CS342 –Project 1

Report

Arif Can Terzioğlu

21302061

Environment

My main system works on Windows 10 OS. On the hardware side, computer processer is 4 cores Intel i7 4700HQ 2.Ghz series. Ram amount is 16GB.

I installed Ubuntu 16.04 64bit via virtual machine. I allocated two core of my computer processor. 5000 megabyte is allocated for main memory.

During Part 'A' and Some Observations

Development of Part A started with learning usage of pipe structure. Especially, supplementary books of course helped me a lot. I defined the queues structure at the top of main function. Main function starts with creating the pipes. I defined two pipes non-blocking that are read end of CM to MP and write end of MP to C1. The reason why I define pipe that is MP to C1 is that for my implementation, during the execution once I increase input size and decrease pipe amount. I saw some errors and inconsistent outputs. This emerged from deadlock while sequence was sending. Sequence was sending at once in my previous implementation. In this case, for exp, pipe amount would be 1. Main process sends one number (2), it will be printed. Then MP sends 3. C1 return this value to MP since it is only child. However; on that moment MP tries to send new value. Therefore, I got stuck during this moment. By doing these non-blocking operations with TA, problem was handled with new sequence sending mechanism.

I created each process in a for loop. Since each process has its own code segment, each process has a separate variable but the code fragment they do the same in terms of work. I used -8 for termination for child processes. Filtering and printing operation can be seen at the code. PR creation is separated from other children processes creations.

At the parent side, parent first creates a sequence and put them in a queue. Then, in a while loop, there are two main things that are sending the sequence and reading the numbers. Main processes sends number in another loop until there is some unsuccessful send operation. While doing this, it also checks for -1 and if -1 is successfully sends to C1, flag is 0. While flag is 0, operation continue with reading the value and when -1 comes back to queue again. Sequence is send to C1 again(flag=1).

Trials of Part B

I worked on a lot on part B. After the doing part A, I tried to transform this structure for part B. Once I learnt usage of message queues, I replaced pipes with message queues in an appropriate form. However, I could not handle the non-blocking side of this part until the deadline. I confused a bit on usage of queue. Since pipes have two control one for writing and one for reading. We can make them non-blocking separately on the pipes. However, on message queues, there is no difference like that. At least, to show what I tried I added to my work into .tar file. Program gives only the first prime numbers and sometimes first M numbers of prime numbers.

Methodology

Each time is measured three times and their average values are taken. After the graphics, results and conclusion are explained.

Inputs: 1000, 10000, 50000, 100000, 250000, 500000

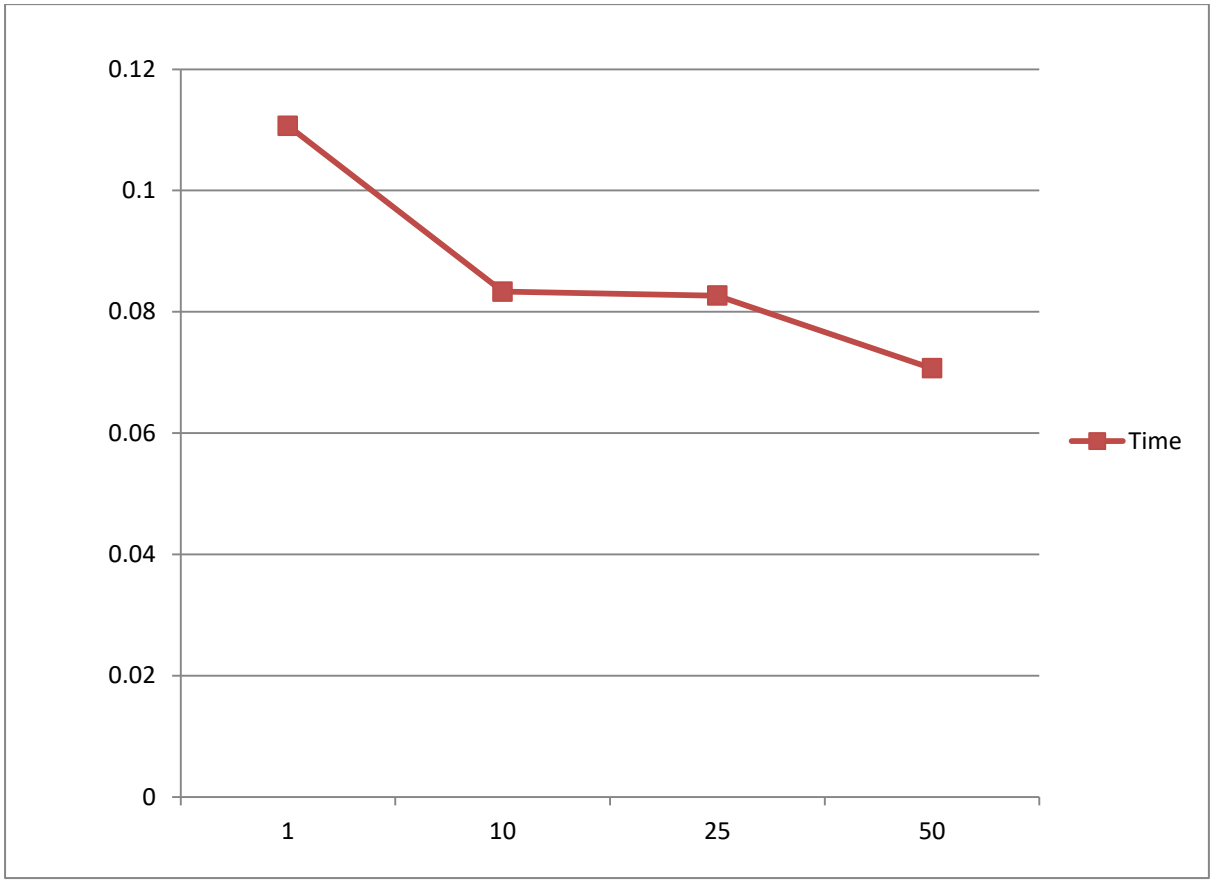
Pipes: 1, 10, 25, 50

Results

Increasing numbers of processes contributes positively based on results. Since each process we filter prime numbers and their multiples are eliminated. Processes are getting faster during the execution.

All results are given in terms of seconds.

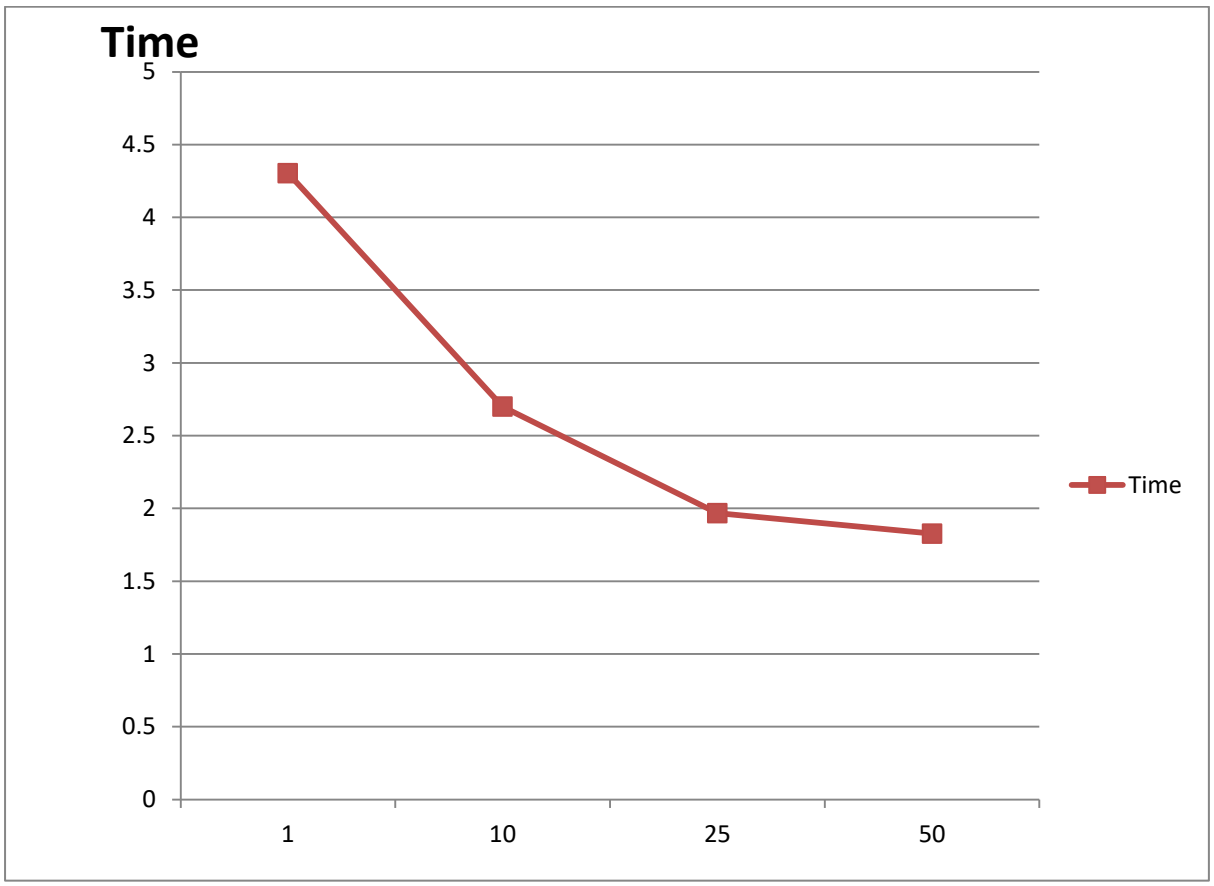
N=1000



When we give 1000 numbers, as expected in a very small amount of time, program completes. Increase on processes amount affects not much because time is already small. As we go from 1 process to 50 processes, time decreases. For one process at most 0.134 and for 50 processes at most 0.062 are observed.

	1	2	3	Avg
1	0.081	0.117	0.134	0.110667
10	0.081	0.087	0.082	0.083333
25	0.082	0.085	0.081	0.082667
50	0.08	0.07	0.062	0.070667

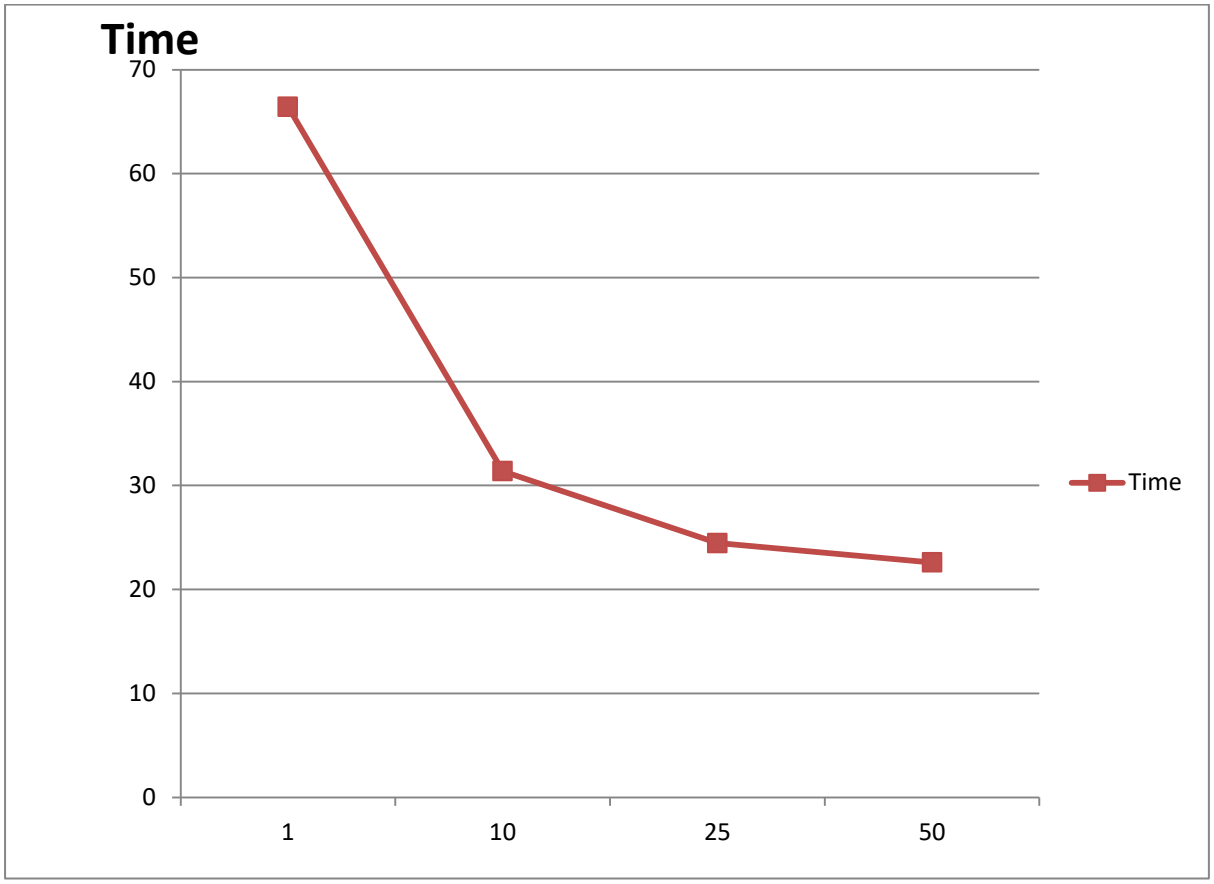
N=10000



When the input size is 10000, effects of increasing processes can be observed better. For one process at most 4.499 and for 50 processes at most 1.936 is observed.

	1	2	3	Avg
1	4.184	4.499	4.228	4.303667
10	2.718	2.585	2.793	2.698667
25	2.083	1.96	2.279	1.967
50	1.741	1.803	1.936	1.826667

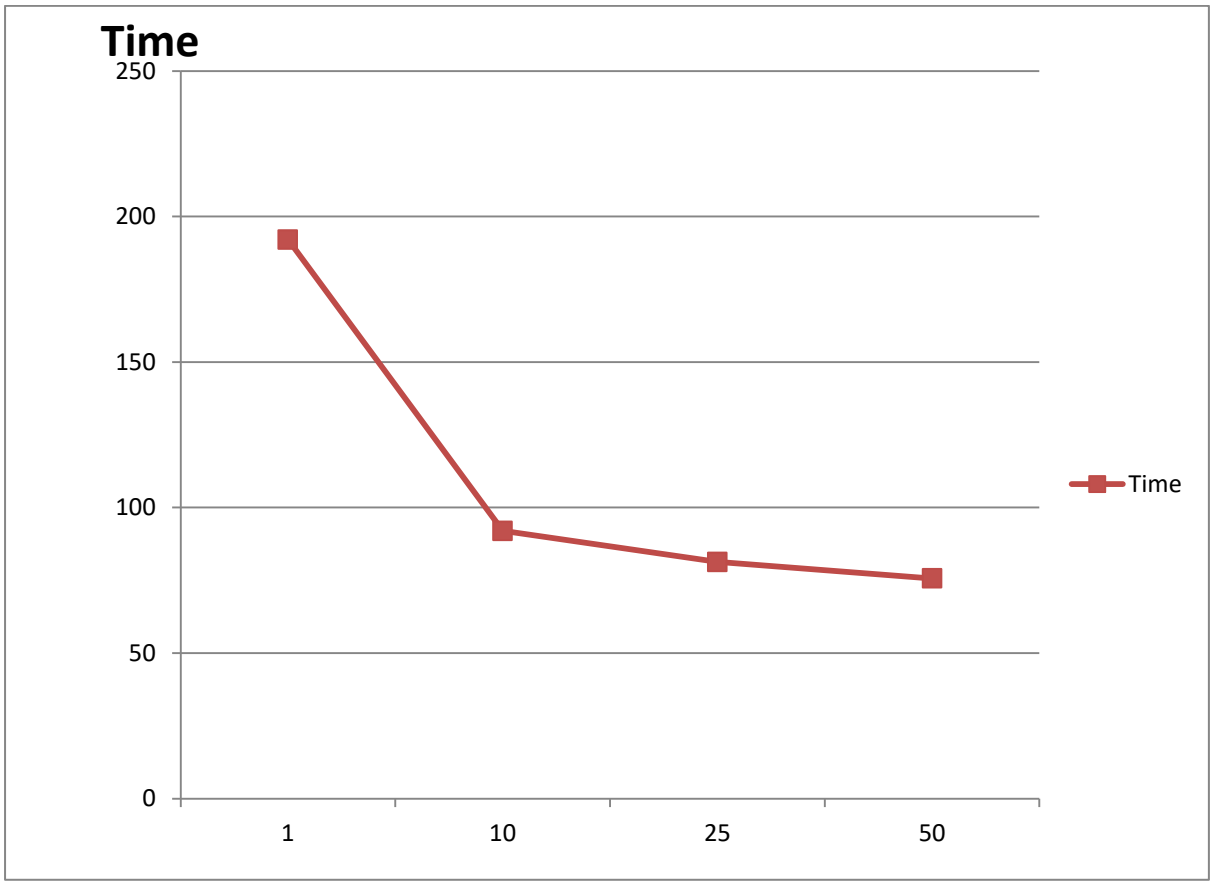
N=50000



When the input size is 50000, effects of increasing processes can be observed better in terms of minutes. For one processes at most 1 minute 7 seconds and for 50 processes at most 23 seconds is observed.

	1	2	3	Avg
1	67	68	64.3	66.43333
10	30.329	32.2	31.578	31.369
25	25.4	24.4	23.6	24.46667
50	23.782	22.1	21.9	22.594

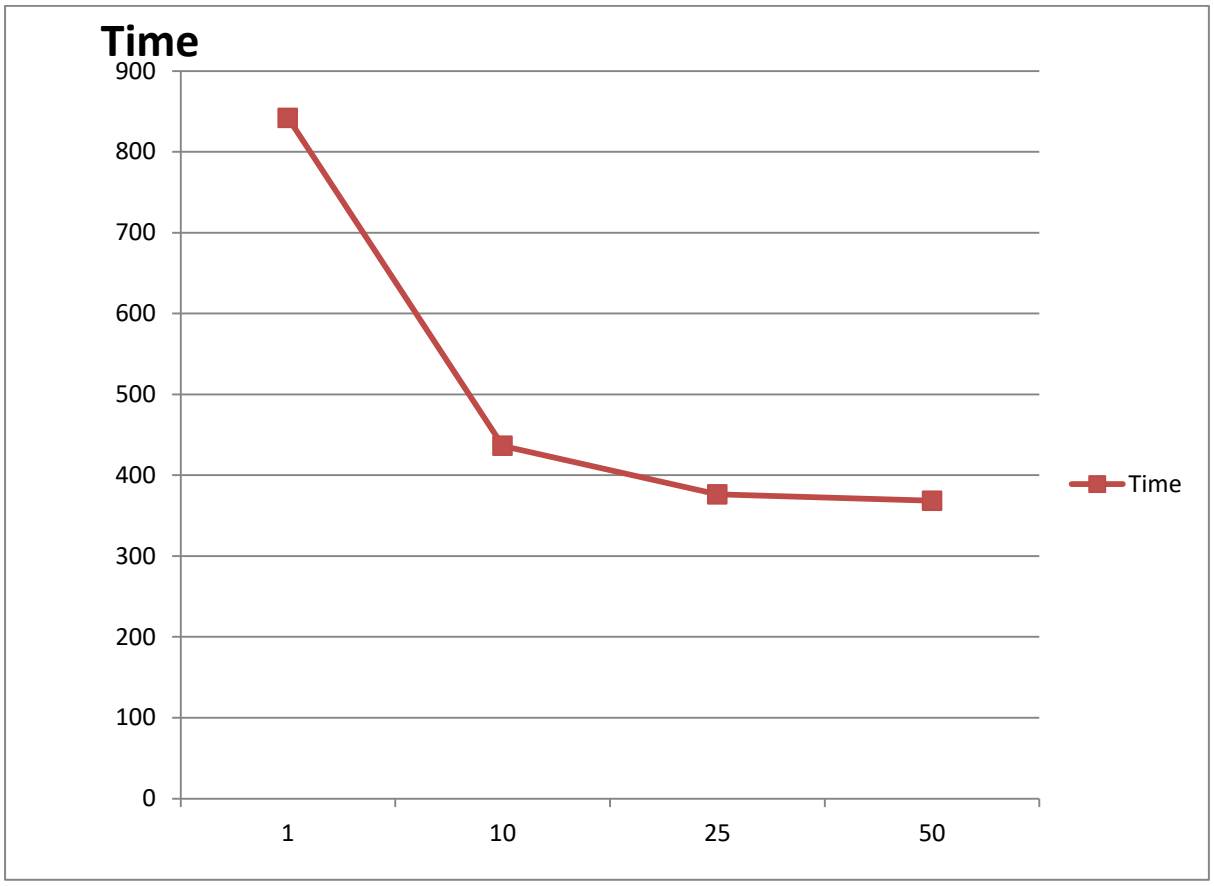
N=100000



As I mentioned on my part 'A' observation part, as n goes to infinity and M goes to minus infinity, time is increasing. Also , due to traversal time between pipes and non-blocking issues can be observed better for large inputs such as 100000 and M =1. Non-blocking issue is explained in observation part of report. In terms of time, for one process at most 3 minute 13 seconds and for 50 processes at most 1 minute 12 seconds is observed.

	1	2	3	Avg
1	193	191	192.2	192.0667
10	92	95	89	92
25	79	82	83	81.33333
50	72	80	75	75.66667

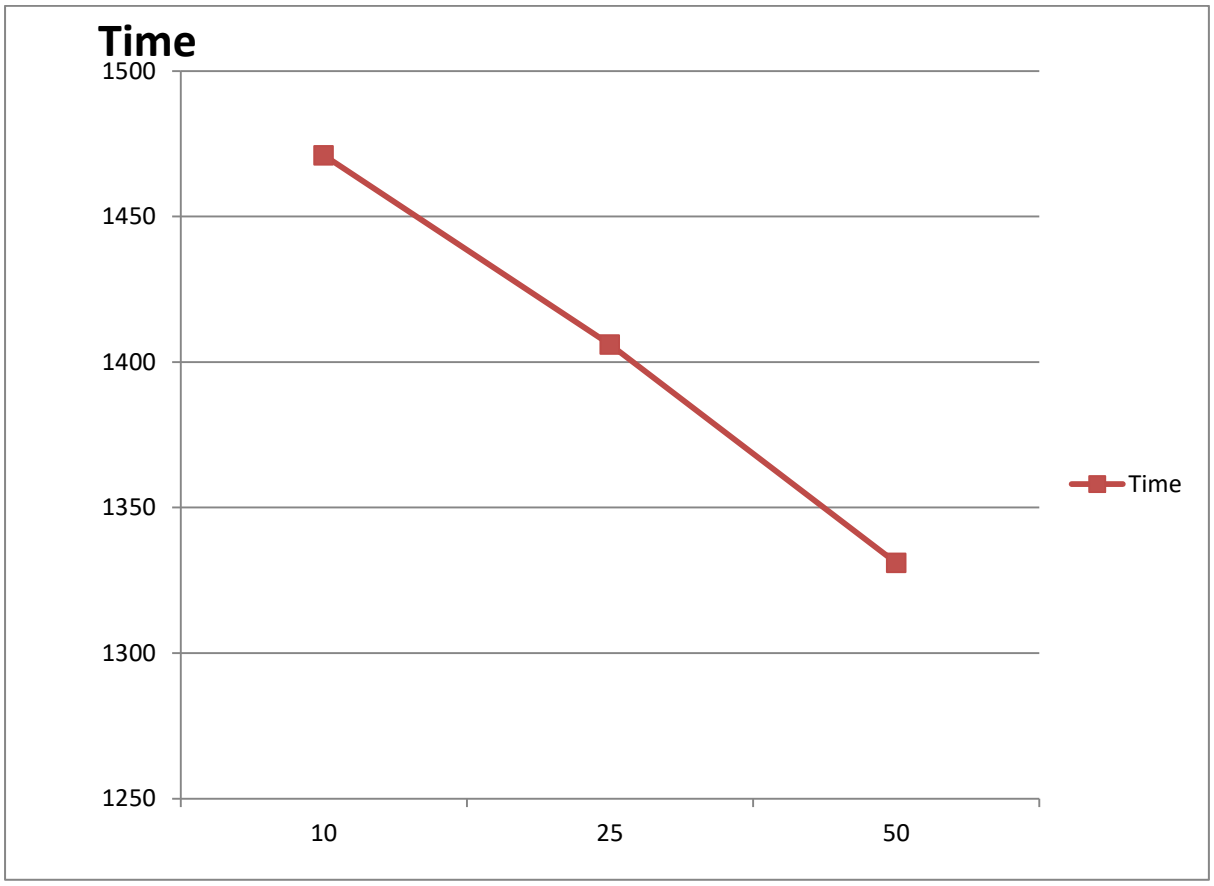
N=250000



In this input size, double digits are observed in time. With one processes, it was a really long process. It took around 14m 45 seconds. We can see that for large amount of data, one process is not useful. In terms of time, for one process at most 14 minute 45 seconds and for 50 processes at most 6 minute 7 seconds is observed.

	1	2	3	Avg
1	885	825	816	842
10	443	398.45	467.6	436.35
25	379	349.81	399.83	376.2133
50	367	368.45	369.65	368.3667

N=500000



I did not have a time to complete 1 million inputs for report but it works.

For N=500000, at M=1 process gave an error at N=69389. The reason of this is predicted at error section. Normally, it does not give an error but due to deadline, I could not measure again. For other values, difference is expected. I assumed that difference between each processes size would be more than 2 minutes for 500000. Difference between each process size is around 1.5 minutes. Difference is still can be observed anyway.

	1	2	3	Avg
10	1475	1355	1583	1471
25	1331	1356	1531	1406
50	1271	1499	1223	1331

Error:

My trial for N=500000 and M=1 was gave an error for one time (Killed at 69389). I thought that it may emerge from, while I am dealing with part b, some of my message queues is not closed. If they were at the background, they may affect the process.