Task 1:

a) We create a 2D array by using the number of vertices +1. The array becomes a vxv matrix. We iterate through the edges and insert them into the 2D array accordingly.

b) We create are dictionary to store the vertices in. and each vertices has an empty list in attached to it We ~~his~~ edges and connect the attached adjacent node and its weight in a tuple with that's list.

2. Task-2.

We create an adjacency list and by following the psudocode of bfs we traverse the graph.

Task 3:

We create an adjacency list and by following the psudocode of the ~~bfs~~ dfs we traverse the graph.

Task 4:

Along with the visited array in dfs we create another array named in_stack to keep all the ~~node~~ vertices that are currently being explored. While traversing through the graph by using dfs if we encounter ~~two~~ the same ~~vertice~~ vertex that's already in stack then we can say that the graph has ~~one~~ cycle.

Task 5.

while traversing through the graph with using bfs we create an another array naming previous_city. We insert the previous city visited of some vertex in that array of that. And by following up through those previous cities we can find the shortest path and time also.

Task 6:

We create and store the graph in an adjacency matrix. We create a checking function so that we don't go out of bounds of the matrix or visit a node that has already been visited. We iterate through our all the dot waypoints and check theirs adjacent position for diamonds. if found we add them all up.