MULTIAGENT PATH FINDING

AIFA-ASSIGNMENT 1

WORK DONE BY:
1) Suparnakanti Das, Roll No: 20AI92S01

2) Ajay Tanala, Roll No 17AE30004

CENTRE OF EXCELLENCE IN ARTIFICIAL INTELLIGENCE

IIT Kharagpur,INDIA

PROBLEM STATEMENT:

Consider a warehouse management system by a group of robots. A group of robots need to pick up items from designated places, deliver those in desired locations, and finally the robots go to their respective destination location. This can be seen as a multiagent system with shared tasks as objective. Assume that the entire planning needs to be done in a 2D grid representation of size n × m. Each location (Li) on the warehouse floor can be denoted by a pair of integers Li=(xi, yi). Each cell on the grid can be categorized in one of the following ways (see diagram below) - source location (P1, P2 ...), destination location (D1,D2,...), temporary storage location (TS1,TS2,....), obstacle (black square),normal (rest of the cells). Source & destination denote pick-up and drop locations respectively.

Temporary storage location denotes the place where robots can keep some items. Obstacles represent a location where no robot can move. Rest of the cells are considered normal cells.

Let there be k number of robots and r number of tasks. The details of robot location and tasks are provided as per the following table.
Let's take k=4 robots and r= 4 tasks for example. Have a look at it.

ROBOTS	LOCATION	
	Initial	Final
ROBOT1	R1	E1
ROBOT2	R2	E2
ROBOT3	R3	E2
ROBOT4	R4	E3

TASKS	LOCATION	
	Pickup	Deliver
Task1	P1	D1
Task2	P2	D3
Task3	P3	D3

Task4	P4 D2

Assume that a robot can move at most one cell (either vertically or horizontally) at a time step, a normal cell can be occupied by at most one robot. Source, destination, temporary storage locations can accommodate multiple robots simultaneously. Our target here is to develop a work schedule that minimizes the time to complete all tasks. You need to develop both optimal as well as heuristic algorithms.

Let's take a grid of size(m*n) and assign start locations and end locations of robot and assign pickup and delivery locations of a task according to problem.Assign storage stations.
Take any m and n values. I took 6*10 grid but you can take whatever you need.

R1			P2		TS2				
						E1			D3
		D3		P1				E3	
	R2		TS1						P4
	D2							D1	
R4		R3			E2	TS3		P3	

Here black coloured boxes are blocked cells where no robot can go through that cell. TS texted are storage stations where robots can store. Blue coloured are starts and end locations of robots and Red coloured are pick-up and end locations of tasks.

The following assumptions are considered while solving the path finding problem.
i. Each Agent can move a single step at a single unit of time.
ii. the static obstacles will not be moving.
Iii. Dynamic obstacles need to be handled by the agents.
Ii. The robots may use the temporary location not in all cases and will have prior knowledge about its usage.

Explanation of our Assignment:

In our source code, we have two input files that are input.yaml file and input.yaml file and a output file named output.yaml.
The code needs to be run like python3 ma.py input.yaml output.yaml

Algorithm: The code inherently uses A star algorithm for getting the best path with the given obstacle as specified in the input file. The algorithm uses python dictionary data structure along with a queue which takes the input and then generates output as optimal path in step by step manner, the output is written in the output file.

Installation and User Manual: python3 is required to be installed and the code needs to be run as python3 ma.py input.yaml output.yaml
As given in the input file specifications. The code can handle m*n size grid with any no of agents.

INPUT file:
It is the input file which we have to predefine the No.of Robots which are available to complete the all tasks. We have to predefine the Robot's start and end locations and Task's pickup and delivery locations.
For any agent let's say agent0, We have 4 tasks as following
In this our format in INPUT file:

- goal: [7, 2]
  name: agent0
  start: [2, 5]
- goal: [3, 3]
  name: agent0
  start: [7, 2]
- goal: [1, 2]
  name: agent0
  start: [3, 3]
- goal: [1, 4]
  name: agent0
  start: [1, 2]

It means agent0 starts from the start position[2,5] and reaches to pickup position[7,2]. From pickuplocation it goes to TS1 location [3,3]. From TS1 location it goes to drop location[1,2]. Finally it reaches to its final location [1,4]. Like this, we have the following four tasks as mentioned below.
Task1: robot start position to pick-up location.
Task2: pickup location to storage station.
Task3: storage station to delivery location
Task4:delivery location to final location of robot.

This way it any number of robots can be added.

In this file, we have other inputs which are obstacle cells is marked as black in the previous diagram.

In our input file we just take two robots but we can take any number of robots we want. We can make our desired grid size and can make any number of obstacles.Care must

be taken in assigning start location,end location,pickup location,end location as these can’t be obstacle cells. This is our input file.

OUTPUT FILE:

In the output file, Every agent starts from their starting position at time t=0 secs to complete its activities specified in input file. Output is in the format that at time t,x and y coordinates of a robot are printed as below. After completion of Task1 by all robots, Task2 will be initiated at time t=x where x is a time taken by the corresponding agent to complete its previous task i.e, Task1 here. Like this, by completion of all tasks by all robots, we will get time taken by a robot to complete all of his tasks.

```
agent0:
- t: 0
  x: 2
  y: 5
- t: 1
  x: 3
  y: 5
- t: 2
  x: 4
  y: 5
- t: 3
  x: 5
  y: 5
- t: 4
  x: 5
  y: 4
- t: 5
  x: 5
  y: 3
- t: 6
  x: 6
  y: 3
- t: 7
  x: 7
  y: 3
- t: 8
  x: 7
  y: 2
agent0:
- t: 8
  x: 7
  y: 2
- t: 9
  x: 6
  y: 2
- t: 10
  x: 5
```

```
  y: 2
- t: 11
  x: 5
  y: 3
- t: 12
  x: 4
  y: 3
- t: 13
  x: 3
  y: 3
agent0:
- t: 13
  x: 3
  y: 3
- t: 14
  x: 3
  y: 2
- t: 15
  x: 2
  y: 2
- t: 16
  x: 1
  y: 2
agent0:
- t: 16
  x: 1
  y: 2
- t: 17
  x: 1
  y: 3
- t: 18
  x: 1
  y: 4
```

The above mentioned steps are the output of the for agent0. Similar output will be given for other agents as specified in the input file.