# ARTIFICIAL INTELLIGENCE

## LAB 1

### Problem Definition:

Construct a program called ‘Animal name guessing game’ using the data structure trees where starting from the root node, the program provides options and moves towards the leaf nodes, according to the player’s response in form of yes/no and provides the answer i.e. the animal in the user’s mind.

### Methodology:

I implemented this game using the programming language python. In my implementation, there are 2 parts, dynamic game generation using binary tree and the guessing game. In my program a user can input the options himself and construct his own guessing game. And that game can again be played by user.

Data structure:

I used the data structure binary tree and queues for this program. I implemented the binary tree using linked lists. I used the level order transversal method to access the nodes of the tree both on the game generation function and the game play function.

#### class node():

#### def \_\_init\_\_(self):

#### self.value = None

#### self.left = None

#### self.right = None

#### self.parent = None

#### self.pos = None

Algorithm:

I used a non-recursive algorithm for implementing the binary tree with the help of queues for both tree generation and tree transversal. It has time complexity O(n) and Space complexity O(n).

Functions:

I constructed the class node() for implementing a single node with a value and 2 children.

I constructed the function level\_order(root) for the user to dynamically create his own tree to make his own guessing game. This function takes a root node as an argument and creates a binary tree.

I constructed the function game(tree) for the user to play the guessing game. This function takes the root node of a tree as an argument and implements the guessing game based on the input tree.

Program flow:

To make the game function level\_order() is called and a root node is passed to it. The nodes are filled one by one in a level order transveral.

To play a game, function game() is called and tree is passed. The program first reads the left node and asks the user if it’s the correct node , if yes, the program transverses the left node, if no, it transverses the right node. This process is repeated in subsequent nodes till the correct leaf node is reached. The program returns the leaf node as the guessed element.

### Output:

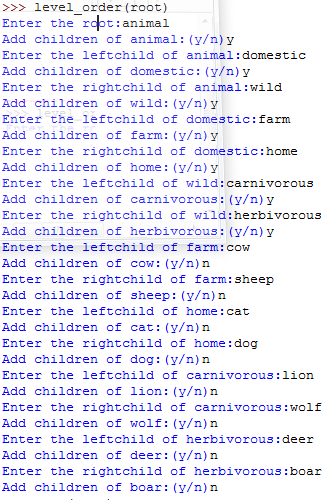


Fig. game creation

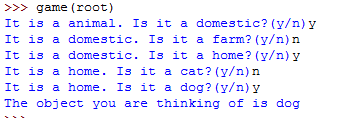


Fig. game play

Analysis:

The implemented game uses binary tree with the help of queues. The level order transversal algorithm used in both game creation and game play have time complexity O(n) and space complexity O(n).

Python Code:

#### class node():

#### def \_\_init\_\_(self):

#### self.value = None

#### self.left = None

#### self.right = None

#### self.parent = None

#### self.pos = None

#### 

#### >>> def level\_order(root):

#### queue = deque([])

#### queue.append(root)

#### while queue:

#### temp = queue.popleft()

#### if temp.parent ==None :

#### temp.value = input("Enter the root:")

#### else :

#### temp.value = input("Enter the " + temp.pos + "child of "+temp.parent+":")

#### choice = input("Add children of "+temp.value+":(y/n)")

#### if choice =='y':

#### temp.left = node()

#### temp.left.parent =temp.value

#### temp.left.pos = 'left'

#### temp.right = node()

#### temp.right.parent = temp.value

#### temp.right.pos = 'right'

#### queue.append(temp.left)

#### queue.append(temp.right)

#### 

#### >>> def game(tree):

#### queue = deque([])

#### if tree == None:

#### return

#### queue.append(tree)

#### while queue:

#### temp=queue.popleft()

#### if temp.left ==None and temp.right==None :

#### print("The object you are thinking of is "+ temp.value)

#### elif temp.left != None and temp.right != None:

#### choice = input("It is a "+temp.value+". Is it a "+temp.left.value+"?(y/n)")

#### if choice=='y':

#### queue.append(temp.left)

#### flag = True

#### else:

#### flag = False

#### if flag == False:

#### choice = input("It is a "+temp.value+". Is it a "+temp.right.value+"?(y/n)")

#### if choice == 'y':

#### queue.append(temp.right)