Idea -

- To find the best possible shortest path without collision for each drone , we need to check all the possible paths from source to destination.
- To check all the possible paths we are going to use DFS.
- During DFS , three possibilities will occur
  - Neighbour node of curr node is visited ; visited[x][y] = 1 .
  - Neighbour node of curr node is visited and occupied ; visited[x][y] = 2 .
  - Neighbour node of curr node is unvisited and unoccupied ; visited[x][y] = 0.

Approach -

- Make a list of lists in which each index represents a list which consists of starting x and y coordinates , ending x and y coordinates and starting time of a drone .
- Traverse through the list of list , for every drone call helper function which takes list of current drone(listdrones[i]) while iterating through a list of list(listdrones), x starting coordinate, y starting coordinate, visited list, result list which stores all the paths of all drones and curr list which stores path of current drone and helper function returns true when it finds the destination coordinates along with appending the curr list into result.
- Helper function marks the current occupied node as 2 in the visited array and if in any of the neighboring four positions of the current node within the matrix or grid and is unvisited (visited[x][y] == 0), then take this node in path and we also take a case when visited[x][y] == 1 to avoid get in infinite loop.
- Call the helper function until we didn't find the destination coordinates.
- After iterating through all the drones , print the paths of all drones by printing the result list.