Question chosen: 1

Team:

Konda Jnanasree	19IM10016
Ankita Radhakrishnan	19IM30005
Pratham Narwade	19IM30014
Penukonda Yeshwanth	19IM30015

1. For the optimal solution, we tried to apply the CBS (Conflict Based Search algorithm).

Paper referred to (for the theoretical understanding of the algorithm): Overview of Multi-Agent Path Finding (MAPF)
Wolfgang Hönig, Jiaoyang Li, Sven Koenig — University of Southern California
Model AI 2020 Assignments: A Project on Multi-Agent Path Finding (MAPF)

Thus, we tried to implement the following logic in our code:

1. Generate a matrix of rows and columns as per the values input by the user. Input k, the number of robots and the start and goal points from the user. Check that the start and goal points fall within the points allowed by the dimensions of the matrix (between 0 and rows - 1 or columns - 1 as required).
2. Create two classes - one for storing the coordinates of the current positions of the robots (which does not take the time into consideration) and the second to store the state and location of the robots (where location.x and location.y will denote the respective coordinates)
3. Create a class Conflicts which initializes the conflicts to zero at the beginning, we can then add the conflicts as and when we encounter them in the code. We need to classify the conflicts as either edge conflicts and vertex conflicts.
4. Create a class each for vertex constraints and for edge constraints. With the vertex constraints, we will try to see if at the same time two robots occupy the same location. With the

edge constraints, our aim is to see whether two robots are at the same two consecutive locations over two consecutive time instances.

5. In this algorithm, we understood that searching is done at two stages. At the first stage, we take the start and goal nodes of each robot, and then appy A* search to it. With the help of this, we are able to generate a minimum path for the robot each time. However, this does not take into account any collisions which can happen. Thus, we need to perform search at a higher level. This searches for possible collisions among the generated paths of the robots, and tries to classify these as vertex and edge constraints.
6. For this purpose, we have defined a class called Constraints in the beginning. This takes into account the vertex and the edge constraints, and the two robots which bring about this conflict.
7. In the case of the A* algorithm, we try to find the next neighbour for each node from start, and we try to choose that with the lowest cost. Thus, here, we try to check if any of the points surrounding a particular point or that point itself are valid at the next time interval.
8. Thus, we have come to the concept of a valid state and a valid transition. As per the constraints of the question, a state is said to be valid, if:
   - If falls within the dimensions of the matrix
   - It is not occupied by another robot
   - If the state is a temporary storage location, or the start or goal state of any of the robots under question
   - If it does not come under any of the obstacle coordinates

   And a transition is said to be valid, if two robots do not have to traverse along the same edge in the same consecutive time steps.

9. After finding the collisions, we try to convert them into constraints, and the algorithm is applied recursively until we arrive at a solution
10. For this purpose, an open and a closed list is maintained. Here, we try to store the minimum of the computed solutions till now, and then keep building from there. Once the open list is empty, and we can return the solutions. And we keep on removing robots from the list once no collision is found for it.

11. From the solutions, we are trying to find the concerned path, and print the cost and the coordinates of the robot at each time step.
12. Since the CBS algorithm guarantees a success for its optimal solution, if this method does not yield any solution, then it implies than an optimal solution is not feasible.