Leminur Çelik – 150190085

BLG 435E ARTIFICIAL INTELLIGENCE HOMEWORK 1

1.

Agent Type	Performance Measure	Environment	Actuators	Sensors
A package sorting system that distributes packages on the belt conveyor depending on their route	Packages sorted correctly	Conveyor belt, packages	Conveyor belt motors, mechanical arms	Camera, package scanners
A tool assessing the similarity of BLG435E assignment submissions both with each other and with web	Plagiarism detection accurately	Documents, web resources	Text comparison algorithms	Assignment submission input, web search api's, database of prior submissions
An autonomous rocket traveling intercontinental	Safety, fast, navigation	Airline, atmosphere, space	Display, signal, rocket engines, guidance system	Cameras, GPS, speedometer, engine sensors
A product recommendation system using user preferences	Appreciation of products, user satisfaction	Users, products, UI,	Product recommendation algorithm, UI controls	Users, product data

Table 1 – PEAS Description of Agents

Agent Type	Observable	Deterministic	Episodic	Static	Discrete	Agents
A package sorting system that distributes packages on the belt conveyor depending on their route	Fully	Deterministic	Episodic	Static	Discrete	Single
A tool assessing the similarity of BLG435E assignment submissions both with each other and with web	Fully	Deterministic	Episodic	Dynamic	Discrete	Single
An autonomous rocket traveling intercontinental	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi

A product recommendation system using user preferences	Fully	Stochastic	Sequential	Dynamic	Discrete	Single

Table 2 – Types of Environments

Type of the Agent Architectures

- A package sorting system that distributes packages on the belt conveyor depending on their route is simple-reflex agent. System's decision is based only on the current location.
- A tool assessing the similarity of BLG435E assignment submissions both with each other and with web is model-based agent. To assist in the similarity assessment process, this tool generates internal models or representations of assignment submissions and web content. The characteristics and features of the assignments as well as the online content are captured by these models.
- An autonomous rocket traveling intercontinental is goal-based agent. The rocket must reach the target according to plan. It will work properly when it reaches the destination.
- A product recommendation system using user preferences is learning agent. A recommendation system enhances recommendation accuracy by analyzing past data and user interactions to comprehend and adjust to evolving preferences. It learns each time based on user satisfaction and offers new suggestions accordingly.

2.

a) Admissible heuristic function's properties are
   a. $h(n) \leq h(n)^*$
   b. $h(n) \geq 0$ and $h(G) = 0$

For this graph to be admissible, the value of h(B) must be less than or equal to 11 and greater than or equal to 0.

$$h(B) \leq 11 \text{ and } h(B) \geq 0, \text{ so } 0 \leq h(B) \leq 11$$

b) Consistent heuristic function's property is
   a. $h(n) \leq c(n, a, n') + h(n')$

For this graph to be consistent, the value of h(B) must be less than or equal to 10 and greater than or equal to 7.

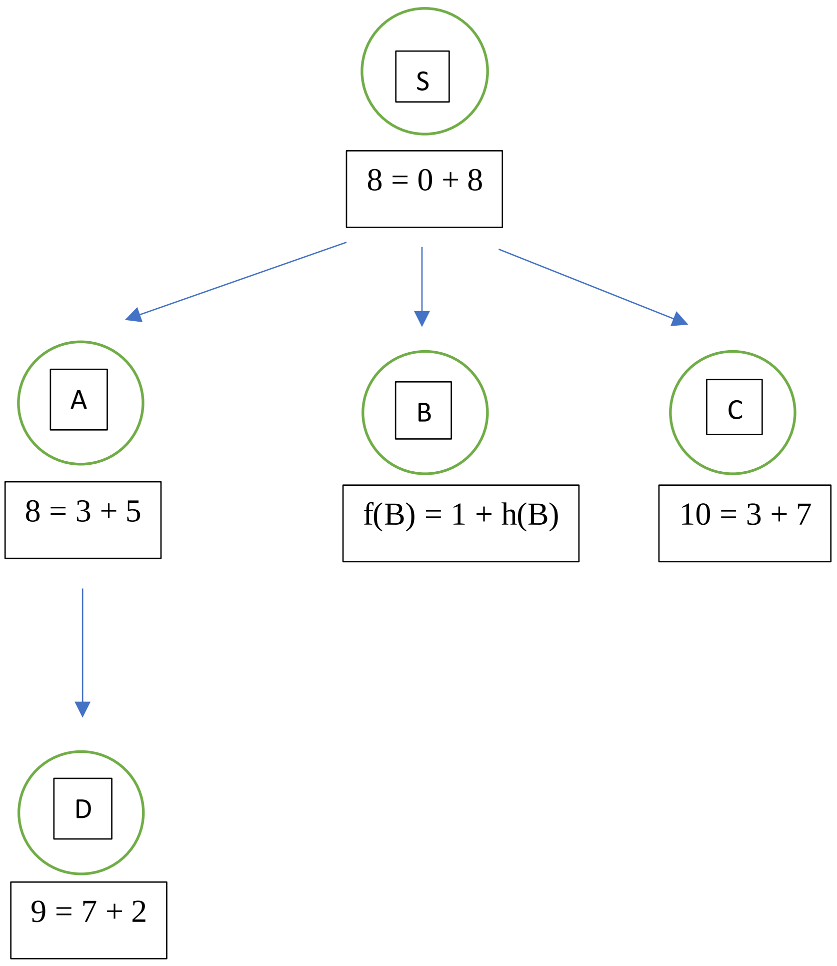
$h(S) \leq c(S, a, B) + h(B)$ $\quad$ $h(B) \leq c(B, a, C) + h(C)$

$8 \leq 1 + h(B)$ $\quad$ $h(B) \leq 3 + 7$

$7 \leq h(B)$ $\quad$ $h(B) \leq 10$

$$7 \leq h(B) \leq 10$$

c) For this graph to expand first node S, then node A, then node B in order, f(B) must be greater than or equal to 8 and less than or equal to 9. If f(B) is equal to 8 that means h(B) is equal to 7, then we expand first node A if we use FIFO rule because A is generated earlier. After that, f(D) is bigger than f(B). Therefore, we choose node B to expand. If f(B) is equal to 9 that means h(B) is equal to 8, then we expand first A because f(A) is less than f(B) and f(C). After that, if we use FIFO rule, we expand first node B because f(B) is equal to f(D) and node B is generated earlier. Consequently, h(B) may be greater than or equal to 7, or less than or equal to 8, depending on the specific application, such that equivalent $7 \leq h(B) \leq 8$.



3. Problem Solving with Search Algorithms

3.1 Comparing Search Algorithms

	Cost				Number of Expanded Nodes				Execution Time (seconds)				Memory Usage (kilobytes)			
Algorithms	BFS	DFS	IDS	A*	BFS	DFS	IDS	A*	BFS	DFS	IDS	A*	BFS	DFS	IDS	A*
Maze 1	63	135	63	63	347	280	149	181	7.7	7	140	4	108	96	32	64
Maze 2	105	179	115	105	374	490	239	279	8.3	11	326	6.5	110	104	112	92
Maze 3	37	49	37	37	142	844	100	86	2.9	15	36	2	77	108	88	84
Maze 4	59	417	59	59	697	752	199	524	13.5	21	424	10	116	140	212	128

Table 3 – Comparing Search Algorithms

Algorithms are compared according to cost, number of expanded nodes, execution time, memory usage and whether they are optimal or not and whether they are complete or not. Complete search algorithm is guaranteed to find a solution if one exists within the search space. All algorithms can reach the destination in mazes. Therefore, they can be described as complete.

Breadth-First Search (BFS) and A* are optimal search algorithms when it comes to finding the shortest path in a graph or a tree, assuming that each edge has a uniform cost. It can be seen from the cost in the Table 3 that BFS and A* are optimal. Since BFS looks nodes at level by level, the number of expanded nodes is high. Depth-First-Search (DFS) has not found the optimal result as it progresses depth by depth.

Depending on the type of mazes, sometimes BFS and sometimes DFS can reach the final point faster. In single-row mazes, when the destination is deep, DFS can reach faster than it goes by looking deeper. If there are not many obstacles around a node, it will take more time to reach the destination if the destination is far away, since BFS will travel to all the nodes around it.

Since the IDS algorithm increases the limit one by one, it takes a long time to reach the destination if it is deep. Therefore, its execution time is higher than other algorithms. The optimality of IDS depends on the specific problem. It achieved optimal results in mazes with not many expanded nodes.

Execution time is calculated by time module in Python. Current time is recorded before and after running the algorithms and the differences are calculated. Memory usage is calculated by psutil library in Python. Memory usage is measured before and after running the algorithms and the differences are calculated.

3.2 Robot Path Planning

When the 5th maze is run with the A* algorithm, a total of 7 turns are seen in Figure 1, while in the specialized version of the A* algorithm, when the turn movements are counted from the cost, 5 turns are seen in Figure 2. In the new heuristic function, it is checked whether the direction has changed or not. I used the Manhattan distance formula in heuristic function. The total of the horizontal and vertical distances between the current state and the desired state is known as the Manhattan distance. It is typically more expensive to travel diagonal paths in a maze than to move only in the horizontal and vertical directions. As a result, the Manhattan distance is an admissible heuristic since it will always be less than or equal to the actual cost of achieving the goal. Since the Manhattan distance formula is admissible, the A* algorithm is optimal.

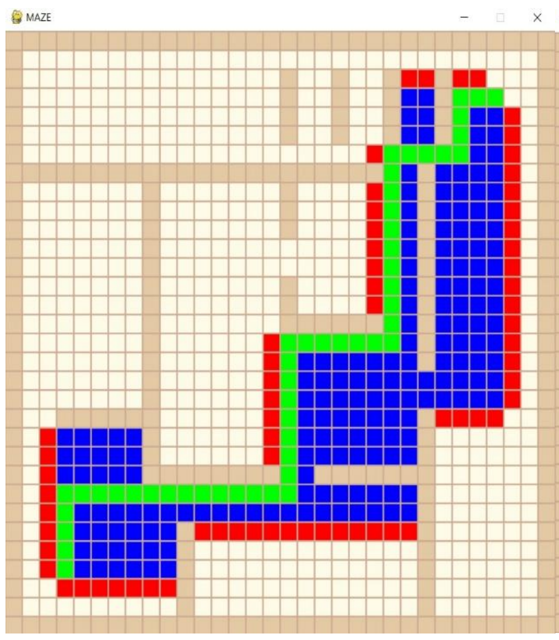


Figure 1 – A* Algorithm

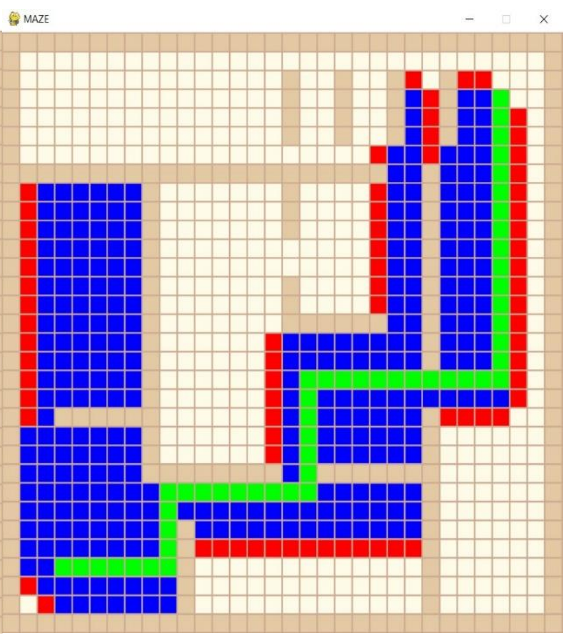


Figure 2 – A* Algorithm with turn cost