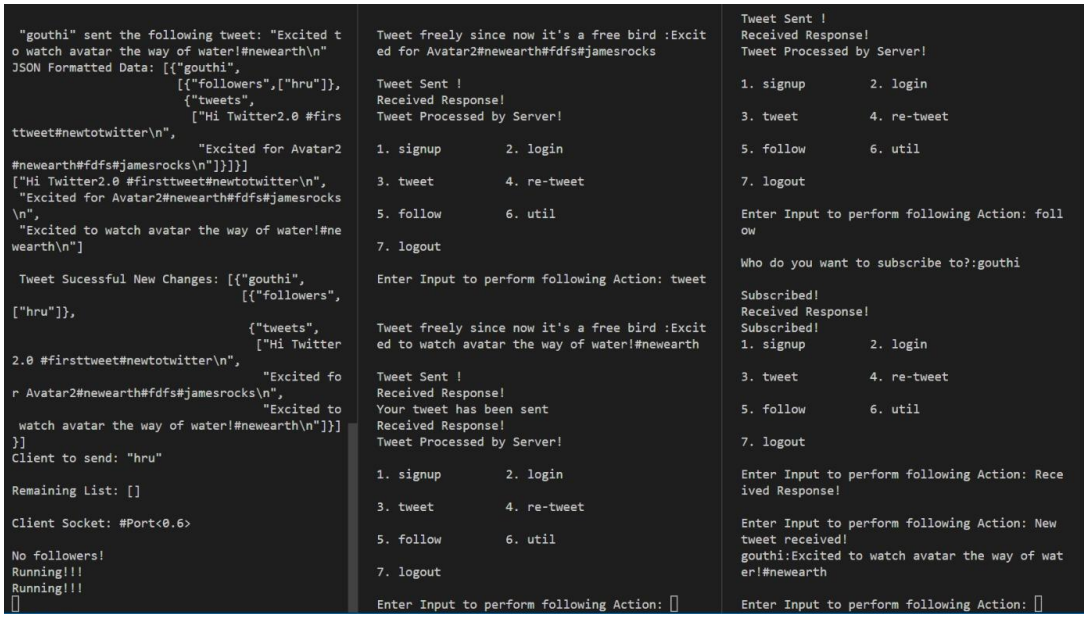


Fig.7: Implementation screenshot of live feed of subscribed user

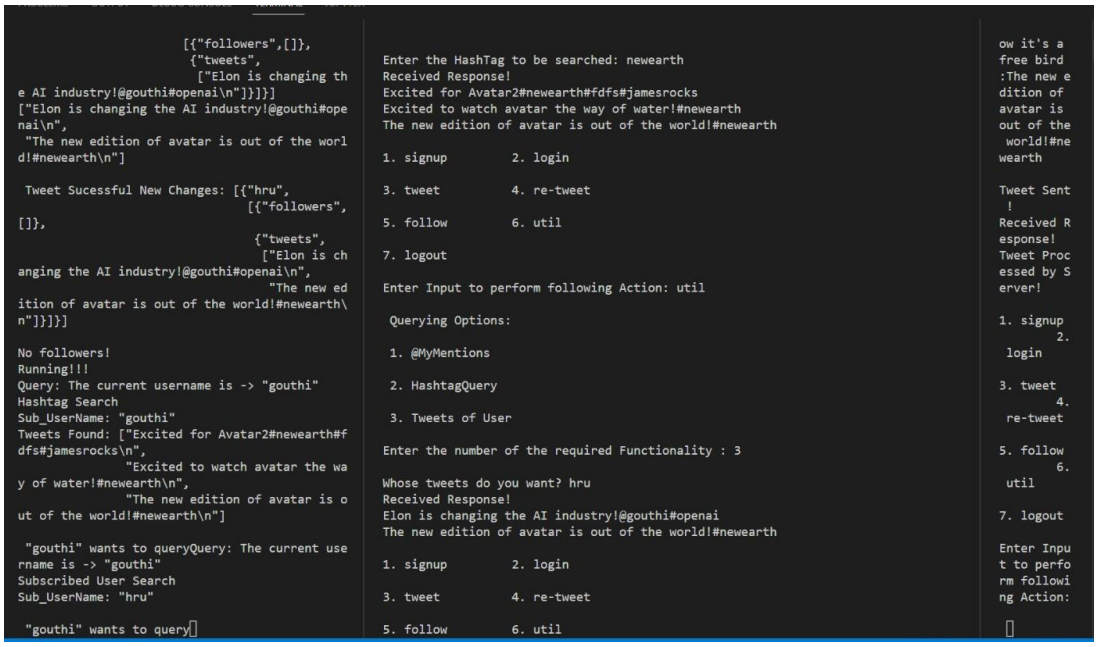


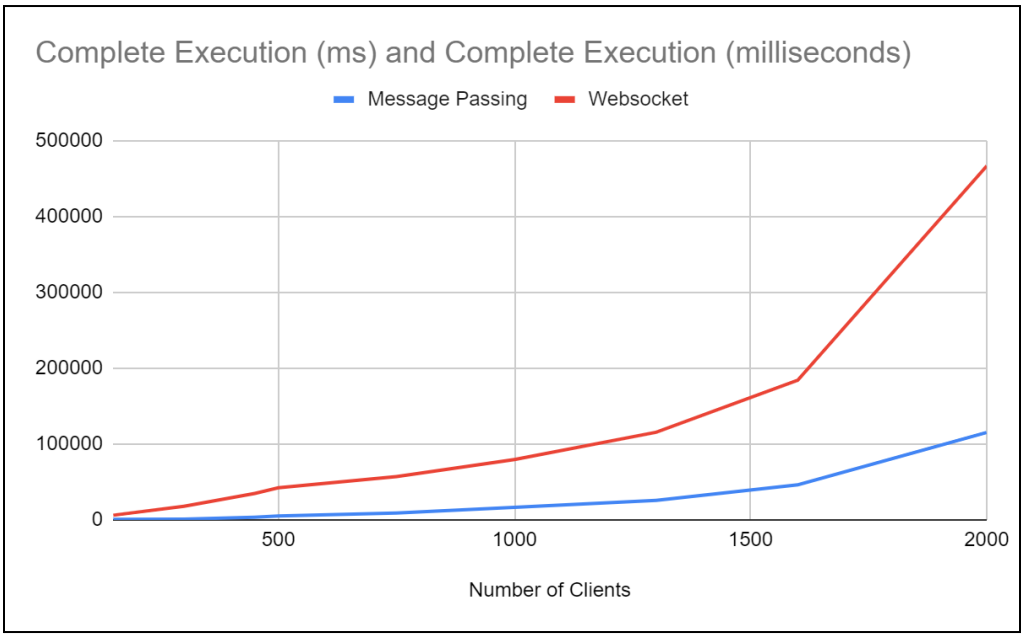
Fig.8: Implementation screenshot of tweets of user "hru" previewed from "gouthi"

Performance Results:

For message-passing systems, a graph is generated to show how the number of clients affects how long the simulation lasts. A maximum of 6,000 clients have been generated on our system.

Performance using Websocket interface when Max Subscribers = Number of Clients

Max Subscribers	Time to Tweet, Retweet (ms)	Query Tweets Subscribed (ms)	Query Tweets by Hashtag (ms)	Query Tweets by Mention (ms)	Query All Relevant Tweets (ms)	Complete Execution (ms)
199	270.54	1254.76	378	80.98	67.34	6436
399	2654.43	16543.87	2345.87	1234.65	387.87	64567
799	3245.76	23456.76	9876.98	9543.76	5432.78	112341
1199	4532.67	47543.79	19876.93	6587.34	5463.26	199634
1999	5437.82	76875.34	13679.65	11543.63	1654.37	575437



Performance using Websocket interface for different number of Max Subscribers / Max Tweets per Account

Max Subscribers	Time to Tweet & Retweet (ms)	Query Tweets Subscribed To (milliseconds)	Query Tweets by Hashtag (ms)	Query Tweets by Mention (ms)	Query All Relevant Tweets (ms)	Complete Execution for Websocket (ms)
200	247.11	224.68	1017.83	594.71	218.51	5467
400	252.4	380.54	1362.83	549.22	197.97	8745
800	244.45	2791.65	1774.67	876.11	589.93	19874
1200	2443.35	18098.63	5158.14	4887.51	2263.81	123456
1999	4104	37803.92	19021.37	12507.01	7031.84	167897