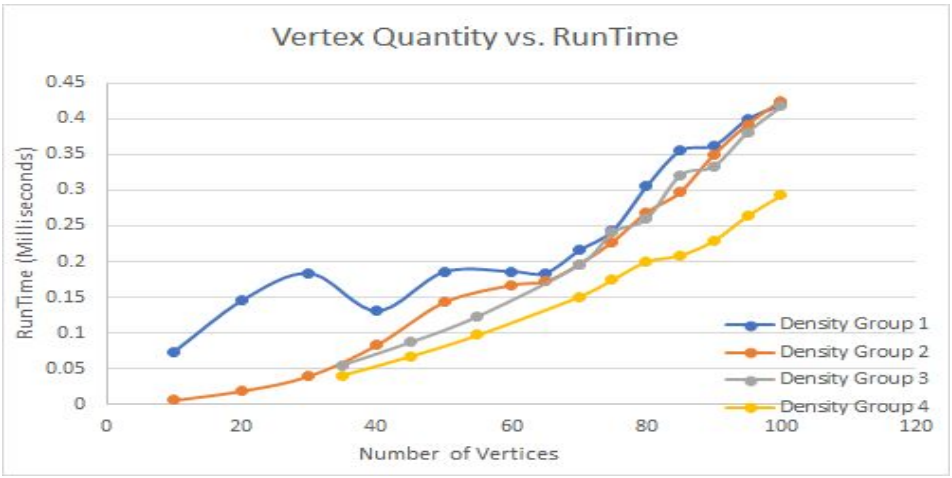
Dsatur Algorithm Analysis

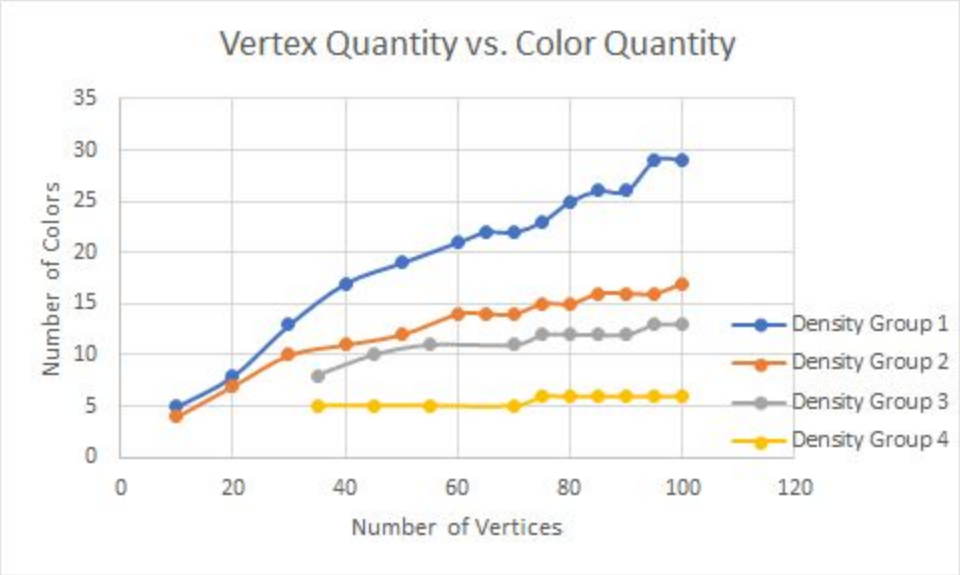
1. In order to reduce the time complexity of this algorithm, the data structure that was used was a HashSet. In my implementation of the algorithm, the HashSet data structure was used in order to update the saturation degree of each adjacent vertex in constant time and then using the size of the hashset in order to make any necessary comparisons between other vertices in the graph as well as updating the saturation degree of the vertex if necessary. Without using a hashset, the algorithm would have to not only iterate through each adjacent vertex, but it would also have to iterate through each vertex again to ensure that the colors between each vertex i.e the color of the vertices are distinct. This adds the E complexity to the algorithm. Additionally, a hashset was used in order to compute the minimum color of the selected vertex based on the colors of the adjacent vertices. I also used a hashset to evaluate the number of distinct colors that were used in the graph. Computing the minimum color of the selected vertex would require an iteration through all of the edges within the subgraph because of the necessary comparison between two edges. However, a hashset eliminates this overhead because a hashset can add the colors of each adjacent vertex and eliminates duplicates through its contains method which operates in constant time. Likewise, using a hashset to compute the number of distinct colors within the graph has an amortized cost of O(1). No such comparisons between 2 vertices for each distinct edge is necessary because the selected color for each vertex is added into the hashset, so the number of distinct colors used can be extracted through the size of the hashset which is done in constant time. Thus, my implementation of the algorithm operates in $\Theta(V^2)$ time complexity because only an iteration through every vertex in addition to updating the saturation value of each vertex and selecting the vertex with maximal saturation degree within the loop is the only cost within this algorithm. Iterating through edges is unnecessary.

2. 

The data presented in this graph suggests that there is a pronounced relationship between the runtime and the number of vertices present in the graph. There is also a

relationship between the density of the graph and the runtime as well in which greatly increasing the number of edges does have an impact on the runtime of the algorithm. However, this was a marginal increase compared to increasing the number of vertices, but the effect is more profound at the lowest density group where its runtime is noticeably lower in all subsets whereas the other density groups seem to converge where the number of vertices is 100. The relationship looks more linear than expected, but this can be attributed to a variety of factors. The worst case time complexity is $\Theta(V^2)$ and the worst case isn't being reached nearly as much as the size of the colorless subgraph decreases as well as the implementation of my algorithm that uses data structures in a way that speeds the runtime of the algorithm considerably.

3.



According to the data, the relationship between the number of vertices in the graph and the estimated minimum number colors within the graph is that the minimum number of colors increases in proportion to not only the size of the graph, but also the density of the graph. My data concludes that with a limited density, the minimum number of colors hardly increases in proportion to the size of the graph as well. This can be inferred because there is a significantly low likelihood that edges will be added as the size increases. Conversely, as the density increases, size has a significantly greater impact on the number of colors required to color the graph. Because the density of the graph is significantly high, there is a significantly greater likelihood that edges would be added to the graph as the size of the graph increases. This is illustrated in my graph in which the highest density group has a linear relationship. However, as we shift towards the lower density groups, the slope approaches 0 which is evident by the lowest density group. Thus, we can conclude that the impact that the size of the graph has on the minimum number of colors is contingent upon the density of the graph. Lower density means that the size of the graph has a significantly lower effect and vice versa with higher density.