CS 342

Operating Systems

Project 1

Mian Usman Naeem Kakakhel

21701015

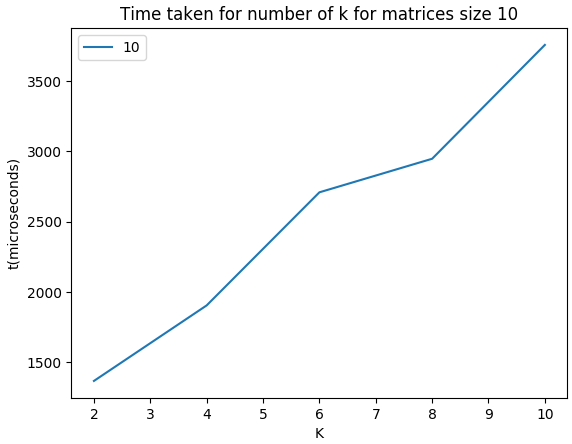
Sec: 01

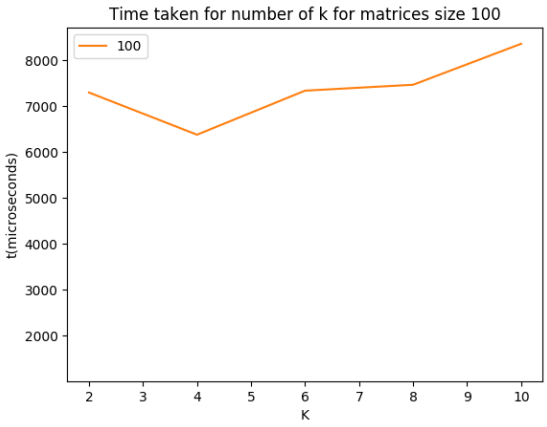
04/03/2020

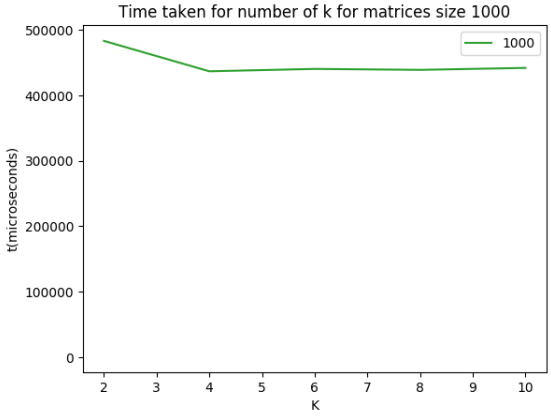
Experiments

The experiments were performed by taking K as 2, 4, 6, 8, 10 and the matrix sizes were taken as 10, 100 and 1000.

a) Processes with intermediate files:

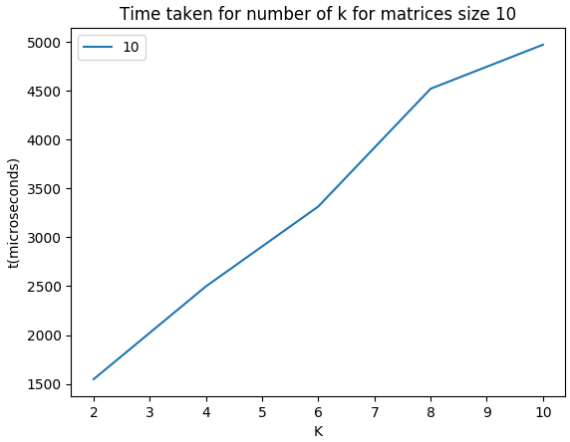


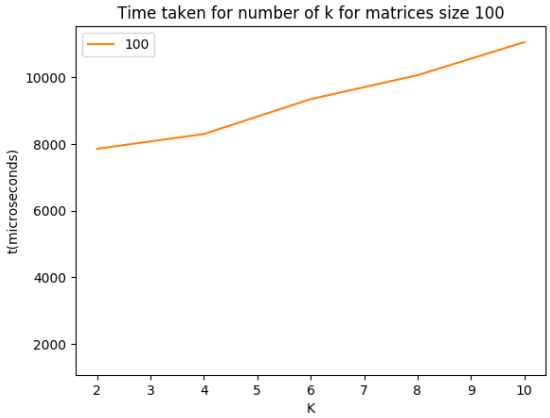


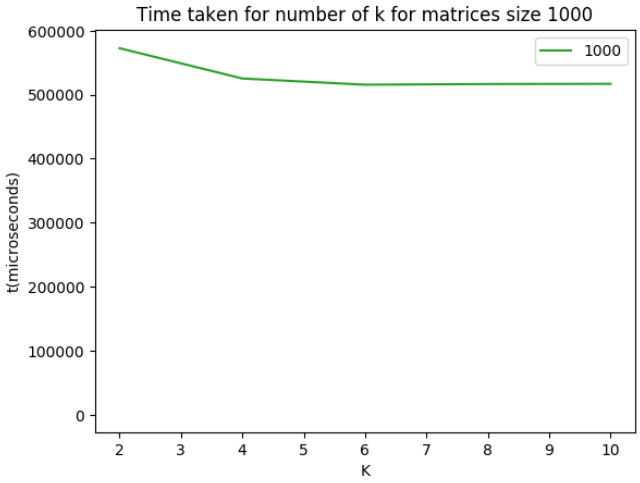


As it can be seen, in the case of matrix size 10, increasing the number of processes created increases the time taken by the program. The reason is the small size of the matrix and dividing the task is considered inefficient. In the case of size 100, it can be seen that increasing the k till 4 lowers the time taken for processing but increasing k after that causes the program to be inefficient. In the case of the matrix of size 1000, we can see that till k = 4, the drop in time is almost 50,000 microseconds but after that the time taken does not really change thus changing k after 4 is still not a good idea in a matrix of 1000 size.

b) IPC with pipes:

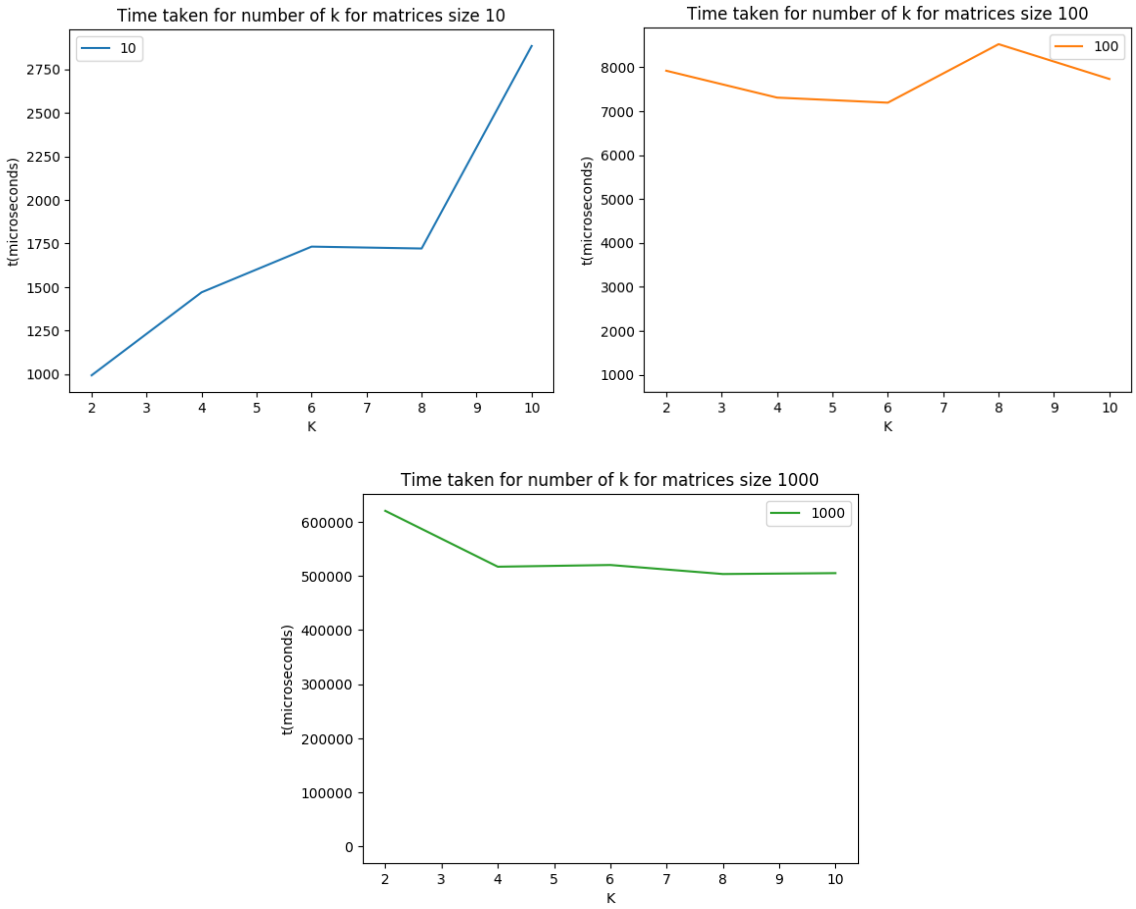






As it can be seen, in the case of matrix size 10, increasing the number of processes created increases the time taken by the program. The reason is the small size of the matrix and dividing the task is considered inefficient. In the case of size 100, it can be seen that increasing the k causes the program to be inefficient but the inefficiency is not as great as in the case of matrix size 10 as the increase in matrix size 10 is of 5000 microseconds while the increase in 100 size is 2000 microseconds. In the case of the matrix of size 1000, we can see that till k = 4, the drop in time is almost 50,000 microseconds but after that the time taken does not really change thus changing k after 4 is still not a good idea in a matrix of 1000 size.

c) Threads:



As it can be seen, in the case of matrix size 10, increasing the number of threads created increases the time taken by the program. The reason is the small size of the matrix and dividing the task is considered inefficient. In the case of size 100, it can be seen that increasing the k till 6 lowers the time taken for processing but increasing k after that causes the program to be inefficient. In the case of the matrix of size 1000, we can see that till k = 4, the drop in time is almost 100,000 microseconds but after that the time taken changes after that but not as drastically as from k = 1 to k =4.

Finally comparing the 3 methods, we can see that increasing k in a small size of matrix is not efficient as it divides tasks which could have been done better without division. In the size 100 and 1000 it seems that k = 4 is the threshold till which the the time decreases and then the division becomes inefficient.