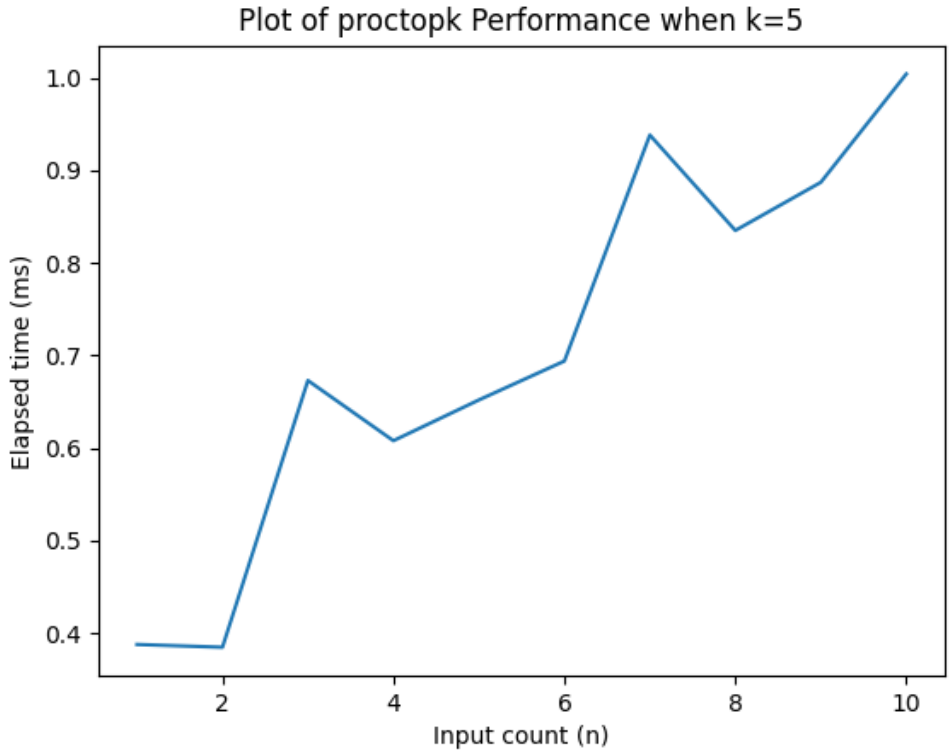
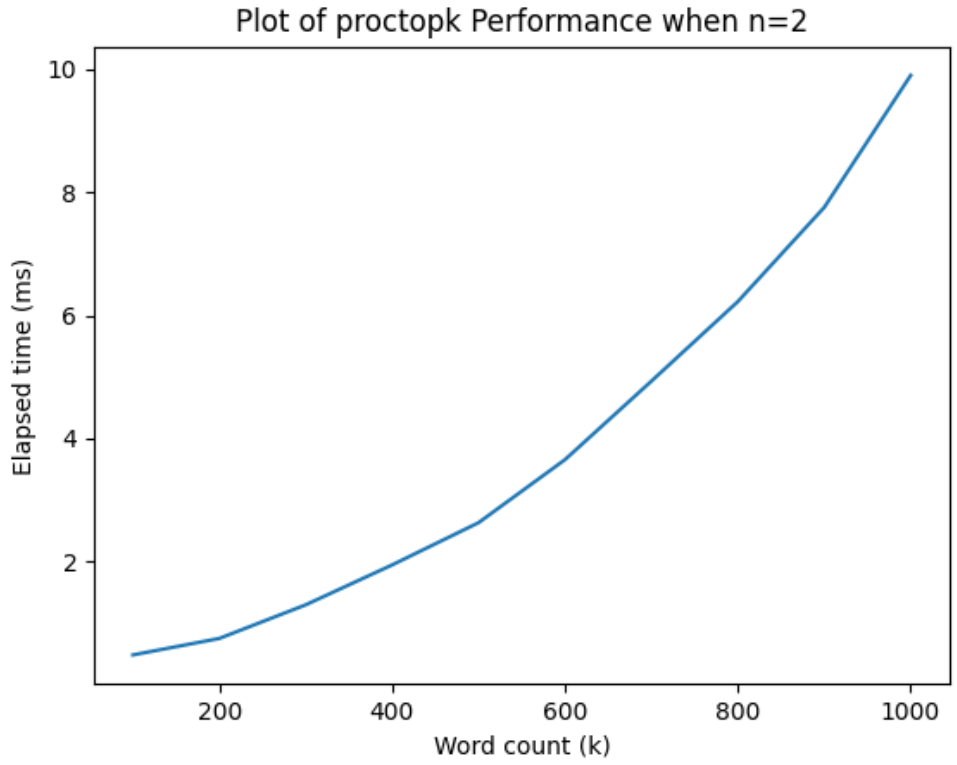
Ali Emir Güzey 22003186
Mustafa Kaan Koç 21903358
21.03.2023

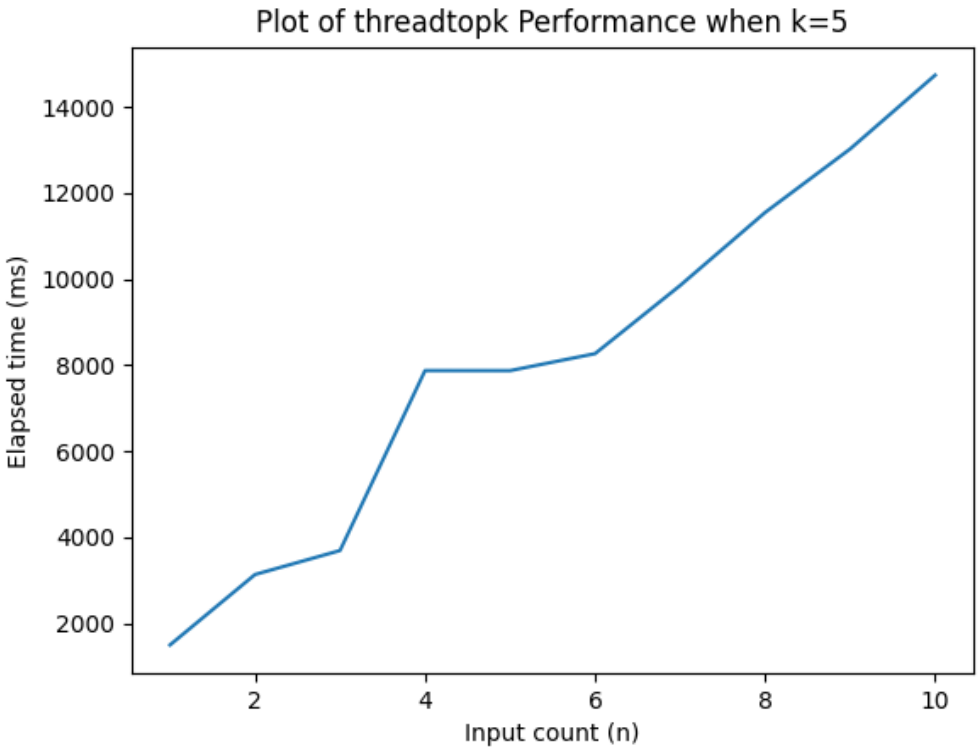
PROJECT 1 REPORT



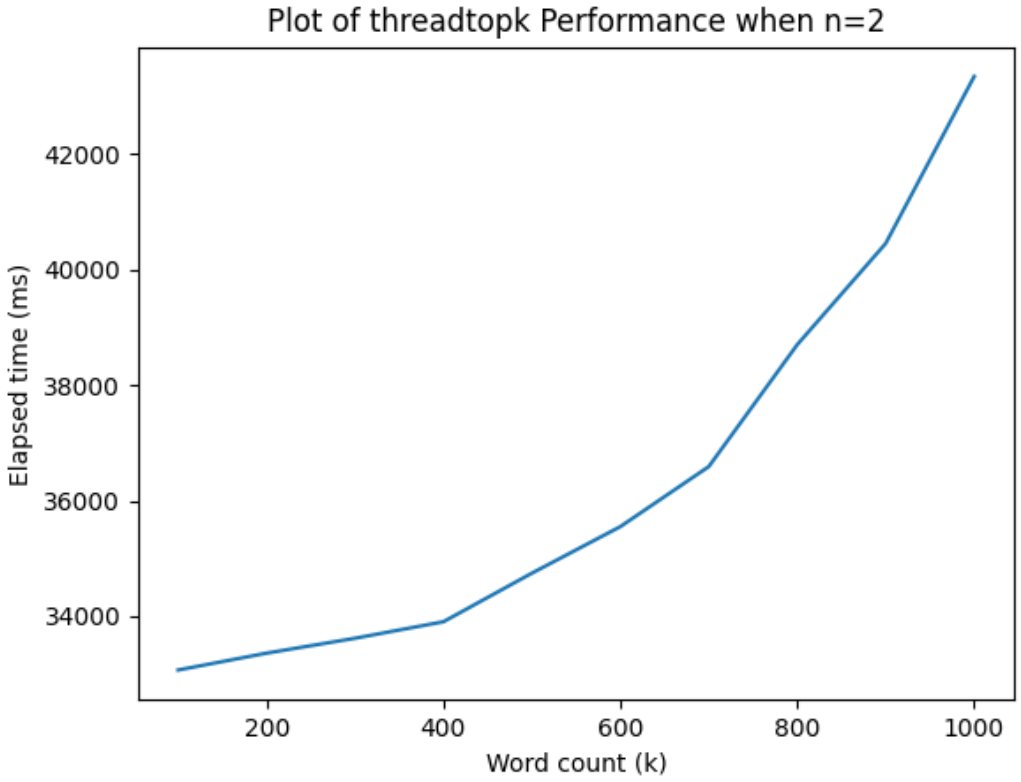
As shown in the graph above, when k is fixed to 5, elapsed time of the program increases step by step as n goes from 1 to 10. Since a virtual machine with two cores is used to run the program on Ubuntu 22.04, the run times for the first two n values are quite close. It is because when the program wants to create a second child, it uses the second core, and the elapsed time would not increase much. When k is fixed, n values for 1 and 2, 3 and 4, 5 and 6, 7 and 8, and 9 and 10 should theoretically be close to each other. However, experimental data has fluctuation that might be caused by environmental reasons such as computer workload.



As expressed in the graph above, when n is equal to 2, elapsed time of the program rises as k values go from 100 to 1000. However, this increase is not linear. As k gets close to 1000, the ratio of the increment of run time increases. The reason behind this situation might be the nested for-loops that boost the workload of each child. On the other hand, since the number of child processes stays the same, concurrency does not influence the performance of the program when n is fixed.



As depicted in the graph above, when k is fixed to 5, elapsed time of the program increases as n goes from 1 to 10. Since we have applied concurrency in the program, elapsed time does not increase linearly. Instead, the increment occurs step by step. However, the theoretical and experimental data differ due to environmental reasons, such as other processes that occupy the processor cores and the overall performance of the machine fluctuating from time to time.



As clarified in the graph above, the increment in the run time rises as k values grow. Since the n value is fixed to 2, two cores run simultaneously, and concurrency applies. However, this is not a linear increase because the program uses insertion sort, which has the time complexity of $O(n^2)$.