Task01

a I created a 2D array and put the weight whe-
re there is an edge between two vertex.

b I created a dictionary where the keys are
vertex from 0 to N. Then, store the last node
and weight as list of tuples as the values of
those keys.

Task02

I created the adjacency list and visited the graph
level wise. I inserted the starting node in the queue
and update in state array as visited. Then I
run a loop while the queue is not empty. Pop
element from queue and append it to the path.
After that, I run a for loop to enqueue the
neighbours of the visiting node if the neighbou-
rs are not visited. ~~While the~~ Once the queue is
empty I returned the path and print.

Task3
I visited the graph from source to depth. I run a for loop and recursively visited all the nodes from source to the last node which is connected with the source. ~~Then~~ In this way, I recursively visited all the nodes which are not visited

Task4
I use DFS algorithm to traverse the graph. and maintained a stack to keep track of the visited nodes and checks for cycles by looking for a back edge. If a back edge is found, it indicates the presence of a cycle. If any cycle is found, I return "Yes" else return "No".

Task5
I use BFS algorithm to find the shortest path from city 1 to D and maintained a queue to store the cities and paths. Whenever, I go from 1 city to another city, I continuously update the path and mark the visited cities until I found the destination city. Once, I found the destination, I

returned the shortest path taken. If there exists no path then I returned an empty list.

Task 6

Here, I considered each cell as a potential starting point and recursively visits adjacent cells and collect diamonds along the way by avoiding obstacles. The count is updated to keep track of the maximum number of diamonds collected throughout the traversal. Finally, I returned the maximum number of diamonds which are possible to collect.