Krishnaraj P T
PA 10.
24-9-23

classmate
Date
Page

AI Theory assignment 1.

Q:1. Explain the classification of environments in detail and explain PEAS analysis for automated taxi agent.

→ Classification of Environment :

They can be classified based on several criteria including observability, determinism, episodic vs sequential, static vs dynamic, discreet vs. continuous and single agent vs multi agent.

A. Observability:

(i) Fully observable : An agent has access to the complete state of the environment at all times.

(ii) Partially observable : The agent has limited or partial access to the state of the environment.

B. Determinism :

1. Deterministic : The next state of the environment is completely determined by the current state; and the actions taken by the agent.

2. Stochastic : The next state is partially random or uncertain even with the same actions and current state.

C. Episodic vs. sequential :

→ Episodic : The agent experience is divided into independent episodes and each episode does not depend on previous ones.

→ Sequential : The current decision and action may affect all future decisions and outcomes.

D. Static vs. Dynamic :

→ Static : The env. does not change while an agent is deliberating.

→ Dynamic : The environment may change while the agent is deciding on its actions.

E. Discreet vs. Continuous

→ Discreet : The state, actions or time are divided into distinct separate units.

→ Continuous : The state actions or time are also continuous and not divided into discreet units.

F. Single agent vs. Multi agent :

→ Single : There is only one agent interacting with the environment.

→ Multi-agent : Multiple agents interact with each other and the environment.

(A) PEAS analysis for automated Taxi Agent:

1. Performance Measure: Can be measured by metrics such as reaching the destination is the shortest time, providing a comfortable ride, adhering to traffic rules, minimizing fuel consumption, etc.

2. Environment: It includes the road network, traffic conditions, weather, other vehicles, pedestrians, traffic signals, etc.

3. Actuators: They include the vehicles steering, accelerator, brake, turn signals, GPS navigation system, etc.

4. Sensors: They include cameras, LIDAR, ultrasonic sensors, GPS, speedometers, traffic ~~as~~ sensors, temperature sensors and ~~mic~~ microphones for understanding environment.

Q. 2. Drive 3 heuristic functions to solve 8 puzzle problem and explain admissibility property of a A* alg in ~~detail~~ detail.

→ Heuristic function estimate the cost from a given state to the global state. These commonly ~~used~~ heuristics are:

1. Misplaced tiles: This heuristic calculates the number of tiles that are not in their

their goal position in the current state. It underestimates the true cost as it assumes that moving a tile to its goal position will not disturb other tiles.

2. Manhattan distance Heuristic : This calculates the sum of the manhattan distances of each tile from its current position to its goal position. The manhattan distance for a tile is the sum of the horizontal and vertical distances between the current position and the goal position

3. Linear Conflict Heuristic

This heuristic extends the Manhattan distance heuristic by considering conflict between tiles in the same row or column. It admissible since it considers a lower bound on the actual cost by accounting for the conflicts.

Admissibility property of A* algorithm :-

→ Admissibility is crucial because it guarantees that A* will find an the optimal solution.

→ A* uses the heuristic guide its search, if the heuristic is admissible, A* is guaranteed to find the shortest path.

→ When the heuristic is admissible, A* explores the search space efficiently, prioritizing nodes that are likely to lead to optimal solutions, improving performance.

Admissibility Definition:

In an A* search, a heuristic $h(n)$ is admissible if and only if it never overestimates the true cost to reach the goal from any given state.

$$h(n) \leq h^*(n)$$

where $h^*(n)$ is the true (optimal) cost from state n to the goal.

(3) Explain the minimax alg with α-β pruning.

→ Minimization and Maximization:

Maximizes aims to maximize the score.

→ Recursive Tree Exploration (Game Tree):

The algorithm explores a game tree where each level alternates between max and min nodes.

Terminal node evaluation:

At the leaf nodes (terminal states) the utility function evaluates the state of the game and provides a numerical value representing the outcome of the max player.

Backward propagation:

The algorithm propagates the values back up the tree, updating the values for each of its children.

Decision:

Finally, Max selects the move that leads to the highest value among its child nodes.

Alpha Beta Pruning:

→ Alpha and Beta are the best value that the maximizing & minimizing player respectively can guarantee at or above the current level.

→ Pruning Condition: During the tree traversal if the algorithm finds a node's value that makes it irrelevant for further exploration, it prunes the subtree rooted at that node.

(*) Pruning Operation:

If the current node is a minimum node and its value is less than or equal to alpha, prune the subtree and return the value.

(*) Updating Alpha and Beta:

When traversing the tree, update alpha for min nodes as the maximum of alpha and the current node's value.

Q.4. Explain local search algorithm. Solve travelling salesman problem using local search algorithm.

→ Initialization:
Start with an initial solution. Set the current solution as the initial solution.

→ Evaluation:
Evaluate the current solution to determine its quality based on the objective function or evaluation criteria.

→ Neighbours Generation:

generate neighbouring solutions by making small changes to the current solution.

④ **Selection:**

→ Select the best neighbours based on evaluation criteria.

⑤ **Termination condition:**

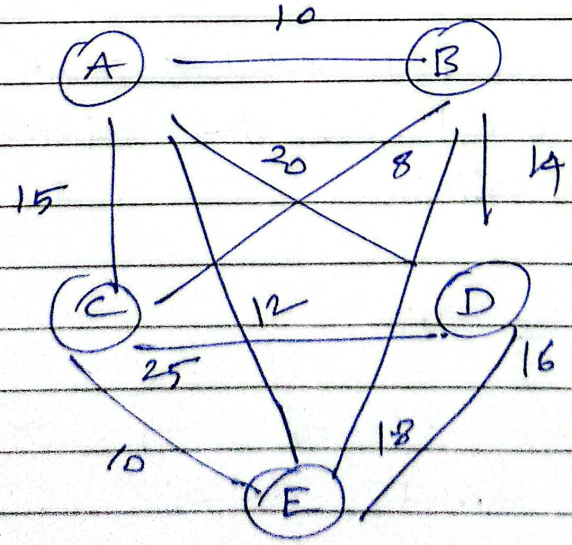
→ Repeat steps 3-4 until a stopping criteria is met.

⑥ **Output**

Return the best solution found during search.

⑦ **TSP**

Let's consider (A, B, C, D, E.) as 5 cities and their pairwise distances.

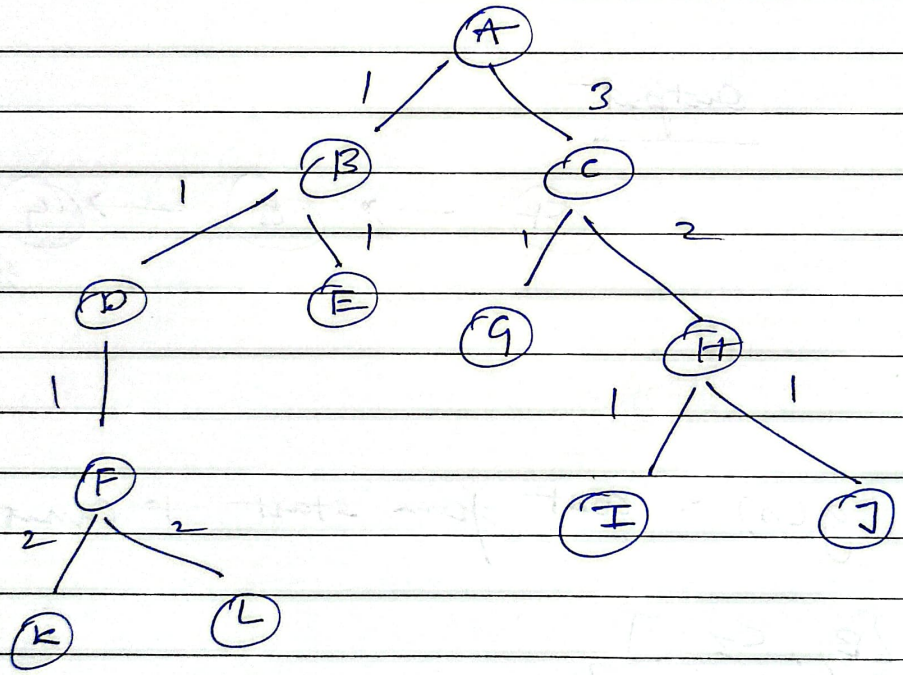


- let us begin with an initial random tour

A → B → C → D → E → A.

→ we apply the local search algorithm iteratively to improve the tour and minimize the total distance. After several iterations we'll reach a locally optimal tours with the shortest total distance

9.5.



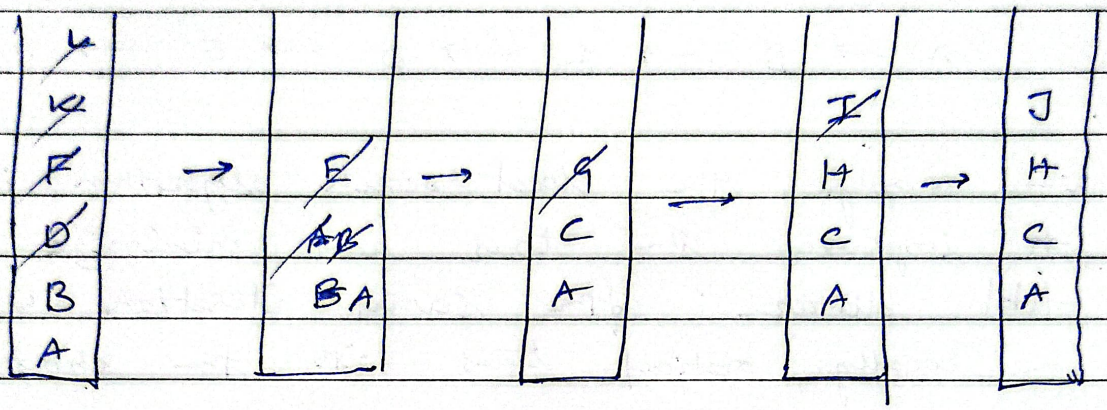
BFS :

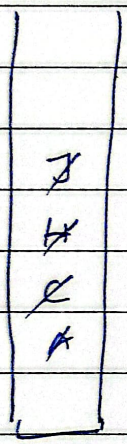
queue : A, B, C, D, E, G, H, F, I, J, K, L

So output when we have to stop at G

A → B → C → D → E → G

DFS





Output

A → C → G

UCS

g(n) = cost from start to current node

A	$[B_1, C_1]$
B	$[D_2, E_2, C_3]$
D	$[E_2, F_3, C_1]$
E	$[F_3, C_3]$
C	$[F_3, G_4, H_5]$
F	$[G_4, H_5, K_5, L_5]$
G	

Total to stop at G

output : A, B, D, E, C, F, G