Razin Safian

ID: 22301219

sec: 16

course: CSE 221

LAB Assignment 4

code explanation

Task 6:

This code implements food fill to explore reachable cells without obstacles. It traverses the grid, counts diamonds encountered in connected regions by recursively visiting adjacent cells ensuring valid moves while avoiding obstacles and tracks the maximum count of collected diamonds.

food_fill_fun:

- it iterates through each cells, checking if it's a valid move (not an obstacle and not visited)
- Recursively explores adjacent cells (up, down, left, right) and counts diamonds while avoiding obstacles.
- Marks visited cells to avoid revisiting

and ensures valid movement

Task 5:

here shortest-path function:

- Uses (BFS) approach to find the shortest path in an undirected graph
  - Takes in the graph, starting vertex, and destination vertex.
  - Utilizes a queue ("deque") to perform BFS traversal
  - Tracks visited vertices to avoid revisiting them
  - Continues BFS until the destination

vertex "D" is found or all reachable vertices are explored.

- Returns the shortest path length (number of edges) and the path takes (as a list vertices).

Task 4:

has_cycle function:

- Utilizes DFS traversal to detect cycles in a graph.
- Marks vertices as visited and keeps a recursion stack to track the path.
- Traverses each vertex's neighbours:
  - if a neighbour is not visited, recursively check for a cycle.
  - if the neighbour is visited and is in the recursion stack, a cycle is detected.

. Updates the recursion stack before returning the result.

2. "Cordin's-cycle" function:

. Creates an adjacent list representation of the directed graph.

. initializes "visited" and "recursion_stack" dictionaries to visited vertices and their presence in the recursion stack

. Initials DFS traversal from each vertex in the graph if it hasn't been visited

yed.

- if a cycle is detected using "has_cycle" during the DFS traversal, return "yes" indicating the presence of a cycle. otherwise, returns "no".

Task 2, 3:

task 2 is the BFS. in BFS we ~~use a queue~~ maintain a queue and in DFS we maintain a stack to seach it's respective neighbour vertexes.

we also use a visited
variable list to ensure
that we dont visit any
node two times.

these psudo codes are given in
the questions.

Task 1 & 2:

Here we are representing
a directed weighted graph by
a matris and a dictionary.

in task 1 I have created a square matrix of size N x N;

N = number of vertices. Then just I input the connections according to row and colom.

In task 2; I create a key for (0 → N) vertices. and add the values in their respective keys.