Name - Vasu Kalariya
Roll - PE29
Sub - AI

Theory Assignment - 1

Q 1 Explain the classification of environment in details and also explain PEAS analysis for automated taxi adgent.

We know that there are different types of adgents in AI. PEAS system is used to categorize similar adgents together. The PEAS system delivers the performance measures with respect to environment, actuators and sensors.

1) Performance Measure:
It is a unit used to define how much successful an adgent is.

2) Environment:
~~This~~ It is surrounding of an adgent at every instant. It keeps changing with time if adgent is ~~seat~~ in motion, There are five major types of environments.
(i) fully observable & partially observable
(ii) Sequential
(iii) Static & Dynamic
(iv) Discrete & Contineous
(v) Deterministic & Stochastic.

3) Actuators:
It is part of the adgent that delivers the output of an actuators to environments.

4) Sensors:
They are receptive parts of an adgent which are taken in the impact of adgent

Q 2 Define heuristic method and Explain admissible property of A^* algorithm in details.

A heuristic is a technique to solve a problem faster than classic methods or to find an approximate solution when classic methods cannot. This is a kind of a shortcuts as we often made one of optimality, completeness, accuracy or precision for speed. A heuristic takes a look a search algorithm At each branching step, it evaluates the available information and makes a decision on which branch to follow.
It does so by ranking alternatives. The Heurstic is any device that is often effective will not guarantee work in every case.

Admissibility of A^*
The heuristic function $\in h(n)$ is called admissible if $h(n)$ is never larger than $h^*(n)$, namely $h(n)$ is always less or equal to true cheapest cost from n to the goal.
A^* is admissible if it uses an admissible heuristic and $h(goad) = 0$.
If the heuristic function, h always understimates the true cost ($h(n)$) is smaller than $h^*(n)$ then A^* is guaranteed to find an optimal solution.
A^* : $f(n) = g(n) + h(n)$
Admissible: $h(n) <= h^*(n)$

3 Explain the MinMax algorithm with example

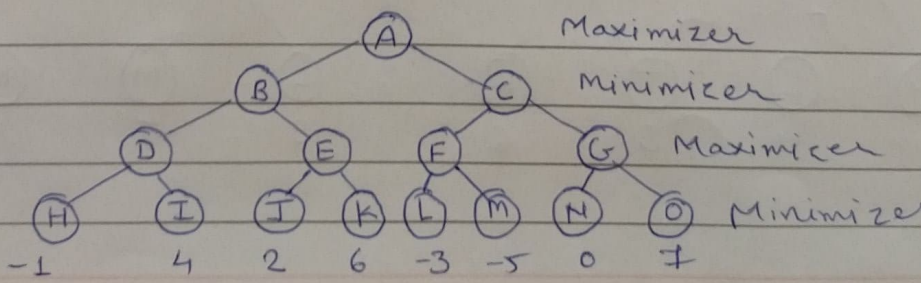
→ Minimax Algorithm

Minimax Algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory.

It is used to find the optimal move for a player assuming that your opponent also plays optimally. It is widely used in two player turn-based games such as Tic Toc Toe, Backgammon, Mancala, Chess etc. In MinMax the two players one called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible.. Every board state has a value associated with it. In a given state if the maximizer has upper hand then, the score of the board will tend to be some positive value. If the minimizer has the upper ~~board~~ hand is that board state then it will tend to be some negative value. The values of the board are calculated by some heuristics which are unique for every type of game.

Eg:

Step 1: Algorithm is solving two player game tree. for terminal states ~~utility function to get utility values for from~~ If maximizer takes first turn which has worst case initial value = $-\infty$ and ~~maxi~~minimizer will takes next turn which has worst case initial value = $+\infty$



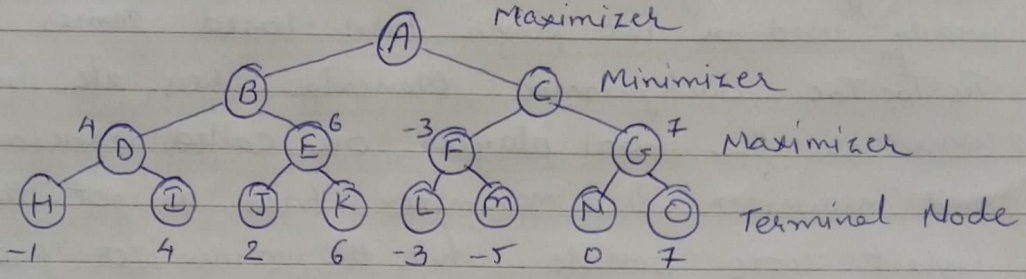
Step 2: First we find utilities values for maximizer, its initial value is $-\infty$, so we will compare each value in terminal state with initial value of Maximizer and determines the higher node values. It will find maximum among all

For Node D $\max(-1, -\infty) \Rightarrow \max(-1, 4) = 4$

Node E $\max(2, \infty) \Rightarrow \max(2, 6) = 5$

Node F $\max(-3, -\infty) \Rightarrow \max(-3, -5) = -3$

Node G $\max(0, -\infty) \Rightarrow \max(0, 7) = 7$



Step 3: Its minimizer turn, so it will compare with $+\infty$ and will find 3rd layer nodes.

for node B = min (4, 6) = 4

node C = min (-3, 7) = -3

Similarly,

Step 4: Its Maximizer turn it will again choose the maximum of all nodes values and find the maximum value for the root node.

for node A = max (4, -3) = 4