Report

1) Go through every vertex, check if it has a path to ‘i’ if the other vertex has that path too, then i is an ancestor.
Find shortest path by repeating above process.
return length of said path, -1 if not found.

2)The vertices that appeared the most were 1-6, 1-5, 8-9 and the least were 7-8, 1-4 and 6-7. What makes an edge appear more frequently on MSTs is if this edge is connecting a larger amount dense subgraphs.

3)

4)The complexity is O(ElogV) (times 50) since the most expensive operation is getting the minimum spanning trees using kruskal’s algorithm.

5) Highest closeness centrality: 7, 8, 2.
Lowest closeness centrality: 0, 13, 12.

Highest betweenness centrality: 7, 8, 0.
Lowest betweenness centrality: 12, 11, 10.

Betweenness centrality indicates how central a vertex is in a graph or more exactly how many times this vertex is included in other shortest paths among the other vertices on a graph. Intuitively it measures how “influential” a vertex is to a network.

Closeness centrality indicates how close this vertex is to the rest of the graph by finding the smallest distance between it and the other vertices. Intuitively it measures how fast information will spread to the rest of the graph if it were placed on a certain vertex.

6) The overall complexity for task 3 is O(V^4) since for every vertex to every vertex the shortest path has to be computed, and since I use the DijkstraSP, its worst case performance is v^2 so it adds up to v^4.