Josh Brown

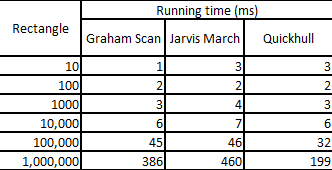
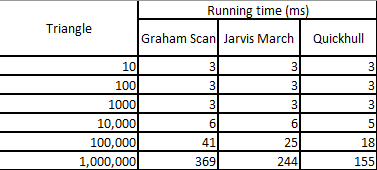
Dr. Duan

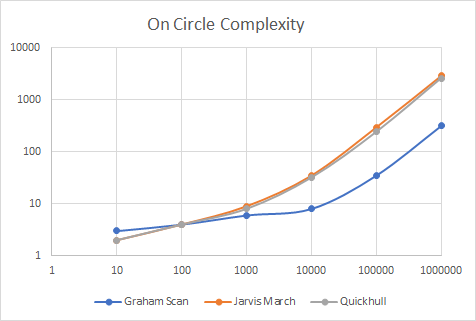
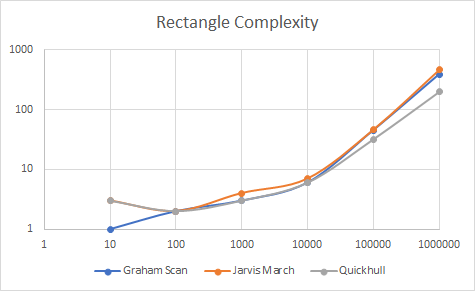
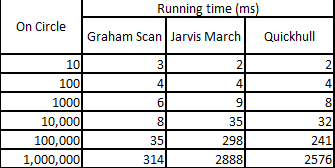
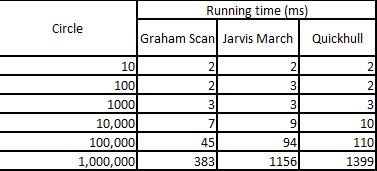
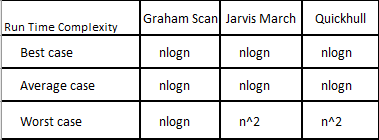
Algorithms

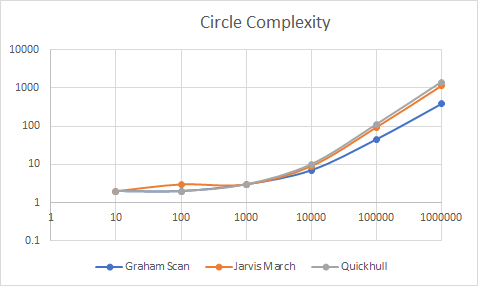
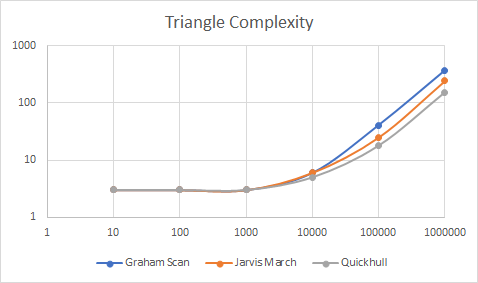
3/26/21

Convex Hull Report

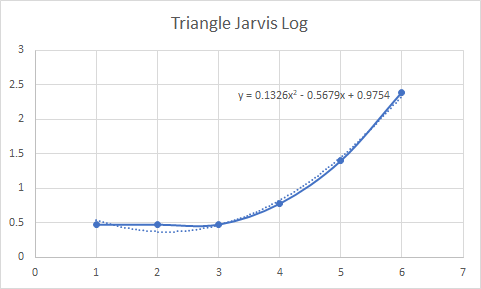
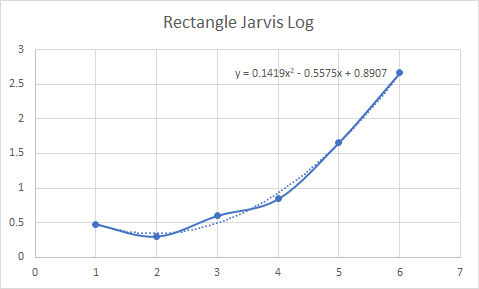
This report is a quick analysis of the results gathered from my program. First, we will examine the raw data, then see how the different algorithms compare to one another, and finally we will look at the Jarvis algorithm in Log form.

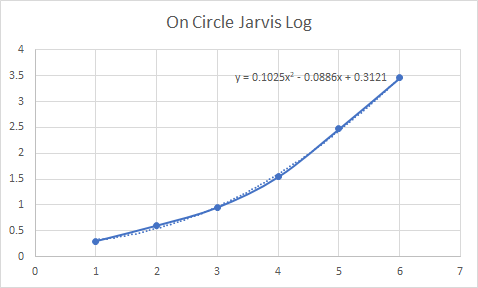
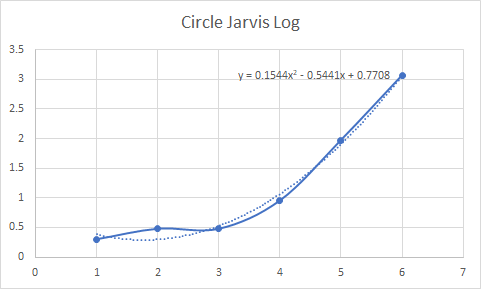
To begin, here is an overview of the raw data:

As we can see, the triangle and rectangle problems are fairly easy for each of the algorithms to solve. Circle and onCircle on the other hand, are not as easy for Jarvis and Quickhull to solve, but Graham Scan still takes no issue with these. This is most likely because of how Graham Scan utilizes its ability to sort all the points before it begins to test them, where Jarvis and Quickhull do not sort the points. These results also perfectly reflect our expectations on runtime complexity, shown in this chart. As we can see, we expect the worst case for Jarvis and Quickhull to be n^2, whereas Graham we expect to be nlogn. In our data, we can see that when thrown a harder problem, like the circles, Jarvis and Quickhull both become n^2 while Graham stays nlogn. This matches our theoretical analysis of the problem. Next, is the log-log graphs of the raw data:



The main point of interest in these graphs is one; they are very similar, showing that each algorithm has very comparable speeds, with the caveat of complex problems. We can see visually, as we described earlier, how Graham Scan beats both other algorithms when it comes to the circle problem. Inversely, when the problem is not complex, like Triangle, we can see that Graham is just a little slower than the other two algorithms.

Finally, we can see from the following graphs, the curve fits for each equation.

Here, we can see that jarvis is fairly similar throughout each problem, as we expected, with the exception being onCircle and Circle. Both of these have a steeper curve, more closely resembling n^2, while Triangle and Rectangle both appear to be closer to nlogn.

In all, each algorithm performs very well, but Graham performs best under difficult situations. Quickhull could be preferred if one expects to be working with simple problems. I think that our empirical analysis greatly matches our theoretical analysis.