Artificial Intelligence Assignment № 1

Uğur Ali Kaplan, 150170042, Istanbul Technical University 24/11/2020

Running the Code

I have used Python 3.7.7 to implement the Q3. It needs Numpy.
If Numpy is installed, and all files are in the same directory (main.py, directions.py, bfs.py, dfs.py, a_star_1.py, a_star_2.py), it can be run from the commandline as:

Listing 1: Running Q3 Code

```
python bfs.py input.txt output.txt
python dfs.py input.txt output.txt
python a_star_1.py input.txt output.txt
python a_star_2.py input.txt output.txt
```

Question 1

Agent Type and PEAS Description

Agent Type	Performance Measure	Environment	Actuators	Sensors
a) Goal Based Agent - Must consider the state of the rest of the world, also must know what will happen because of its actions	Safe, efficient, undamaged goods	Air travel, birds, other drones, animals, people	Rotary wing	Gyroscope, camera, lidar
b) Simple Reflex Agent - Only needs to consider the current percept	Sleeping Baby	Daycare centers, baby rooms, bedrooms, hospitals, newborn unit	Speech, Audio, Speaker	Camera, Microphone
c) Model Based Reflex Agent - Must consider the state of the rest of the world	Correct detection	Footage of airport, passengers, planes	Display of anomolous activities	Camera, Audio Recorder
d) Simple Reflex Agent - Only needs to consider the current percept	Correct detection	Email, academic, business, personal, communication	Display of spam, placing spam into junk folder automatically	Text entry

Environment Description

a) i. Partially Observable $\implies$ Drone does not know everything required to make a decision.

ii. Multi-agent $\implies$ There are other agents in the environment that are trying to maximize their performance.

iii. Stochastic $\implies$ Drone does not necessarily know what exactly is going to happen after their move.

iv. Sequential $\implies$ Drone's future decisions depend on the prior decisions and prior information.

v. Dynamic $\implies$ Environment can change while drone is deciding what to do with the current information.

vi. Continuous $\implies$ Positions of elements in the space, time and everything else can take a value from continuous ranges.

b) i. Fully Observable $\implies$ Robot only needs to observe the baby and can understand if baby is sleeping or not.

ii. Single-agent $\implies$ There are no other agents trying to maximize their performance in the environment.

iii. Stochastic $\implies$ Environment state is stochastic, sometimes babies sleep in peace and sometimes they do not, no matter the conditions.

iv. Episodic $\implies$ Agent is only responsible of singing a lullaby after the current lullaby or a certain time period ends. It does not need to check the status of the baby constantly.

v. Static $\implies$ Baby does not sleep in the time period it is required to decide whether to sing or not.

vi. Discrete $\implies$ There are two discrete states, and those are: baby is sleeping, baby is awake.

c) i. Fully Observable $\implies$ If there are enough cameras in the airport, it can fully observe the environment, planes, and the passengers.

ii. Single-agent $\implies$ There are no other agents trying to maximize their performance, unless the model is an ensemble model.

iii. Stochastic $\implies$ Agent cannot be certain of the state of the environment in future time steps from the current information.

iv. Sequential $\implies$ Depending on the implementation of the agent, it could be sequential. For example, it might be considering the last 10 seconds, than it could be thought of as sequential.

v. Dynamic $\implies$ Environment is constantly changing and an anomaly can happen at any second. So, environment can change while agent is deciding.

vi. Continuous $\implies$ There are people, planes, luggage and more to keep track of and combination of these can considered as the state of the environment. Since there will be

many different objects in many different places in the environment, we can say environment is continuous.

d) i. Fully Observable $\implies$ It has the content of the mail, sender, time information and so on. Agent has everything it needs to make a decision.

ii. Single-agent $\implies$ There are no other agents trying to classify the spam or generate the spam, at least in the same environment as our agent runs.

iii. Stochastic $\implies$ From the emails so far, a spam classifier would not be able to predict if the next email is a spam or not.

iv. Episodic $\implies$ Each email can be considered separate if this classifier is not constantly trained with new information.

v. Static $\implies$ Contents of the mail cannot change when the agent is deciding.

vi. Discrete $\implies$ Environment can be in two states: spam or not spam.

Question 2

a) To have an admissible heuristic, we should be optimistic and do not overestimate. When we look at the graph, we see going to G from B costs us $1+6+2=9$. So, this can be a heuristic function if $0 \leq h(B) \leq 9$.

b) In this case, we want our heuristic to be less than or equal to the cost of going to the goal node. So, following properties should hold:

$$h(S) \leq c(S,B) + h(B)$$

$$h(B) \leq c(B,C) + h(C)$$

When we fill in the necessary values:

$$6 \leq 1 + h(B) \implies 5 \leq h(B)$$

$$h(B) \leq 1 + 5 \implies h(B) \leq 6$$

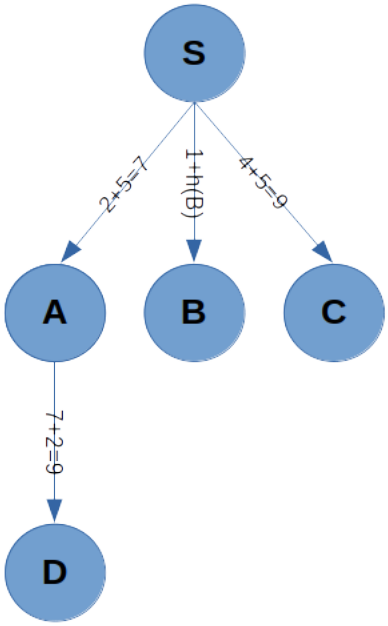
If we want our heuristic to be consistent,

$$5 \leq h(B) \leq 6$$

c) Since this is A*, our evaluation function is

$$f(n) = g(n) + h(n)$$

Where $g(n)$ is the step costs. If we first expand S, and then A, this is how our search will look like:



If we preferred expanding A, this means cost of expanding A is less than the cost of expanding B and C according to our evaluator function $f(n)$. Therefore, we can say:

$$7 \leq 1 + h(B)$$

Then, if we are expanding B, this means cost of expanding B is less than expanding C and D according to $f(n)$. Therefore, we can say:

$$1 + h(B) \leq 9$$

When we use our findings, we get the following inequality:

$$7 \leq 1 + h(B) \leq 9$$

$$6 \leq h(B) \leq 8$$

In the question, it is said this the graph search version of A*, therefore our heuristic should be consistent. This means $5 \leq h(B) \leq 6$.

In the end, only the following value is possible for $h(B)$ if the given scenario in c occurs:

$$h(B) = 6$$

Problem 3

a) **State Definitions**: I have used agent positions as state definitions. For two states to be the same, same agents should be in the same position.

Actions: To generate new nodes, I create permutations of "UDLRP" which represent the moves each agent can make. Python's permutation and combination functions are a little weird, and I needed at least the same amount of movements as there are agents, so I have created a list consisting of 9 "U"s, 9 "D"s, 9 "L"s, 9 "R"s, and 9 "P"s. There are 9 of each of them because maximum number of agents is stated as 9 in the homework file.

After the generation of the states, we assess the positions of agents and check if there are any illegal moves. In the case of illegal moves, we eliminate the state.

I have defined step cost to be the amount of movement agents make. Since each agent is only able to move one unit, if n agents moved in a state, step cost would be n.

b) I have used "input_1.txt" to run and compare the algorithms. Since it is stated that we should compare the three algorithms in the same maze, I have also run the A* algorithm with two different tie breakers.

Listing 2: BFS Results

```
10 2
RR
RR
RR
RP
DL
PL
PL
PL
UL
RL

BFS Generated Nodes: 103
Number of Expanded Nodes: 51
Maximum number of nodes kept in memory: 88
Running Time: 0.273593305 s
```

Listing 3: DFS Results

```
38 2
LL
PR
PR
PR
PR
```

```
RP
RP
LL
RD
RU
LL
PR
LL
RL
RR
RR
PR
RP
DP
PL
UL
LL
LL
LL
LL
RP
RP
PR
RP
PR
RP
PR
DP
PL
UL
LL
RP
RP
DFS Generated Nodes: 214
Number of Expanded Nodes: 43
Maximum number of nodes kept in memory: 179
Running Time: 0.189486844 s
```

When we compare BFS and DFS, we see that BFS generated less nodes. This is probably because the solution was in an early level. If there were more levels, it would be possible for DFS to generate less nodes than the BFS. We see that BFS expanded more nodes than the DFS, but it kept fewer nodes in the memory. Also, running time of the DFS was better compared to the BFS. BFS has found an optimal solution, whereas DFS found "a solution". DFS does not need its result to be optimal, it just needs it to work.

Listing 4: A* with Manhattan Distance Tie Breaker

```
11 2
RP
PP
RR
RR
RD
RU
PL
PL
PL
PL
PL
A* Generated Nodes: 510
Number of Expanded Nodes: 135
Maximum number of nodes kept in memory: 377
Running Time: 1.680508096 s
```

Listing 5: A* with Manhattan Distance + State Level Tie Breaker

```
10 2
RR
RR
RP
RD
RU
PL
PL
PL
PL
PL
A* Generated Nodes: 447
Number of Expanded Nodes: 120
Maximum number of nodes kept in memory: 329
Running Time: 1.498204258 s
```

When we compare A* with the other two, we see that it generated and expanded more nodes. Its memory requirements are larger. It took longer for A* to find the solution. This is related to the fact that solution was easy to find, so an easy algorithm was more suitable to the task. For example, when we use BFS, DFS, and A* in the 3^{rd} maze, BFS and DFS are either unable to find the solution, due to memory constraints, or they take their time finding the result. However, A* is able to find a solution, optimal or not, relatively easy.

c) For this part, I have used A* algorithm on "input_3.txt". I have used two different tie-breaking

strategies:

1. **Heuristic Function**. When evaluation scores are the same ($f(s) = g(s) + h(s)$), I have prioritized the value of the heuristic function.
2. **Heuristic Function + State Level**. Under normal circumstances, we use (heuristic + step cost) to evaluate which state to expand. If they are the same, I am using (heuristic + state level). In $3.a$, I have explained the step cost as the number of agents moved in each successive stage. I define the level as how many time steps have elapsed since the beginning.

Since I want a consistent and admissible heuristic, I have picked the Manhattan Distance. Here are the results of A* in maze 3:

Listing 6: A* with Manhattan Distance Tie Breaker

```
7 4
RLLU
RUUU
RUPU
DPPR
DPPR
DULR
PPLP
A* Generated Nodes: 2338
Number of Expanded Nodes: 24
Maximum number of nodes kept in memory: 2316
Running Time: 13.596645207 s
```

Listing 7: A* with Manhattan Distance + State Level Tie Breaker

```
7 4
RLLU
RUUU
RUPU
DPPR
DPPR
DULR
PPLP
A* Generated Nodes: 1520
Number of Expanded Nodes: 13
Maximum number of nodes kept in memory: 1509
Running Time: 7.761803501 s
```

We see that when we use the heuristic + state level as the tie breaker, search ends faster. Less nodes are generated, less nodes are expanded, and less nodes are kept in the memory.

Neither of them were able to find the optimal result, because it is hard to select a good tie-breaker, and optimality of the A* might get compromised depending on the tie-breaker. In my experiments, I have seen that they are able to find the best solution sometimes. So, it might be good idea to run the function a couple of times and select the best result.