Q 1 Agents and Environments

Scenario 1 - Intelligent Monitor

P - The quality of video , the number of targets captured
E - The storehouse
A - The video recorder
S - Motion sensor

Features

- Fully observable - entire storehouse visible
- Single agent - (Only the monitor)
- Deterministic
- Episodic
- Static - Environment won't change
- Discrete - recording or not
- Known

Least powerful agent - Simple reflex agent
An action is triggered when the sensor is set off. A simple reflex agent is sufficient to accomplish this.

Scenario 2 - Deep sea drone

P - Accuracy of readings of salinity and temperature, successful transmission of data, Reached required depth
E - Sea, sea creatures
A - Electro-magnetic devices to record readings and maintain depth and data transmitting device
S -Sensors to detect salinity, temperature and to maintain depth

Features

- Partially observable - entire environment not known
- Single agent - (Only the drone assuming the creatures do not affect the drone)
- Stochastic - Next state is unpredictable
- Episodic - a one-shot action
- Dynamic - Environment changes
- Continuous - recordings are
- Unknown - The drone does not fully understand the environment

Least powerful agent - Model -based reflex agent
Since the world evolves, a simple reflex agent will not be sufficient, but a model based agent will be.

Scenario 3 - Tax Assistance

P - Ability to assess financial situation
E - Repository of forms, database of questions
A - Monitor directing them to forms
S -Questions

Features

- fully observable - entire environment known
- Single agent - (Only the tax assistance system)
- Deterministic - Next state is known
- Sequential - Qs are asked in an order based on response
- Static - Environment does not change
- Discrete - Q and ans
- Known

Least powerful agent - Simple reflex agent
An action is triggered based on the response of the users. A simple reflex agent is sufficient to accomplish this.

Scenario 4 - Intelligent Infrastructure Management

P - minimal power usage and fast user responses
E - online resources of organisation
A - Software to handle storage efficiently
S - Sensor to detect power usage and latency handler

Features

- Partially observable - entire environment not known as the number of machines allocated my suddenly change and cannot be predicted
- Single agent - (Only the software)
- Deterministic - Next state is known (action)
- Episodic - a one-shot action (particular action taken when an event occurs)
- Dynamic- machine usage can change anytime

- Continuous - Monitoring
- Known

Least powerful agent - Goal based agent
The environment keeps changing and hence a simple reflex model is not sufficient and our goal is to minimize power usage and hence goal based agent.

Scenario 5 - Robot Soccer Player

P - Accurate detection of teammate and ball
E - Soccer field, players, ball
A - Kicking, scoring, dribbling
S - Proximity sensor (to detect ball and goal and players)

Features

- Fully observable - entire field is observable
- Multi agent - All players
- Stochastic - Next state is unpredictable
- Sequential - Based on passes
- Dynamic - Environment changes based on possession
- Continuous - Previous state (ball possession) determines next
- Known - The field is the environment and in known

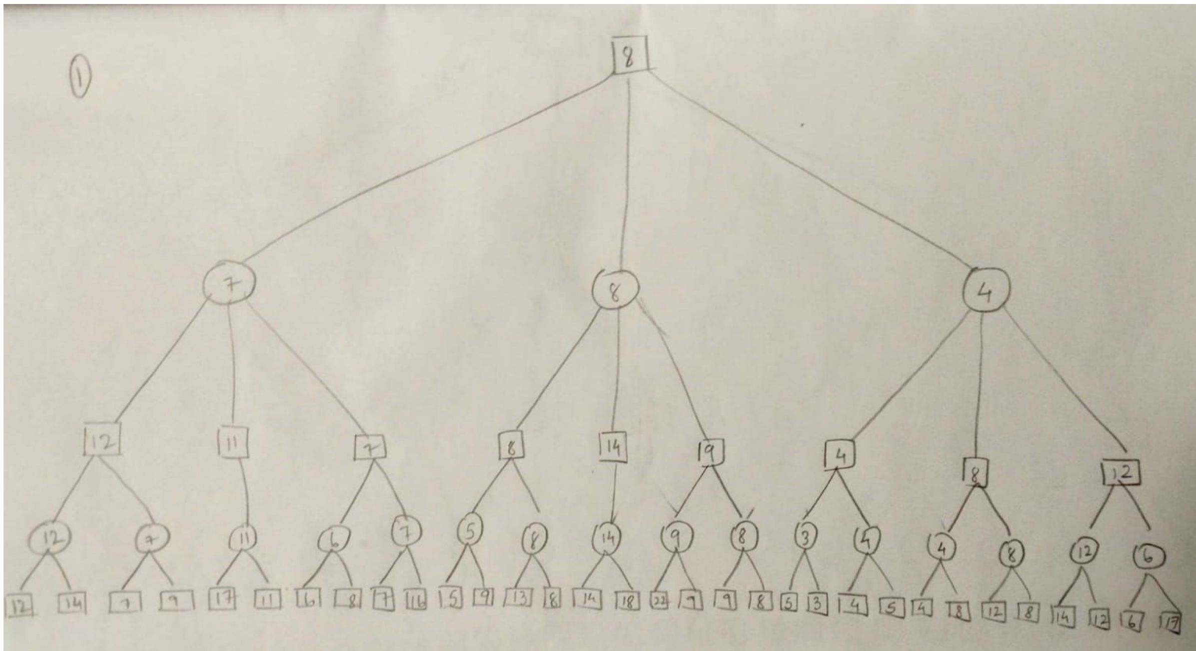
Least powerful agent - Goal-based reflex agent
Since the world evolves, a simple reflex agent will not be sufficient and since are goal is to score maximum goals in 90min, we can say that it is goal based

Question 2 Belief Maintenance

Present state -	Action -	Next state -	Belief state -
a	→	b	{a,b}
b	→	c	{a,b,c}
c	↑	f	{b,c,d,f}
f	←	g	{c,e,f,g}
g	←	h	{g,h,f}
h			{g,h,i}

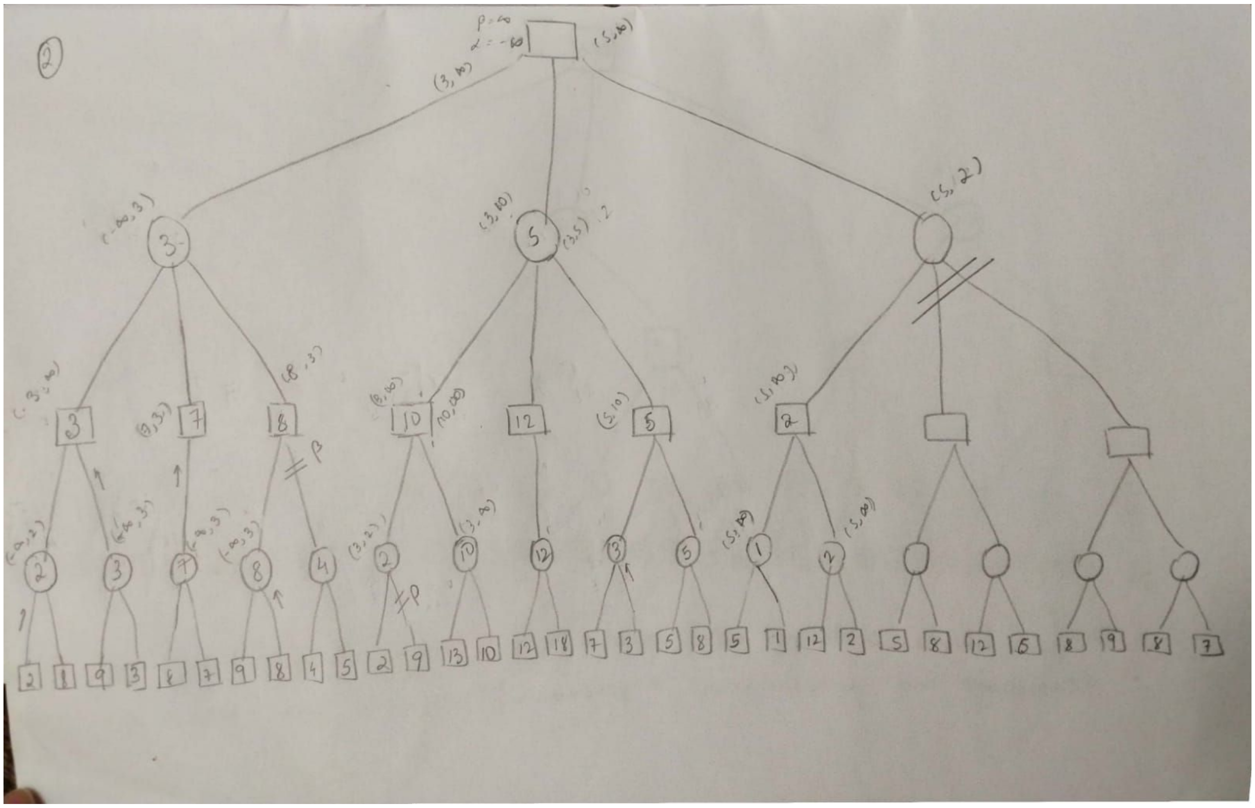
Question 3

1.

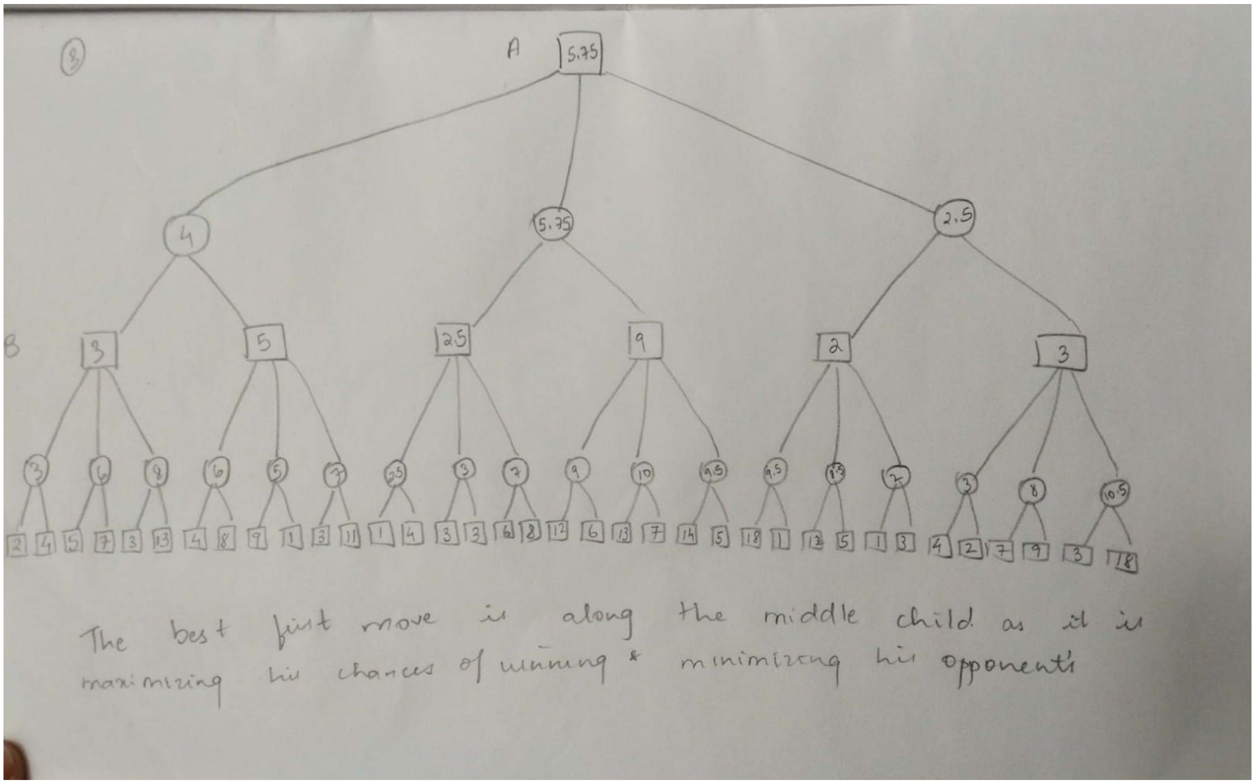


Best first move for max is the middle child node : 8

2.



3.



The best first move is along the middle child as it is maximizing his chances of winning & minimizing his opponents

Question 4 Search

1. Heuristic function -
   If the node is in the same row or column as the goal node, h(n)=1, it would take one hop to reach the goal node from the current node in the best case scenario
   If the node is not in the same row or column as the goal node , h(n) = 2 , meaning, it would take 2 hops to get to the goal from the current node in the best case scenario.
   It is admissible as the estimated cost is less than the real goal, and can also be consistent as the maximum h value will be less than the real cost to any location plus its' h value
   Limitation - It will underestimate a lot , i.e,I it will have a very large contour which will lead to a lot of excess work.

2.

4*4 grid

```
 ---  DFS  ---
Path -  [(0, 0), (2, 0), (1, 0), (3, 0), (3, 3), (3, 2), (0, 2), (0, 1), (1, 1), (2, 1), (3, 1)]
Number of states :  19

 ---  BFS  ---
Shortest path -  [(0, 0), (2, 0), (2, 1), (3, 1)]
Number of states :  6

 ---  Astar  ---
Shortest path -  [(0, 0), (2, 0), (2, 1), (3, 1)]
Number of states :  12

BestFirst
 ---  Greedy best first search  ---
Shortest path -  [(0, 0), (2, 0), (2, 1), (3, 1)]
Number of states :  7
```

6*6 grid

```
 ---  BFS  ---
Shortest path -  [(0, 0), (0, 5), (4, 5), (4, 1), (1, 1), (1, 4), (3, 4), (5, 4), (2, 4), (2, 0), (1, 0),
(1, 3), (5, 3), (5, 5)]
Number of states :  34
---

 ---  DFS  ---
Path -  [(0, 0), (5, 0), (3, 0), (3, 3), (0, 3), (2, 3), (2, 1), (2, 5), (0, 5), (4, 5), (4, 1), (1, 1),
(1, 4), (3, 4), (5, 4), (2, 4), (2, 0), (1, 0), (1, 3), (5, 3), (5, 5)]
Number of states : 39

 ---  Astar  ---
Shortest path -  [(0, 0), (0, 5), (4, 5), (4, 1), (1, 1), (1, 4), (3, 4), (5, 4), (2, 4), (2, 0), (1, 0),
(1, 3), (5, 3), (5, 5)]
Number of states :  12219
---
```

8*8 grid

```
 ---  BFS  ---
Shortest path -  [(0, 0), (0, 7), (0, 6), (0, 3), (6, 3), (6, 4), (6, 5), (4, 5)]
Number of states :  29
---

 ---  DFS  ---
Path -  [(0, 0), (7, 0), (7, 7), (0, 7), (1, 7), (2, 7), (2, 2), (0, 2), (5, 2), (5, 5), (3, 5), (3, 3),
(4, 3), (6, 3), (5, 3), (1, 3), (1, 2), (1, 0), (2, 0), (2, 4), (6, 4), (6, 5), (4, 5)]
Number of states :  49
---

Astar
 ---  Astar  ---
Shortest path -  [(0, 0), (0, 7), (0, 6), (0, 3), (6, 3), (6, 4), (6, 5), (4, 5)]
Number of states :  114
---

BestFirst
 ---  Greedy best first search  ---
Shortest path -  [(0, 0), (0, 7), (0, 6), (0, 3), (6, 3), (6, 4), (6, 5), (4, 5)]
Number of states :  64
```

12*12a

```
 ---  BFS  ---
Shortest path -  [(0, 0), (2, 0), (4, 0), (4, 1), (7, 1), (2, 1), (5, 1)]
Number of states :  56
---

 ---  DFS  ---
Path -  [(0, 0), (2, 0), (4, 0), (3, 0), (1, 0), (7, 0), (6, 0), (6, 6), (1, 6), (7, 6), (2, 6), (3, 6),
(4, 6), (0, 6), (0, 5), (4, 5), (10, 5), (10, 11), (5, 11), (2, 11), (2, 8), (8, 8), (3, 8), (4, 8), (10,
8), (9, 8), (7, 8), (5, 8), (5, 5), (1, 5), (1, 2), (4, 2), (0, 2), (6, 2), (8, 2), (5, 2), (2, 2), (7, 2),
(7, 1), (2, 1), (5, 1)]
Number of states :  55
---

Astar
 ---  Astar  ---
Shortest path -  [(0, 0), (2, 0), (4, 0), (4, 1), (7, 1), (2, 1), (5, 1)]
Number of states :  187
---

BestFirst
 ---  Greedy best first search  ---
Shortest path -  [(0, 0), (2, 0), (4, 0), (4, 1), (7, 1), (2, 1), (5, 1)]
Number of states :  17
```

12*12b

```
BFS
 ---  BFS  ---
Shortest path -  [(0, 0), (6, 0), (8, 0), (8, 8), (0, 8), (0, 11), (5, 11), (5, 4)]
Number of states :  69
---

DFS
 ---  DFS  ---
Path -  [(0, 0), (6, 0), (4, 0), (11, 0), (3, 0), (8, 0), (8, 8), (0, 8), (3, 8), (11, 8), (10, 8), (10, 4), (2,
4), (5, 4)]
Number of states :  18
---
```

Astar

```
 ---  Astar  ---
Shortest path -  [(0, 0), (6, 0), (6, 2), (10, 2), (10, 5), (10, 4), (2, 4), (5, 4)]
Number of states :  274
---
```

25*25

```
 ---  BFS  ---
Shortest path -  [(0, 0), (9, 0), (17, 0), (13, 0), (19, 0), (16, 0), (16, 6), (24, 6), (24, 7)]
Number of states :  358
---
```

DFS

```
 ---  DFS  ---
Path -  [(0, 0), (9, 0), (1, 0), (2, 0), (6, 0), (12, 0), (10, 0), (14, 0), (16, 0), (22, 0), (11, 0), (4, 0),
(4, 2), (13, 2), (9, 2), (7, 2), (17, 2), (6, 2), (2, 2), (8, 2), (8, 1), (18, 1), (12, 1), (4, 1), (2, 1), (7, 1),
(17, 1), (22, 1), (22, 11), (11, 11), (9, 11), (8, 11), (7, 11), (17, 11), (17, 21), (9, 21), (5, 21), (11,
21), (4, 21), (1, 21), (1, 18), (11, 18), (4, 18), (7, 18), (10, 18), (5, 18), (2, 18), (0, 18), (9, 18),
(17, 18), (6, 18), (6, 13), (3, 13), (10, 13), (9, 13), (0, 13), (0, 4), (6, 4), (5, 4), (16, 4), (10, 4), (2,
4), (12, 4), (22, 4), (22, 10), (13, 10), (4, 10), (10, 10), (1, 10), (0, 10), (0, 6), (3, 6), (8, 6), (7, 6),
(6, 6), (15, 6), (11, 6), (1, 6), (1, 16), (7, 16), (3, 16), (11, 16), (6, 16), (15, 16), (5, 16), (9, 16),
(13, 16), (21, 16), (12, 16), (12, 7), (10, 7), (0, 7), (8, 7), (16, 7), (13, 7), (7, 7), (2, 7), (5, 7), (5,
2), (3, 2), (3, 5), (12, 5), (9, 5), (6, 5), (1, 5), (10, 5), (14, 5), (11, 5), (21, 5), (20, 5), (20, 15), (18,
15), (12, 15), (10, 15), (1, 15), (7, 15), (7, 4), (18, 4), (24, 4), (24, 2), (23, 2), (22, 2), (19, 2), (11,
2), (11, 10), (5, 10), (2, 10), (2, 8), (8, 8), (5, 8), (5, 5), (5, 9), (3, 9), (7, 9), (6, 9), (1, 9), (9, 9), (8,
9), (13, 9), (2, 9), (2, 20), (0, 20), (11, 20), (4, 20), (10, 20), (9, 20), (1, 20), (7, 20), (5, 20), (12,
20), (22, 20), (19, 20), (15, 20), (8, 20), (18, 20), (16, 20), (13, 20), (13, 18), (16, 18), (16, 8),
(10, 8), (1, 8), (3, 8), (3, 3), (1, 3), (1, 1), (11, 1), (21, 1), (10, 1), (15, 1), (19, 1), (19, 8), (11, 8),
(12, 8), (13, 8), (14, 8), (6, 8), (6, 14), (13, 14), (9, 14), (0, 14), (10, 14), (21, 14), (12, 14), (3,
14), (3, 11), (12, 11), (6, 11), (15, 11), (23, 11), (19, 11), (21, 11), (14, 11), (16, 11), (16, 1), (16,
10), (14, 10), (17, 10), (23, 10), (15, 10), (15, 0), (18, 0), (8, 0), (19, 0), (19, 3), (10, 3), (2, 3),
(11, 3), (4, 3), (4, 6), (14, 6), (12, 6), (9, 6), (18, 6), (18, 17), (11, 17), (8, 17), (5, 17), (12, 17),
(16, 17), (16, 6), (24, 6), (23, 6), (20, 6), (17, 6), (17, 3), (14, 3), (24, 3), (18, 3), (23, 3), (23, 8),
(21, 8), (21, 19), (10, 19), (6, 19), (16, 19), (14, 19), (11, 19), (7, 19), (5, 19), (13, 19), (24, 19),
(24, 8), (22, 8), (22, 14), (11, 14), (7, 14), (5, 14), (4, 14), (4, 7), (4, 4), (3, 4), (3, 0), (7, 0), (17,
0), (21, 0), (21, 4), (21, 9), (20, 9), (12, 9), (11, 9), (0, 9), (0, 5), (0, 16), (2, 16), (8, 16), (8, 23),
(5, 23), (16, 23), (11, 23), (20, 23), (10, 23), (14, 23), (12, 23), (15, 23), (7, 23), (17, 23), (17,
15), (17, 5), (8, 5), (8, 3), (8, 12), (3, 12), (12, 12), (21, 12), (17, 12), (7, 12), (6, 12), (5, 12), (13,
12), (23, 12), (14, 12), (16, 12), (9, 12), (20, 12), (20, 18), (14, 18), (15, 18), (15, 13), (20, 13),
(11, 13), (5, 13), (4, 13), (14, 13), (14, 24), (9, 24), (6, 24), (12, 24), (1, 24), (2, 24), (13, 24), (13,
17), (22, 17), (14, 17), (23, 17), (19, 17), (24, 17), (24, 15), (16, 15), (16, 14), (8, 14), (14, 14),
(15, 14), (20, 14), (19, 14), (19, 12), (10, 12), (10, 21), (8, 21), (0, 21), (0, 15), (0, 22), (10, 22),
(9, 22), (2, 22), (12, 22), (14, 22), (19, 22), (19, 16), (19, 6), (10, 6), (21, 6), (21, 15), (21, 20),
(24, 20), (24, 16), (24, 7)]
Number of states :  763
---
```

Astar

```
 ---  Astar  ---
Shortest path -  [(0, 0), (9, 0), (9, 8), (9, 14), (18, 14), (18, 7), (18, 15), (24, 15), (24, 7)]
Number of states :  2346
---
```

3.

8*8 grid

```
 ---  DFS  ---
Number of unique paths :  1110
The number of states expanded: 54257
```

6*6 grid

```
 ---  DFS  ---
Number of unique paths :  79
The number of states expanded: 4202
```