CSC263, Problem Set 3 Q3

Youngsin Ryoo, [redacted]

Question 3

a)
1. Remove a vertex and all its incident edges. Store this vertex and edges. Decrease the number of vertices by one, and set this as a new $|V|$.
2. Run BFS on the graph. If the number of all reachable vertices from any vertex is $|V|$, the graph is connected. The program return the vertex that had been deleted. If the number of all reachable vertices is not $|V|$, proceed to the next step.
3. Add the deleted vertex and edges stored back to the graph. Increase the number of vertices by one. Repeat step 1 and step 2 until the program find a vertex that does not disconnect the graph.

b) The number of vertices will decrease by one since a vertex has been removed. When the program runs BFS, the number of reachable vertices from any vertex would be equal to the number of vertices. To be specific, BFS checks whether the vertices are visited as it runs. If all the vertices are visited, that means the graph is connected. Since the program store the removed edges and vertex, if the graph is not connected, the program should add back those removed edges and vertex, and instead remove another vertex and repeat BFS to check whether the graph is connected or not. If the program find a vertex that does not disconnect the graph (which means when BFS runs all the vertices are visited) the program should return that vertex. Otherwise, we will repeat the process until we find one. If no vertex does not disconnect the graph, the program shall return False. To sum up, our algorithm only returns a vertex when it finds a vertex that does not disconnect the graph by running BFS and checks whether all the vertices are visited. Otherwise, it will return False as that means every vertex will disconnect graph when they get removed.

c) The worse case running time of my algorithm is $\mathcal{O}(V^2)$. This is because in the worst case we will have to check every vertex in the graph. When the program removes a vertex, the rows in the adjacency matrix will be moved to the left and columns will be moved up to represent the graph without the deleted vertex and edges to restructure the matrix, traversing all pairs of vertices. This will take $\mathcal{O}(V^2)$. Adding the vertex and edges back to the graph will take $\mathcal{O}(V^2)$ because all pairs of vertices are traversed in the same manner as it does when a vertex is removed. When BFS runs, it will check if the vertex is visited, for every vertex. This will take $\mathcal{O}(V^2)$. Therefore, the worst case running time is $\mathcal{O}(V^2) + \mathcal{O}(V^2) + \mathcal{O}(V^2)$, which is $\mathcal{O}(V^2)$.