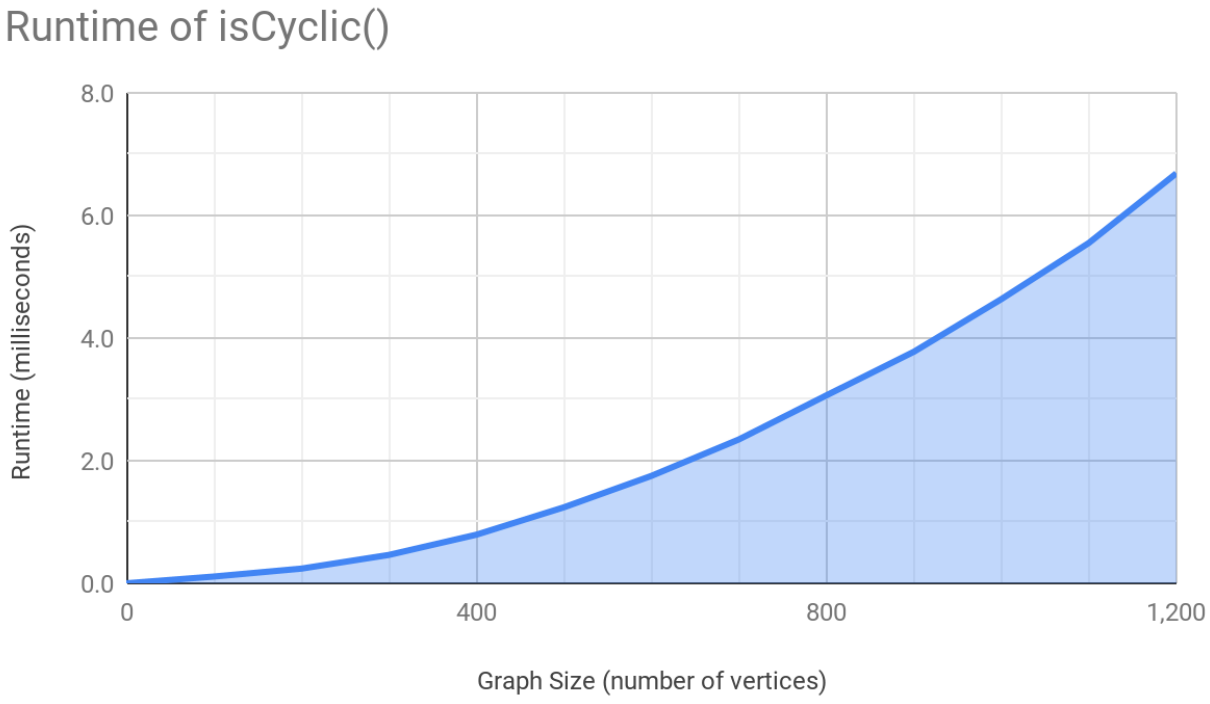
Assignment 7: Graph Utilities

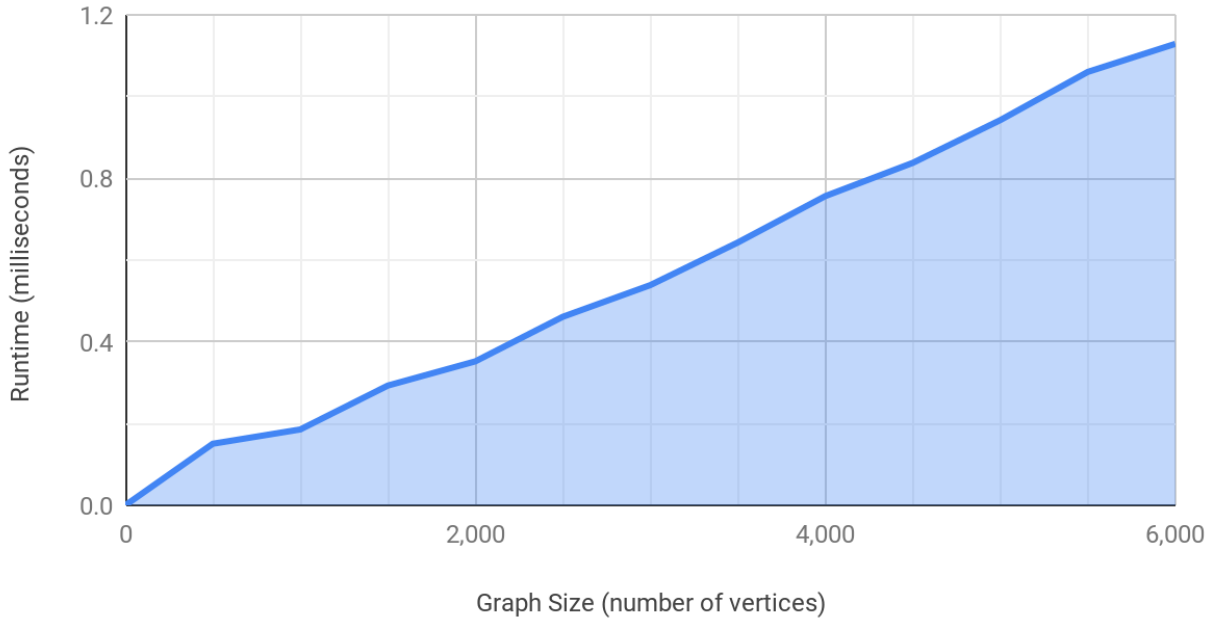
Analysis Document by Qianlang Chen (u1172983)

In this programming assignment, and for the first time in the semester, I had Kevin Song as my programming partner. Kevin is an experienced programmer, which made us work so efficiently that we finished the main part of the assignment in under 3 hours. I am planning to work with him again in future assignments.

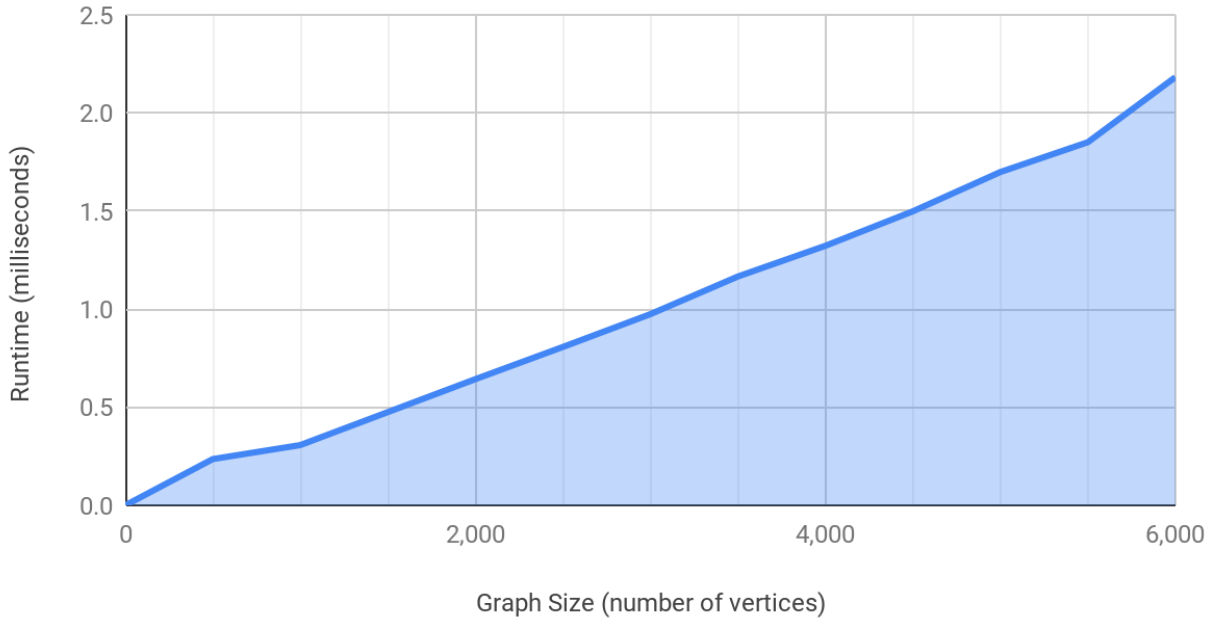
Our class contained three main tool methods: *isCyclic()*, *areConnected()*, and *sort()*. Since this was the first time we worked with graphs instead of linear data such as arrays and linked-lists, we wanted to figure the runtime behavior of our class. We understood that the runtime of a graph was dependent on both the number of vertices and the number of edges (and of course, how they are laid out and connected to each other), and since it was hard to control both of the two variables at the same time and observe their overall relationship with the runtime, we decided to **fix the number of edges to always be 2 times the number of vertices**, therefore the number of edges is linear to the number of vertices. We timed our methods with different numbers of vertices, and here are our results:



Runtime of areConnected()

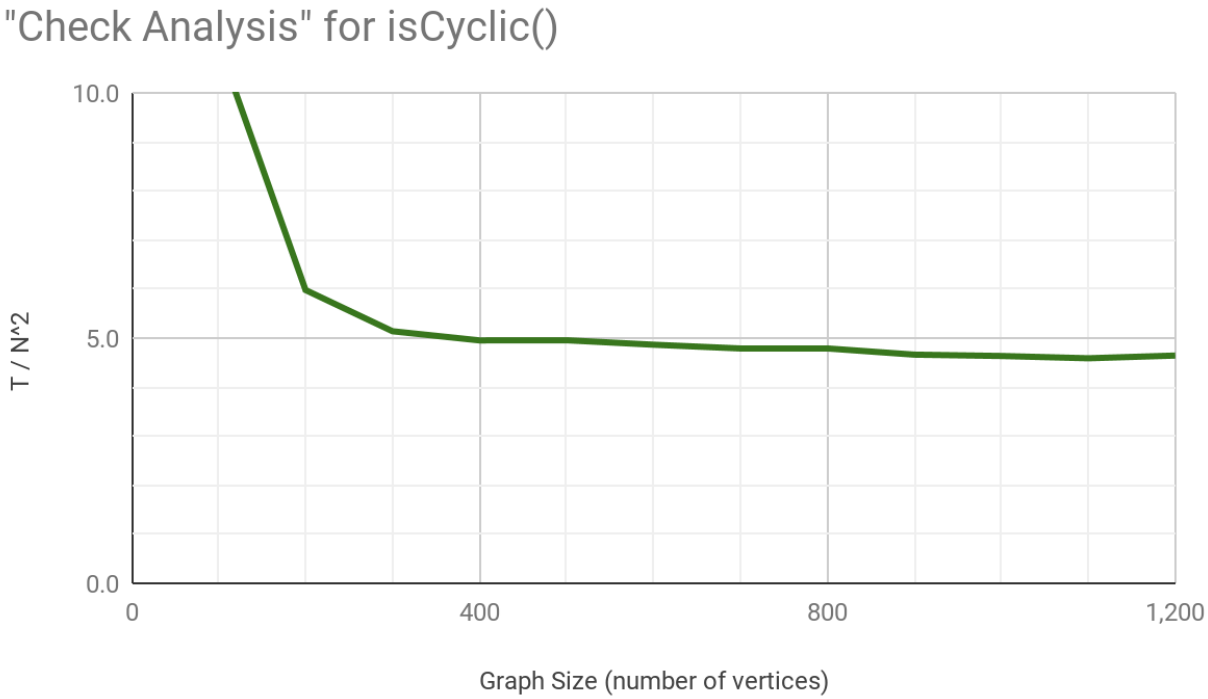


Runtime of sort()



From our results, we saw easily that the *areConnected()* method and *sort()* method both have a **linear** runtime behavior. Our *sort()* method is a Topological Sort, and the mechanism if it suggests that it visits each vertex and each edge in graph **exactly once**, so it makes sense to have a runtime corresponds to both the number of vertices and edges and such linear behavior (assuming the graph is acyclic). The *areConnected()* method uses a Breadth-First Search to find if there exists a path between a given source and a destination, whose runtime also depends on the number of vertices and number of edges; even though this method does not necessarily visit every single vertex and edge every time it runs, the number of operations still depends on the overall number of vertices and edges. Therefore, *areConnected()* method also has a linear runtime behavior.

Our *isCyclic()* method, on the other hand, somehow shows a **quadratic** runtime behavior. This method uses a Depth-First Search to find if there exists a cycle with a given starting vertex, and the algorithm itself is linear to the number of edges since it visits every edge exactly once during the searching process for a single starting vertex. However, the graph might be disconnected and contain some separate groups of vertices, so this method would have to choose multiple starting vertex to perform the Depth-First Search. The number of starting vertex depends on the number of vertices, therefore it has a runtime that corresponds to **the number of vertices times the total number of vertices plus edges**. Finally, to ensure that the behavior is quadratic, we used the “Check Analysis” technique on our results of timing the *isCyclic()* method by taking the runtimes divided the number of vertices squared:



The result seems to converge to a constant, which verifies the runtime behavior of our *isCyclic()* method being quadratic.