1A

first we will take an empty list to hold the 2D array. Now we will iterate n+1 time where n is rows in array. the loop will initialize each row with n+1 columns. where it initialize all elements to None or 0. Now we will create a list name row. and append the arr 2D lists.

1B

first we will create a dictionary. The keys of the dictionary will be node numbers from 0 to N. and the values will be lists containing tuples representing the neighbouring nodes and their corresponding edge weights.

2

Here we will do BFS. We will initialize a queue and an auxiliary array name visited. We will also take a list name as bfs_result. Now we will run a while loop. and we will pop (0) and append it in bfs results. Now we will run a loop and will check if the vertex is visited or not if not

we will append in queue.

Task 03

In this we will use dfs. first we will initialize a dist name graph. then we will initialize a auxaliary array visited and a stack. Now we will do a while loop where A and a for loop where we will check if it's in the graph and if it's not visited we will append it to stack.

Task-04

It is similar to task 3. we just have to check an if condition extra here. If dfs_result is equal to list(1, num_cities+1) It will print "Yes" else it will print "No".

Task-05

Here we will initialize a queue [1,[],0]. if the Node is equal to the destinated city the shortest path will be path + the index of that Node. and it will break.

Task-06

We will intialize a dict g. A and append row in it. We will create a 2D array as name dirs. Here we will check if g[r][c] = 'D' or not if its not visited we will add it to stack. We will ~~initate~~ intialize a variable named as ~~max~~ max_diamonds and diamonds. ~~to~~ here it will increment if g[r][c] >= 'D'. After this we will exchage it and return maximum diamonds