ARTIFICIAL INTELLIGENCE

ASSIGNMENT-1

1.1) Branching factor (BF) is the number of children at each node or the out degree. As we can see from the following figure, each parent node has nine (9) child nodes. So, its branching factor is Nine (9), and it is uniform.

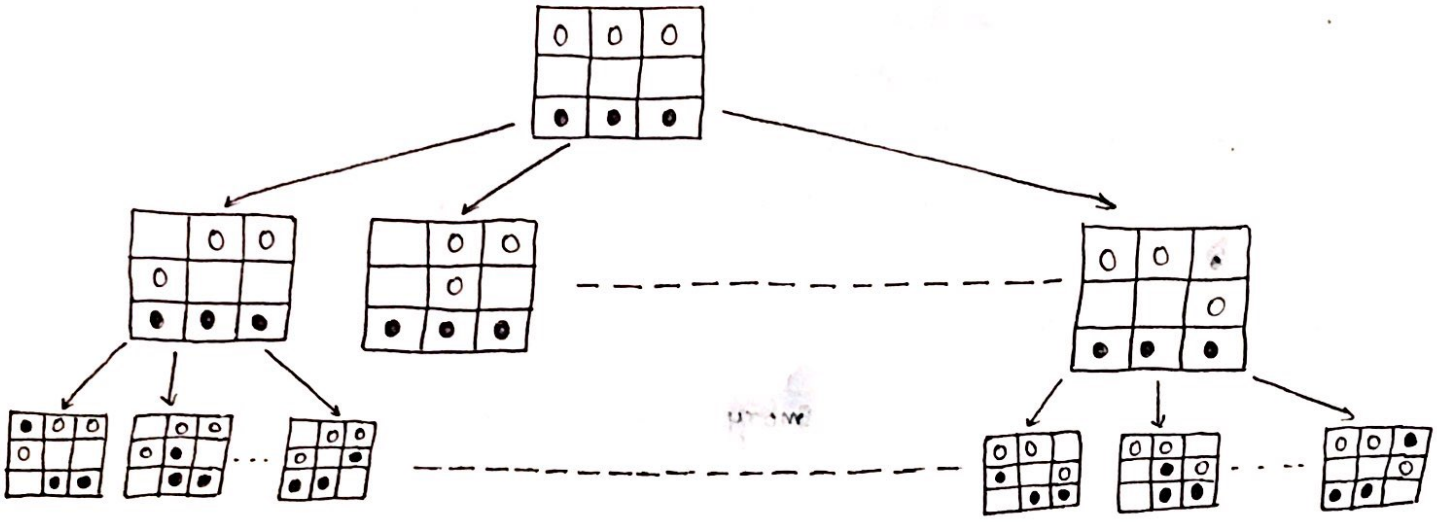


FIGURE.1 – The Search tree for a game 'SEGA'.

1.2) Search tree for the given 'SEGA' game is shown in FIGURE.2 below:

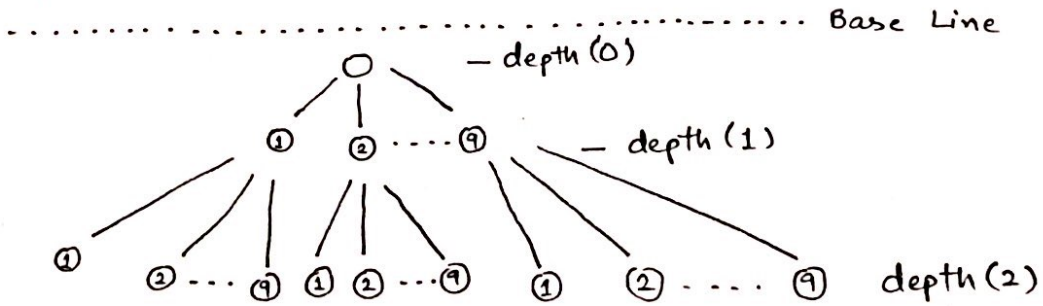


FIGURE.2 – Depth-2 Search tree for 'SEGA' game.

In this search tree, at depth (0) we have one root node, which has '9' child nodes at depth (1) and further each node at depth (1) also have again (9) child nodes proceed to depth (2). Hence (81) total nodes at depth (2).

Scanned by CamScanner

1.3) Based on the following criterions, we have evaluated our search technique.

- Optimality
- Time Complexity
- Space Complexity
- Completeness

Among Uninformed search techniques, the technique which fits best to implement this problem would be Breadth first search (BFS) technique. As it will always give optimal solution to the problem by finding shortest path. Thus we can always guarantee to find the solution. The space and time Complexity for BFS will be low even for worst cases.

But BFS consumes more memory. Thus to reduce the memory use, we can implement DEpth first Search (DFS) technique with following conditions.

- We should always be able to find the solution.
- It should have finite depth and loop.

Thus we can use BFS with DFS to reduce space and time Complexity.

1.4) The following is the attached code for the game :

Scanned by CamScanner

1.5) Actually, we went through this game and played many times. we noticed that 'SEGA' could not stuck in a loop if both the players are playing with winning intentions.

But for the case, Let us consider both the players are hungery and playing 'SEGA' just for time pass and have no intention to win a game, then this game have many possibilities to get stuck in a loop.

Below figure.3 shows one of the loop case for 'SEGA'.

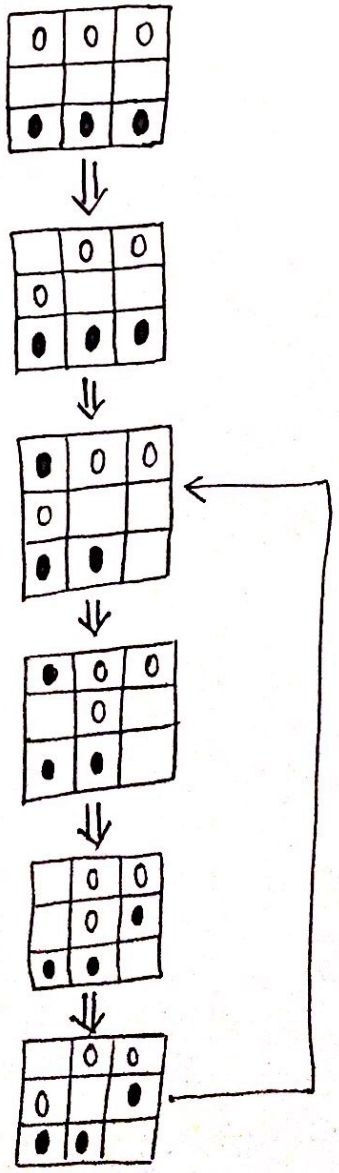


FIGURE.3
Shows game got stuck in a loop.

Scanned by CamScanner

2.1 We can use Dijikstra algorithm to extract all possible attack scenerios route and to find shortest path

Accounting →

I $\xrightarrow{90}$ M.A $\xrightarrow{120}$ A.S $\xrightarrow{330}$ DBS $\xrightarrow{610}$ Accounting → 1150

I $\xrightarrow{90}$ M.A $\xrightarrow{120}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{30}$ DBS $\xrightarrow{610}$ A 1150

I $\xrightarrow{50}$ W.A $\xrightarrow{120}$ W.S $\xrightarrow{90}$ A.S $\xrightarrow{330}$ DBS $\xrightarrow{610}$ A → 1200

I $\xrightarrow{70}$ RAPI $\xrightarrow{80}$ W.S $\xrightarrow{100}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{30}$ DBS $\xrightarrow{610}$ A → 1190

Work Station → 1 more case I → RAPI → 1910

I $\xrightarrow{90}$ M.A $\xrightarrow{120}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ P.N $\xrightarrow{710}$ LDAP $\xrightarrow{240}$ W.S → 1870

I $\xrightarrow{90}$ M.A $\xrightarrow{20}$ W.S $\xrightarrow{100}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ PN $\xrightarrow{710}$ LDAP $\xrightarrow{240}$ W.S

I $\xrightarrow{50}$ W.A $\xrightarrow{120}$ W.S $\xrightarrow{90}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ PN $\xrightarrow{710}$ LDAP $\xrightarrow{240}$ W.S

I $\xrightarrow{40}$ E.P $\xrightarrow{170}$ VPA $\xrightarrow{120}$ PN $\xrightarrow{710}$ LDAP $\xrightarrow{240}$ W.S → 1280

1920

M.S →

I $\xrightarrow{40}$ E.P $\xrightarrow{200}$ E.A $\xrightarrow{356}$ M.S → 596

2460

R&D Server

I $\xrightarrow{90}$ M.A $\xrightarrow{120}$ A.S $\xrightarrow{300}$ DMZN $\xrightarrow{410}$ PN $\xrightarrow{710}$ LDAP $\xrightarrow{830}$ R&D

I $\xrightarrow{50}$ W.A $\xrightarrow{120}$ W.S $\xrightarrow{90}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ PN $\xrightarrow{710}$ LDAP $\xrightarrow{830}$ R&D

I $\xrightarrow{70}$ RAPI $\xrightarrow{80}$ W.S $\xrightarrow{100}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ PN $\xrightarrow{710}$ LDAP $\xrightarrow{830}$ R&D

2500

I $\xrightarrow{40}$ E.P $\xrightarrow{170}$ VPN $\xrightarrow{120}$ PN $\xrightarrow{710}$ LDAP $\xrightarrow{836}$ R&D 2510

1870

Scanned by CamScanner

H.R Server

I $\xrightarrow{40}$ E.P $\xrightarrow{170}$ VPN $\xrightarrow{120}$ PN $\xrightarrow{530}$ HR → 860

I $\xrightarrow{90}$ M.A $\xrightarrow{120}$ A..S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ PN $\xrightarrow{530}$ HR → 1450

I $\xrightarrow{50}$ W.A $\xrightarrow{120}$ W.S $\xrightarrow{90}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ PN $\xrightarrow{530}$ HR → 1500

I $\xrightarrow{70}$ RAPI $\xrightarrow{80}$ WS $\xrightarrow{100}$ A.S $\xrightarrow{300}$ DMZ $\xrightarrow{410}$ PN $\xrightarrow{530}$ HR →

1490

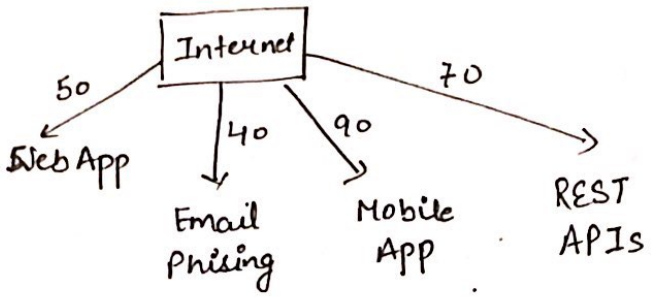
Scanned by CamScanner

2.2

From the given diagram, total cost to reach goal node = 5160

By depth 1 ~~stretch~~ search,

$$\text{Max. depth} = \frac{5160}{250} = 20.64 \approx 20$$



Multiplying 20 by each node, we get

Web App → 50 × 20 = 1000

Email Phishing → 40 × 20 = 800

Mobile App → 90 × 20 = 1800

REST API's → 70 × 20 = 1400

After adding this cost to each, we get

Web App → 50 + 1000 = ~~1500~~ 1050

Email Phishing → 40 + 800 = 840

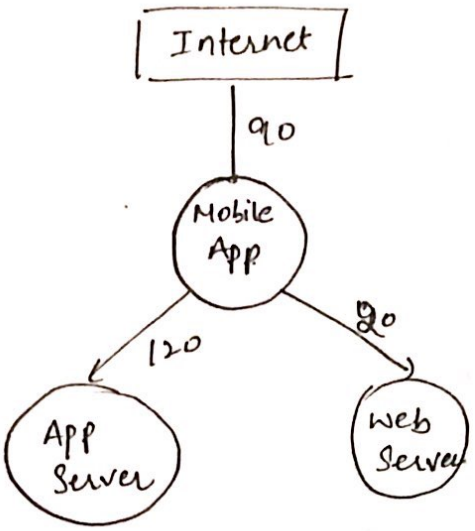
Mobile App → 90 + 1800 = 1890

REST API's → 70 + 1400 = 1470

As we can see, Attacker can get maximum gain from Mobile App.

Scanned by CamScanner

Expanding nodes of Mobile App



Calculating depth for second node $= \frac{4910}{250+140} = \frac{4910}{390} = 12.589 \approx 13$

App Server → 13 x 120 = 1560

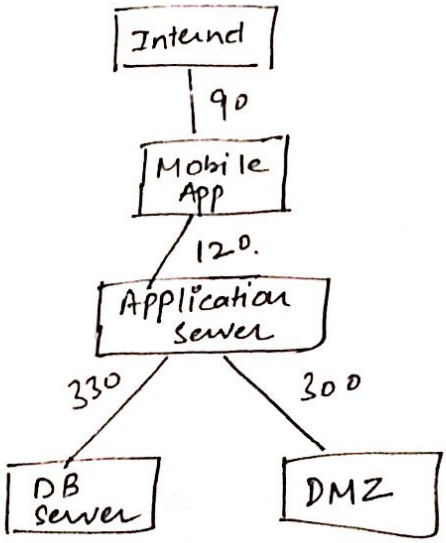
Web Server → 13 x 20 = 260

Cost : = App Server → 120 + 1560 = 1680

Web Server → 20 + 260 = 280

So we can say that Attacker will get maximum gain from App Server

Expanding nodes of App Server.



Scanned by CamScanner

DB Server → 330 × 4 (depth) = 1320

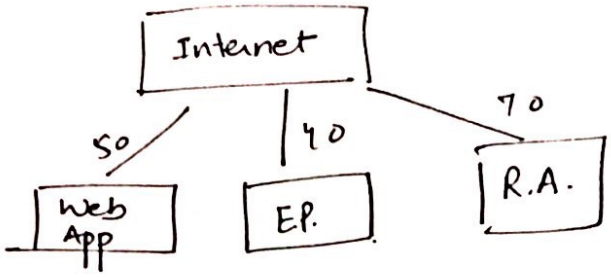
DMZ → 300 × 4 = 1200

Cost : DB Server → 330 + 1320 = 1650

DMZ → 300 + 1200 = 1500

Here, DB Server is max. gain for attacker

So we are left with 850 as remaining budget



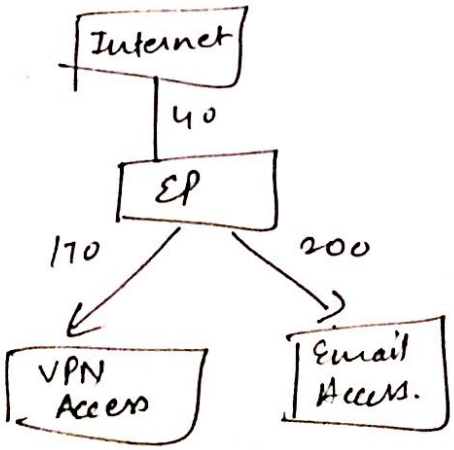
Maximum depth = $\frac{401\phi}{16\phi} = 25.\alpha \approx 25$

Web App → 50 × 25 = 1250

Email Phishing → 40 × 25 = 1000

REST API's → 70 × 25 = 1750

Picking the least value & Expanding



Scanned by CamScanner

VPN Access → $170 \times \left(\frac{3850}{530}\right) = 1190$

Email Access → $200 \times 7 = 1400$

Cost : VPN Access → $170 + 1190 = 1360$

Email Access → $200 + 1400 = 1600$

Email has maximum gain and we are left with ~~280~~ 254 as remaining budget which is too low to reach another goal node.

Scanned by CamScanner

Solution 1.4 –

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] arr1 = { 1, 1, 1 };
            int[] arr2 = { 0, 0, 0 };
            int[] arr3 = { 2, 2, 2 };

            Console.WriteLine("Welcome to the Game.Lets start the game.\n make sure that
the sending and receing position should be between 0 and 2");


            for (var i = 0; i < 9; i++)
            {
                Console.WriteLine("For first person From which array");
                int sendingarr1 = Convert.ToInt32(Console.ReadLine());

                Console.WriteLine("For first person Enter which array you want to
place");
                int receivedarr1 = Convert.ToInt32(Console.ReadLine());

                Console.WriteLine("For first person Enter which position in the array you
want to place");
                int receivingposition1 = Convert.ToInt32(Console.ReadLine());

                Console.WriteLine("For first person Enter from which position u r
moving");
                int sendingpostion1 = Convert.ToInt32(Console.ReadLine());

                if (i == 0)
                {
                    //For First person
                    if (sendingarr1 == 1)
                    {
                        if (receivedarr1 == 2)
                        {
                            arr2[receivingposition1] = 1;
                            arr1[sendingpostion1] = 0;
                        }

                        if (receivedarr1 == 1 || receivedarr1 == 3)
                        {
                            Console.WriteLine("You cannot move in here.");
                            break;
                        }
                    }
```

```
                    if (sendingarr1 == 3)
                    {
                        if (receivedarr1 == 2)
                        {
                            arr2[receivingposition1] = 2;
                            arr3[sendingpostion1] = 0;
                        }

                        if (receivedarr1 == 1 || receivedarr1 == 3)
                        {
                            Console.WriteLine("You cannot move in here.");
                            break;
                        }
                    }

                    if (sendingarr1 == 2)
                    {
                        Console.WriteLine("Sorry there are no elements in the 2nd
array.");
                        break;
                    }
                }

                else
                {
                    //First Person
                    if (sendingarr1 == 1)
                    {
                        //if (sendingpostion1 == 1)
                        //{
                        if (receivedarr1 == 2)
                        {
                            if (arr2[receivingposition1] == 0)
                            {
                                arr2[receivingposition1] = 1;
                                arr1[sendingpostion1] = 0;
                            }

                        }
                        if (receivedarr1 == 3)
                        {
                            if (arr3[receivingposition1] == 0)
                            {
                                arr3[receivingposition1] = 1;
                                arr1[sendingpostion1] = 0;
                            }
                        }

                        if (receivedarr1 == 1)
                        {
                            if (arr1[receivingposition1] == 0)
                            {
                                arr1[receivingposition1] = 1;
                                arr1[sendingpostion1] = 0;
                            }
                        }
                        //}
```

```
    //else
    //{
    //    Console.WriteLine("Sorry you cannot move this one");
    //    break;
    //}
}



if (sendingarr1 == 2)
{
    //if (sendingpostion1 == 1)
    //{
    if (receivedarr1 == 2)
    {
        if (arr2[receivingposition1] == 0)
        {
            arr2[receivingposition1] = 1;
            arr2[sendingpostion1] = 0;
        }

    }
    if (receivedarr1 == 3)
    {
        if (arr3[receivingposition1] == 0)
        {
            arr3[receivingposition1] = 1;
            arr2[sendingpostion1] = 0;
        }
    }

    if (receivedarr1 == 1)
    {
        if (arr1[receivingposition1] == 0)
        {
            arr1[receivingposition1] = 1;
            arr2[sendingpostion1] = 0;
        }
    }
    //}
    //else
    //{
    //    Console.WriteLine("Sorry you cannot move this one");
    //    break;
    //}
}

if (sendingarr1 == 3)
{
    //if (sendingpostion1 == 1)
    //{
    if (receivedarr1 == 2)
    {
        if (arr2[receivingposition1] == 0)
        {
            arr2[receivingposition1] = 1;
            arr3[sendingpostion1] = 0;
        }
```

```
}
if (receivedarr1 == 3)
{
    if (arr3[receivingposition1] == 0)
    {
        arr3[receivingposition1] = 1;
        arr3[sendingpostion1] = 0;
    }
}
```

Solution-3.1

Turing test, is a test of ability of a machine to exhibit intelligent behavior which is equivalent to or indistinguishable from, that of a human. It is debatable so as to the fact that is turning test is a valid and fair approach to measuring the intelligence of search techniques Artificial Intelligence or not. Turing's proposed test to determine whether a computer thinks he then called as the Imitation Game which is now as the Turing Test.

Turing's paper claimed that suitably programmed digital computers would be generally accepted as thinking by around the year 2000, achieving that status by successfully responding to human questions in a human-like way. To accept this idea, he then gave explanation on what a digital computer is, presenting it as a special case of the "discrete state machine"; he explained a capsule explanation of programming such a machine means; and gave nine arguments against his thesis that such a machine could be said to think.

However, conceive of simple algorithms that would pass a Turing Test but may or may not definitely not be intelligent.

For example, consider an algorithm that traverses a conversation tree, like a state machine, in which nodes are conversational states and edges alternate between being what the machine just received as input (type A edges) and what the machine produces as output (type B edges). Now say we put a constraint on the human user such that the user can type in whatever he or she wants but his or her input must be grammatical and must be less than, say, 200 characters long. Then for each node with type A edges going out of it we provide links for all possible grammatical at most 200 character strings and for each node with type B edges coming out we provide, say, a million canned responses given the conversational state represented by the node.

Now, such a tree couldn't be constructed in the real world as it would be enormous but if it could interactively traversing this tree in the obvious manner which means following the appropriate type A edges and randomly selecting type B edges then would clearly pass the Turing Test but the user clearly wouldn't be interacting with an intelligent machine: the user would be interacting with a random number generator.

So passing a Turing Test can't be viewed as some kind of absolute criteria for exhibiting intelligence but in practice such systems could not be easily constructed, and Turing Tests are therefore valuable pragmatically.

Solution-3.2

In the AI literature, it is very common to describe search technique (algorithms) as search agent. In our opinion the term 'agent' is appropriate to describe search techniques in AI.

Let us discuss few definitions of an agent:

According to MuBot Agent: The term agent is used to represent two concept. The first is the agent's ability for autonomous execution. The second is the agent's ability to perform reasoning.

***According to AIMA agent*:** An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

If we talk about programs, then a payroll program in a real world environment could be said to sense the world via it's input and act on it via its output, but is cannot be considered as an agent because its output would not normally effect what it senses later.

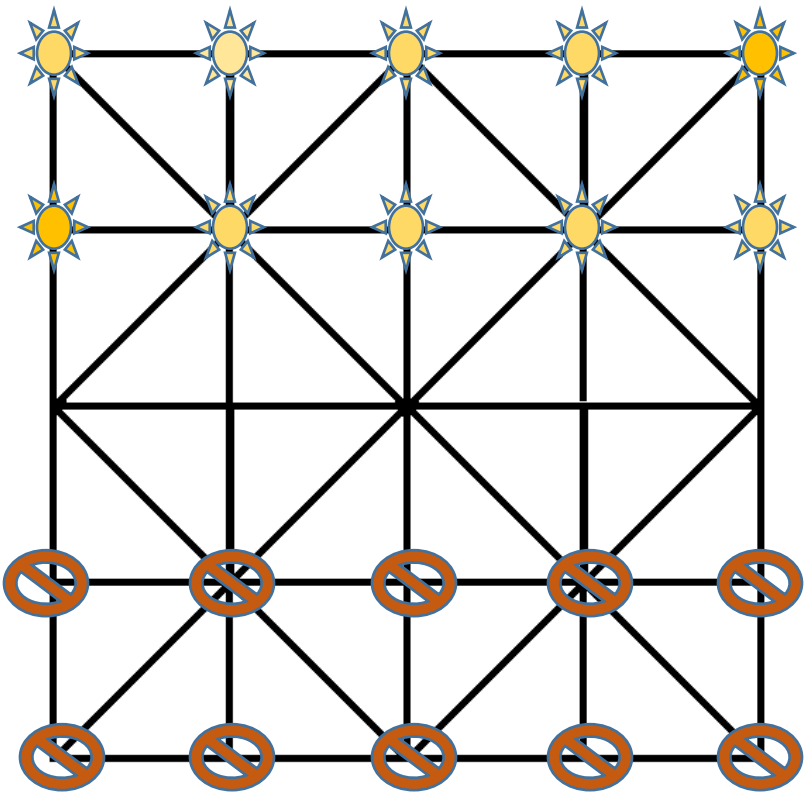
Agents can be classified with the subset of properties that they enjoy. Every agent, by the definition, satisfies the properties like goal-oriented purposeful similar to properties of any Artificial Intelligence search technique.

Each and every agent is situated in, and is a part on some environment. It senses its environment and act autonomously upon it. No other entity is required to feed it input, or to interpret and use its output. Each acts in pursuit of its own agenda, whether satisfying evolved drives as in pursuing goals designed in by some other agent. Each acts so that its current actions may effect its later sensing, that is its actions effect its environment. Same as an Artificial intelligence agent which directs its activity ot achieve goals and acts upon with correspondence to the environment.

When we talk about a software agent then if the software agent, once invoked, it typically runs until it decides not to. An artificial life agent often runs until its eaten or otherwise dies.

Thus, we can say that the term 'agent' is appropriate to describe search techniques in AI.

Bonus Question



12 – Bit Strategic Board Game

12-Bit is a board game, involves thinking and planning. It is a two player game, who takes turn moving by their tokens. Players begin game with 12 tokens each, placed on upper and bottom side nodes of the board as shown in figure and can be moved from node to node along a line.

How to make movements:

Each player can move token

- only to an empty node.
- to an adjacent node or next to adjacent node if the respective adjacent node is booked with opponent's token.

Who is winner?

- If a player's token lands on a node next to an adjacent node (booked with opponent's token), then the opponent's token is killed and removed from the board.
- A player who first kills all opponent's tokens is declared winner.