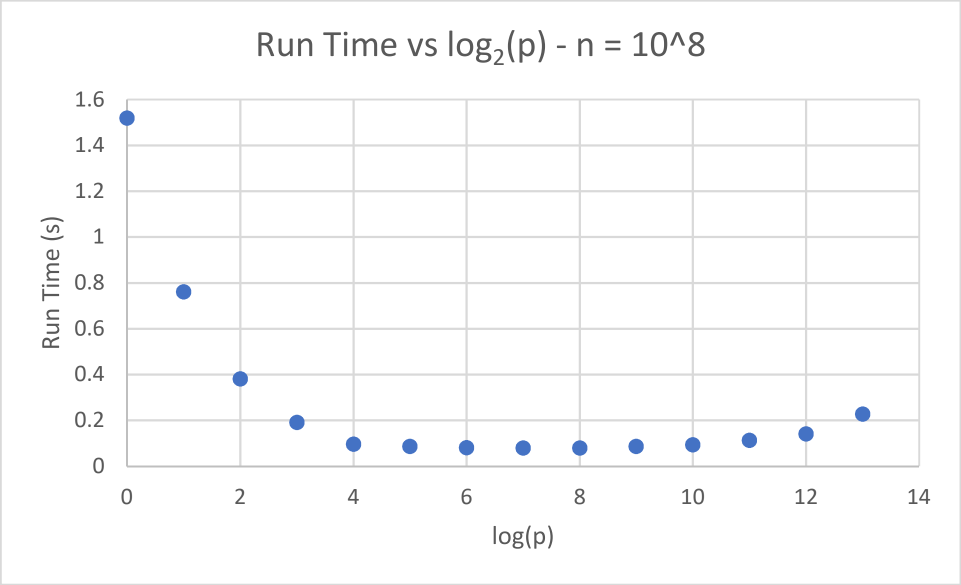
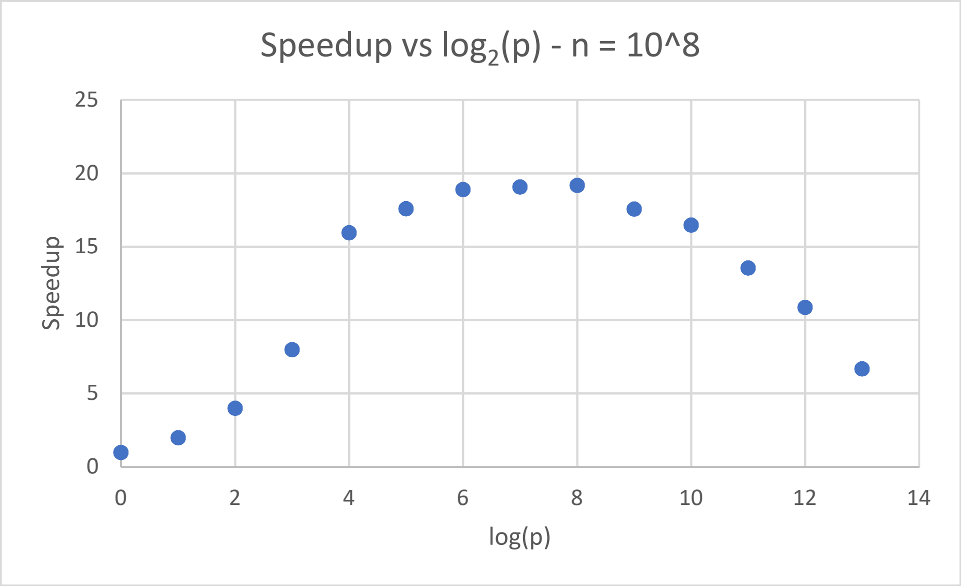
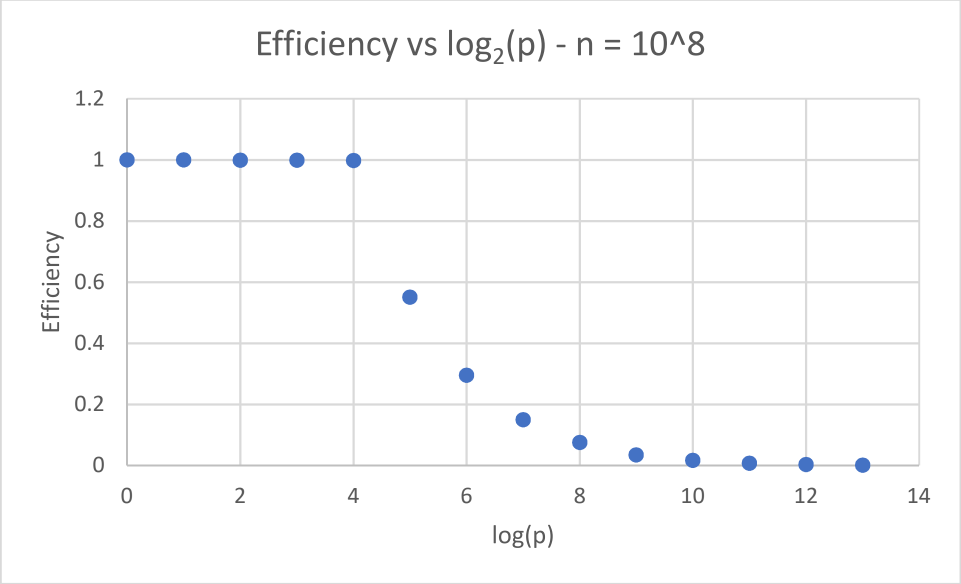
Jacob Hessong

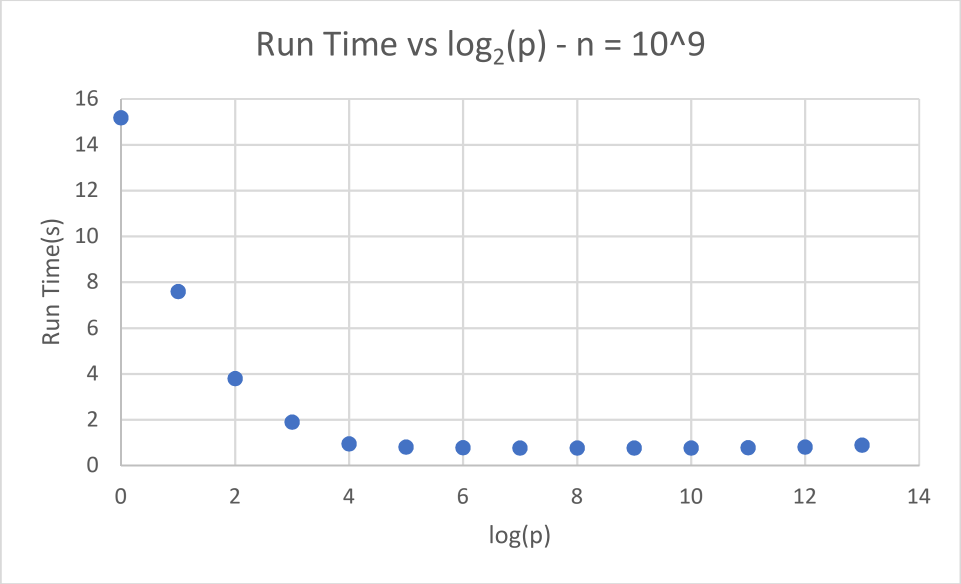
CSCE 435 HW 1

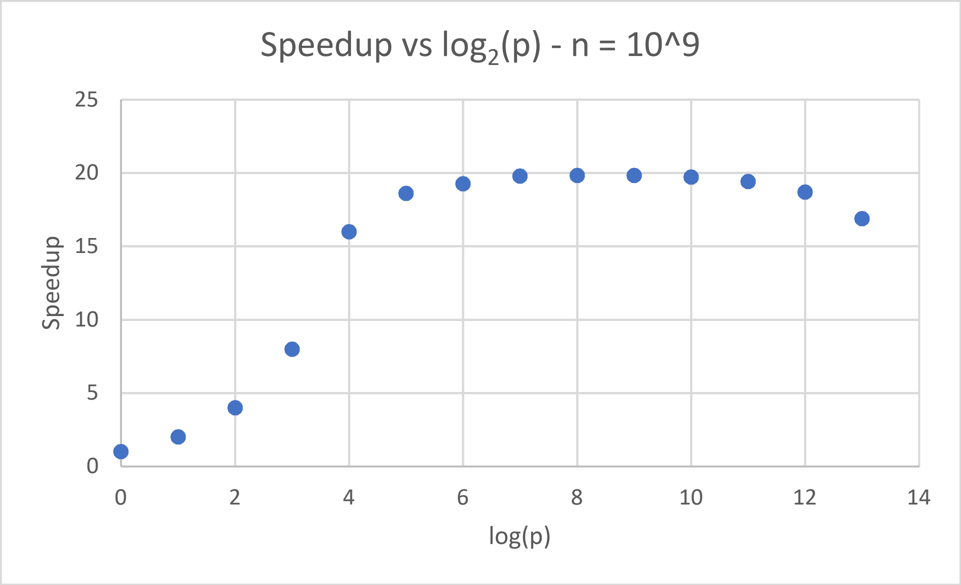


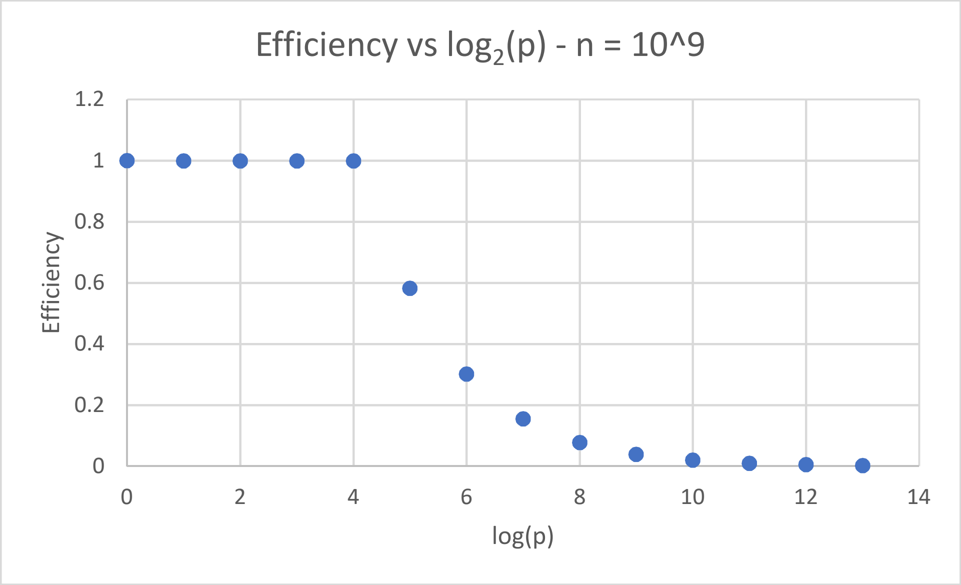




In these experiments, in which n, the number of trials, is equal to 10^8, the value of p that minimized parallel runtime was p = 256. This value also maximized speedup, however its efficiency was lower than trials with fewer processors.



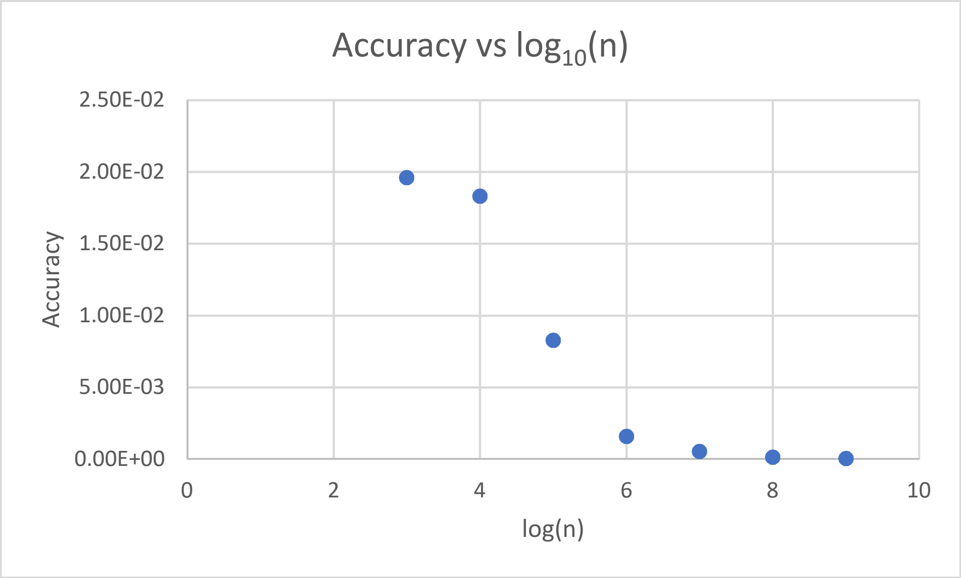




In the second run of the experiment in which $n = 10^9$, the value of p that had the lowest parallel runtime was also p = 256. Similar to the previous case, this p value had the highest speedup but lower efficiency than the previous trials.

When increasing p, one would think that having more threads would lead to increasingly faster and faster execution. However, at a certain point, the addition of threads leads to there being more overhead involved throughout the run time. Although there are more threads to run the code, switching between these threads and ensuring synchronization leads to diminished performance past a certain number of optimal threads.

In my experiments, the optimal value for p was the same for both runs despite the second one using 10x more trials. However, the different n values could potentially lead to different optimal values of p. This would likely happen due to the fact that there is much more work to be done when increasing the number of trials, therefore the extra overhead from having more threads may be outweighed by the added time from the extra trials. This would allow the optimal number of threads to increase.


Accuracy vs $\log_{10}(n)$
Accuracy
2.50E-02
2.00E-02
1.50E-02
1.00E-02
5.00E-03
0.00E+00
log(n)