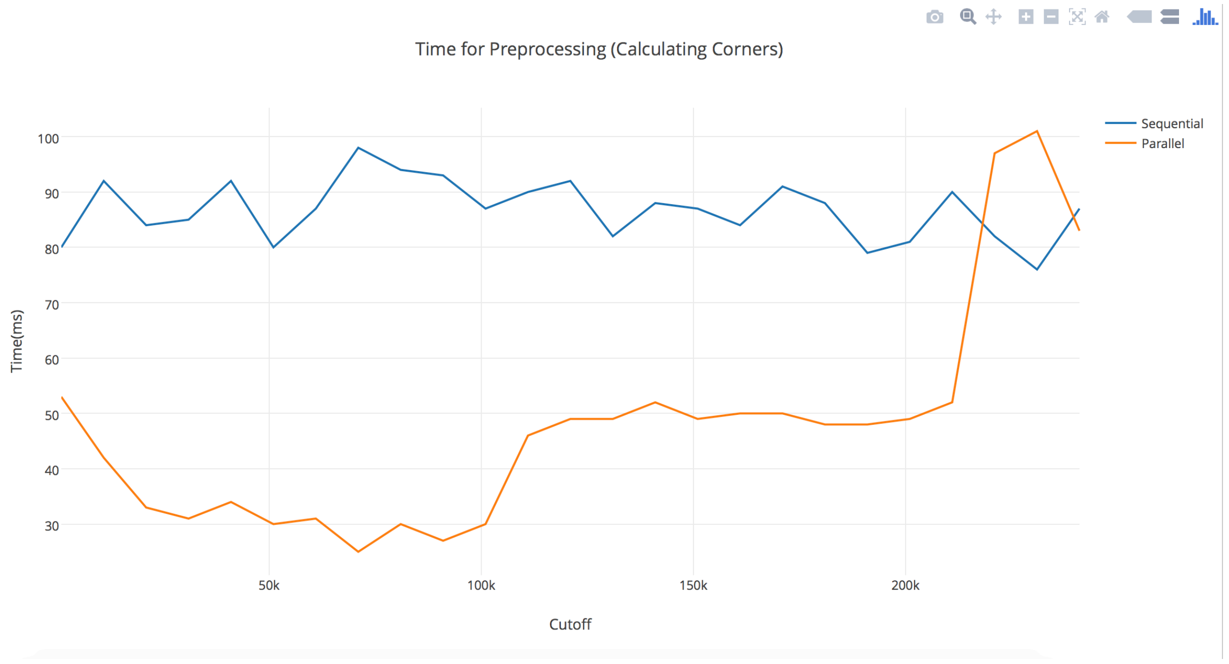
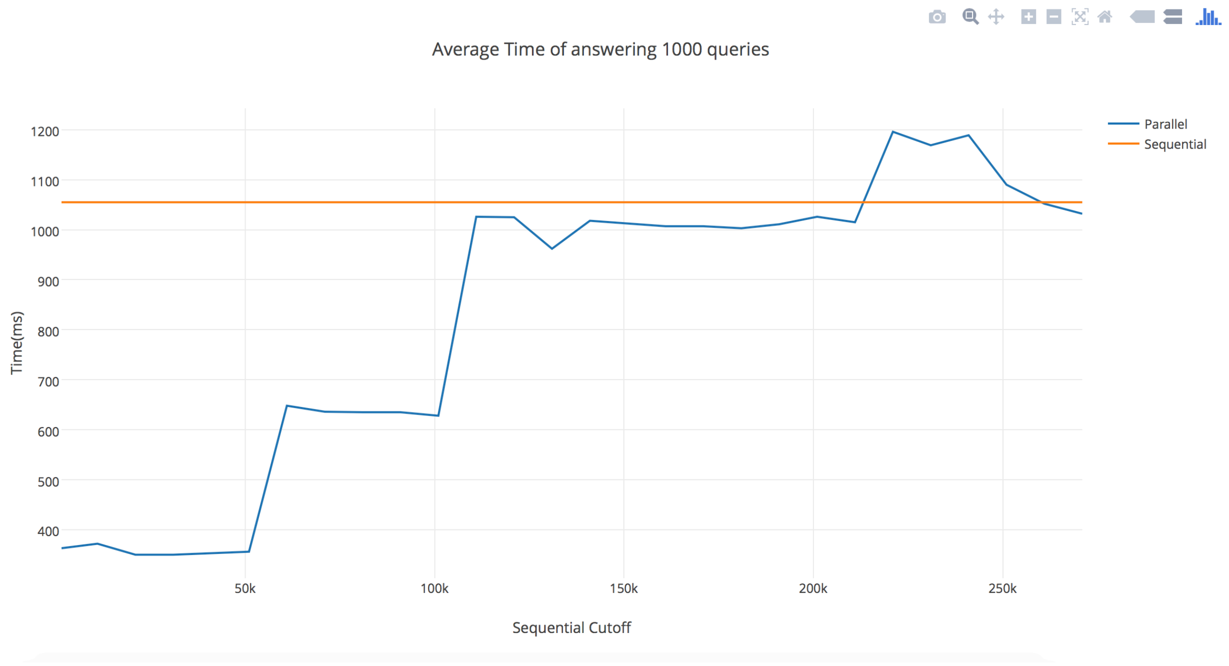
Assignment 4: Writeup

Question 6:

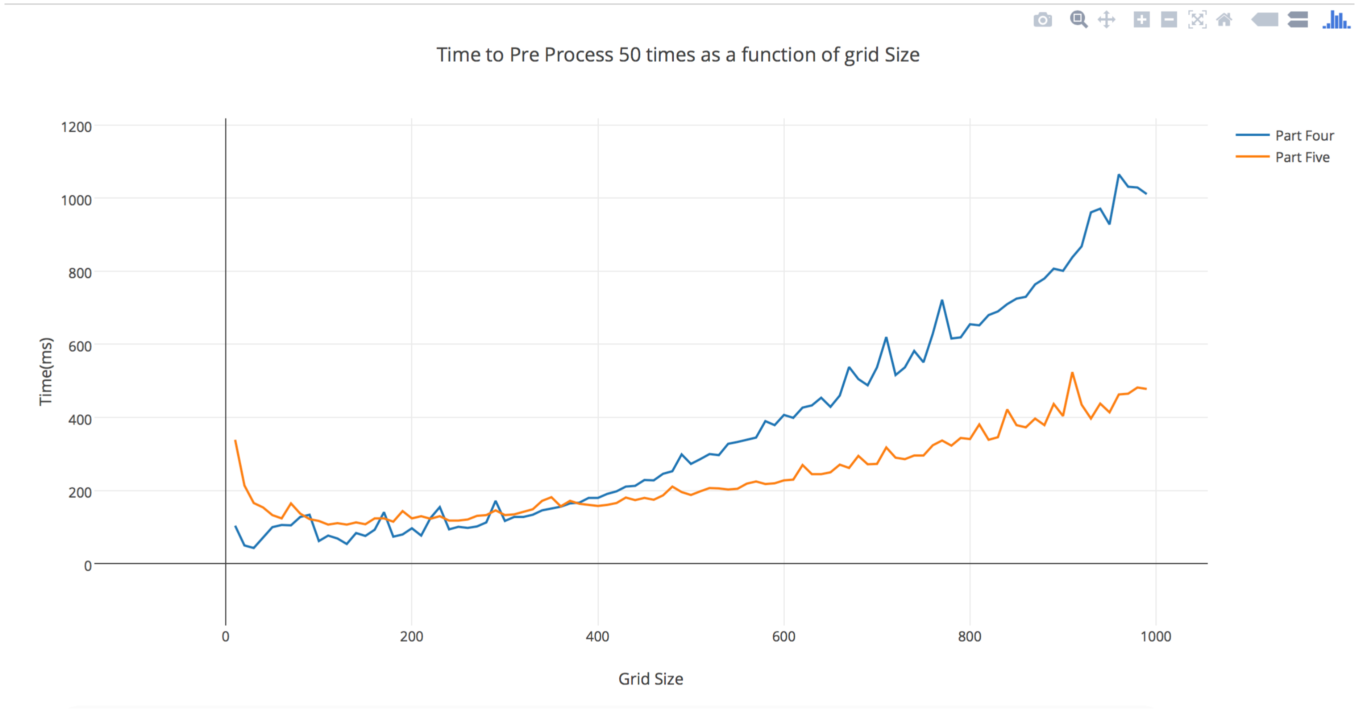


As can be seen from the graph above, the optimal sequential cutoff seems to be about 70000. The size of the data array is 220,000 and the performance of parallel algorithm reaches the performance for sequential at sequential cutoff approximately greater than the grid size. The change is abrupt as even a sequential cutoff data_size – 1 will divide the problem into two sub problems.

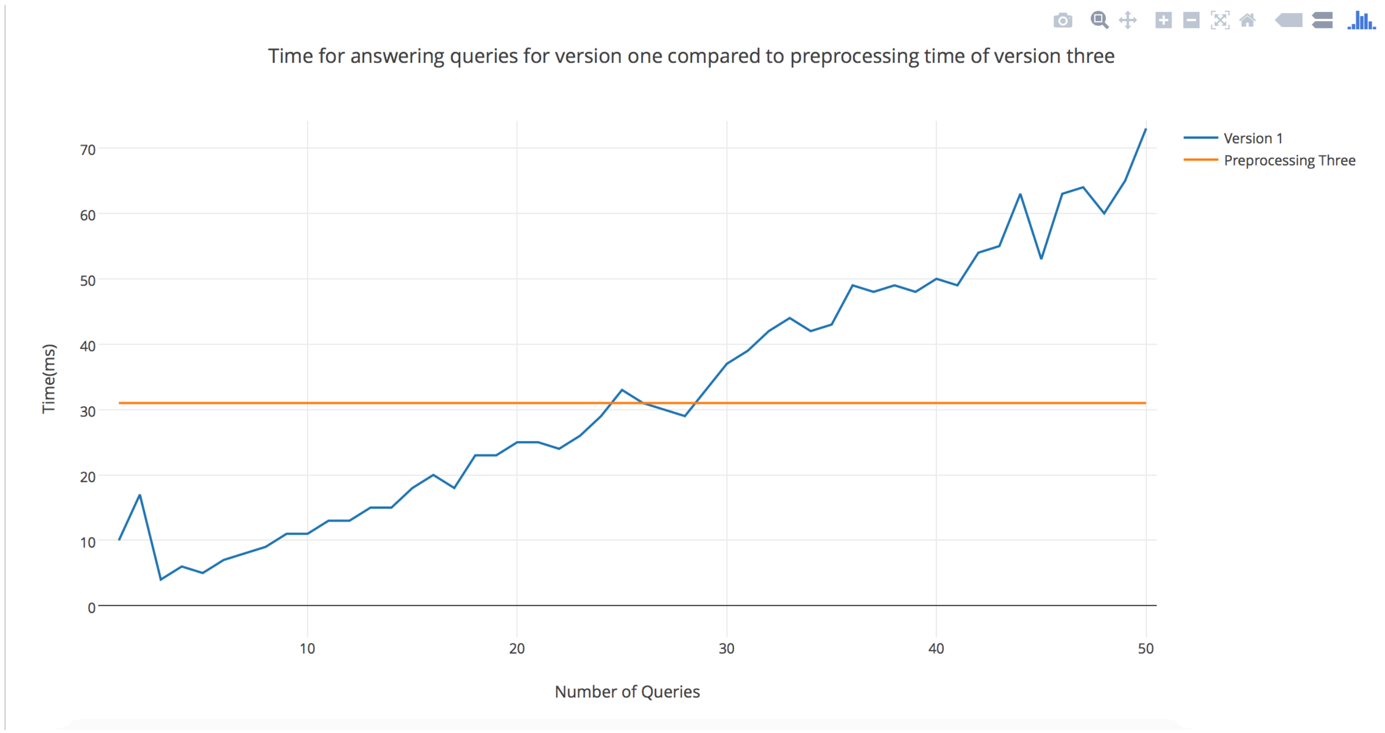


The above graph shows the time to answer 1000 queries (using version 3) as compared to a single average value of answering the same queries using version 1. The abrupt jumps in the time as Sequential Cutoff changes can be explained by the fact the amount of sub problems change at large intervals of the cutoff. As the sequential cutoff becomes greater than the data size, both sequential and parallel versions converge.

Question 7:
Intuitively, version 4 should be better for smaller grids and version 5 should be better for larger grids. The reason is that for smaller girds, version 4 will have to parallel sum a smaller grid at each level. The reason that version 5 is slower for smaller grids is that the possible grid positions that a CensusGroup can take is smaller and grows as grid size increases. The probability that two threads will have to make writes at the same grid location is high leading to higher contention of locks. At larger grid sizes, grid is divided on a finer grained basis leading to decrease in contention between threads. This is clearly visible in the graph below. Multiple runs were taken and the results averaged.

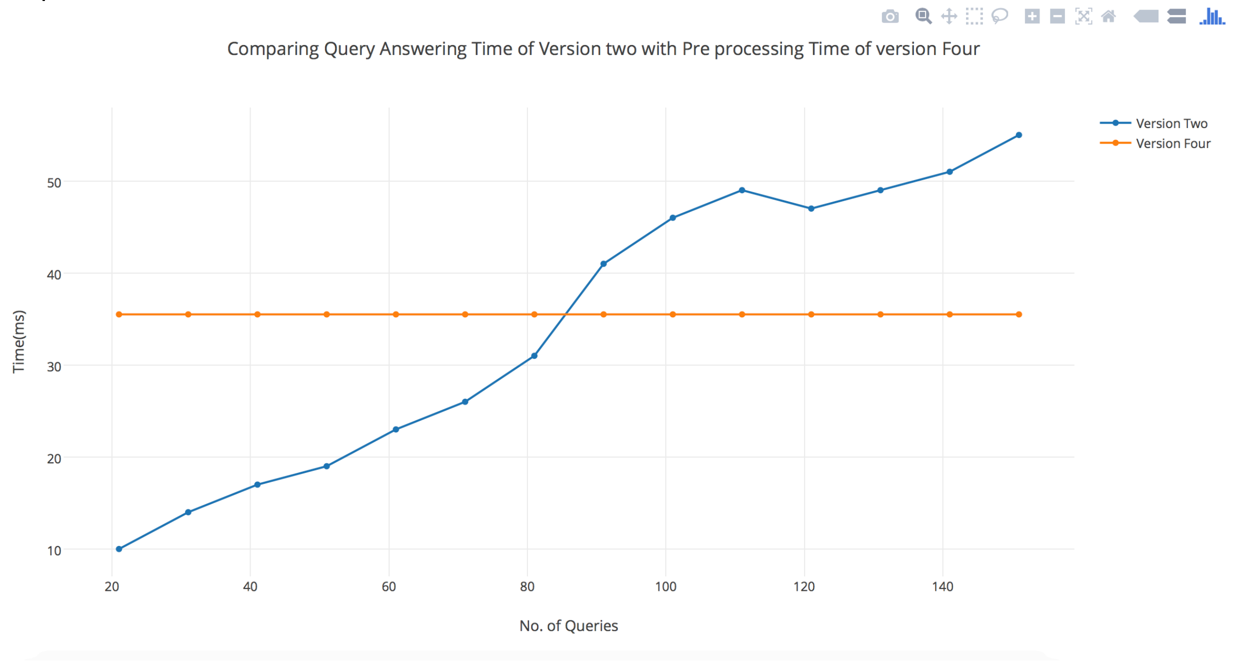


Question 8:
Comparison of version 1 with version 3



The graph shows the time for answering some number of queries using version 1 compared to the preprocessing time of version 3. Since version 3 answers queries in constant time, As number of queries increase the running time increases linearly. Even at very large number of queries, the running time shows 0 milliseconds. As we can see from the graph above, the preprocessing becomes worth at number of queries approximately greater than 30. Below that we can just use the linear version to get results faster.

Comparison of version 2 with version 4



This is exactly the same as above, however the point at which preprocessing becomes worth is increased to around 85 queries. This is probably because the amount of benefit gained due to parallelism of query finding (version 2) is greater than the benefit of parallelizing the preprocessing of version 4. This makes sense as multiple linear traversals are being parallelized in the former case.