**DEERWALK INSTITUTE OF TECHNOLOGY**

**Lab 1**

**Artificial Intelligence**

**Submitted to: Birod Rijal**

**Submitted By: Saroj Shrestha**

**0342**

# 1. Use of Trees

Ever since the beginning of AI, there has been a great fascination in pitting the human expert against the computer. Tic-tac-toe, chess, checkers and bag-chal are some examples of “adversarial" games. Such games can be seen as a collection of board configurations (state) and valid moves linking the states. Specially, a board configuration is nothing but a node in a tree (game tree). When a player makes a move (such as move the bishop), a new state will be produced. This new state is added as the child of the previous state. The edge represents the action that produced the state. When selecting the best move, the game playing program explores as many alternative moves and consequences as possible. So there is a search procedure involved as well. However implementing a complete game tree with search is beyond the scope of this practical.

# 2. Tasks

**Animal Guessing Game**

For this practical we will implement a very simple “animal guessing game". In this game, the player thinks of an animal and the computer tries to guess which animal the player is imagining by asking questions which can be answered `yes' or `no'. To begin each round of the game, you think of an animal. For example, suppose you are thinking of a dog. The computer will generate the question at the root node “Are you a mammal?" as the first question. The user has to press 'y' or 'n' (1 or 0) as the response. Based on the response the computer will generate the second question. The game continues until the leaf node is reached. At the leaf node the computer will generate the guess (dog, mouse, trout or sparrow in our example).

# 3. Methodology

This problem has been solve by object oriented approach. Each problem is represented by an object. Node object contains the string value or question, its associated id and its left and right node.

**public class** Node {  
 Node **leftChild**,**rightChild**;  
 Integer **id**;  
 String **question**;  
 **public** Node(String question, **int** id) {  
 **this**.**question** = question;  
 **this**.**id** = id;  
 }  
}

Various nodes are required to be stored in the tree as given in the game. There is a root node and if user selects 1(yes), he/she moves to left child and if 0 (no), moves to right child.

Nodes are initialized and passed to the function where these nodes are added to respected position.

There are two functions addLeftNode () and addRightNode () that add nodes to left child and right child respectively. This approach search for every node using recursion to find parent node, if found new node is added as it's child node either left or right.

*//This functions creates and add the left child to the node having the given id***public void** addLeftNode(Node root, **int** parentId, **int** id, String question) {  
 **if** (root.**id** == parentId) {  
 Node newNode = **new** Node(question, id);  
 root.**leftChild** = newNode;  
 **return**;  
 }  
 **else** {  
 **if** (root.**leftChild** != **null**) {  
 addLeftNode(root.**leftChild**, parentId, id, question);  
 }  
 **if** (root.**rightChild** != **null**) {  
 addLeftNode(root.**rightChild**, parentId, id, question);  
 }  
  
 }  
}  
*//This functions creates and add the left child to the node having the given id***public void** addRightNode(Node root, **int** parentId, **int** id, String question) {  
 **if** (root.**id** == parentId) {  
 root.**rightChild** = **new** Node(question, id);  
 **return**;  
 } **else** {  
 **if** (root.**leftChild** != **null**) {  
 addRightNode(root.**leftChild**, parentId, id, question);  
 }  
 **if** (root.**rightChild** != **null**)  
 addRightNode(root.**rightChild**, parentId, id, question);  
 }  
  
}

After the nodes are added to the tree. We at first take the root node and display the question associated with the root node. Then if the user enter 1(YES) the left node to the current node is expanded and if the user enters 0 (NO) the right node to the current node is expanded. The expansion is carried out until we find the leaf node, and if the current node is leaf node then the value associated with this node is animal guessed by the game.

# 4. Output

We were able to solve the problem and get the desired output. Some of the screenshots of the output are provided below.

# 4th

# 2nd

# C:\Users\sabin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1st.png

# C:\Users\sabin\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3rd.png

# 4. Analysis

# 

In the given program, 1 represents Yes and 0 represents No. At first, the program asks if it is mammal or not. If mammal, there are more questions that is specific to mammal. If it is not mammal, there are questions regarding other animals apart from mammal. In this way, with series of yes/no answers we will reach to the leaf node, which is the required answer.

# 5. Annex

Implementation of this problem using Java is given here.

**package** SearchingAI;  
  
**import** java.util.Scanner;  
  
*/\*\*  
 \* Created by saroj on 3/20/2016.  
 \*/***public class** GameImpletation {  
  
 *//This function is used to add root Node* **public** Node addRootNode(String question,**int** id)  
 {  
 Node root=**new** Node(question,id);  
 **return** root;  
 }  
 *//This functions creates and add the left child to the node having the given id* **public void** addLeftNode(Node root, **int** parentId, **int** id, String question) {  
 **if** (root.**id** == parentId) {  
 Node newNode = **new** Node(question, id);  
 root.**leftChild** = newNode;  
 **return**;  
 }  
 **else** {  
 **if** (root.**leftChild** != **null**) {  
 addLeftNode(root.**leftChild**, parentId, id, question);  
 }  
 **if** (root.**rightChild** != **null**) {  
 addLeftNode(root.**rightChild**, parentId, id, question);  
 }  
  
 }  
 }  
 *//This functions creates and add the left child to the node having the given id* **public void** addRightNode(Node root, **int** parentId, **int** id, String question) {  
 **if** (root.**id** == parentId) {  
 root.**rightChild** = **new** Node(question, id);  
 **return**;  
 } **else** {  
 **if** (root.**leftChild** != **null**) {  
 addRightNode(root.**leftChild**, parentId, id, question);  
 }  
 **if** (root.**rightChild** != **null**)  
 addRightNode(root.**rightChild**, parentId, id, question);  
 }  
  
 }  
  
 **public static void** main(String[] args) {  
  
 GameImpletation addNode = **new** GameImpletation();  
 Node rootNode=addNode.addRootNode(**"Are you a mammal"**,0);*//Root node is created  
   
 //The nodes are added to the tree* addNode.addLeftNode(rootNode, 0, 1, **"Are you bigger than a cat?"**);  
 addNode.addLeftNode(rootNode, 1, 3, **"Dog"**);  
 addNode.addRightNode(rootNode, 1, 4, **"Mouse"**);  
   
 addNode.addRightNode(rootNode, 0, 2, **"Do you live underwater?"**);  
 addNode.addLeftNode(rootNode, 2, 5, **"Trout"**);  
 addNode.addRightNode(rootNode, 2, 6, **"Sparrow"**);  
  
 System.***out***.println(**"Press 1 for yes and 0 for No"**);  
 *startGame*(rootNode);*//rootnode is passed to startGame* }  
  
 *//This function check if the given node is leaf node or not* **public static boolean** checkLeaf(Node node)  
 {  
 **boolean** isLeaf=**false**;  
 **if**(node.**leftChild**==**null** && node.**rightChild**==**null**)  
 {  
 isLeaf=**true**;  
 }  
 **return** isLeaf;  
  
 }  
  
  
 **public static void** startGame(Node node) {  
 *//This loop runs until break is initiated* **while** (**true**) {  
  
 Scanner input = **new** Scanner(System.***in***);  
 System.***out***.println(node.**question**);  
 **int** answer = input.nextInt();  
 **if** (answer == 1) {  
 node=node.**leftChild**;*//switch to left child* } **else if** (answer == 0) {  
 node=node.**rightChild**;*//switch to right child* } **else** {  
 System.***out***.println(**"Invalid entry"**);  
 }  
 **if**(*checkLeaf*(node) == **true**)*//the current node is leaf node* {  
 System.***out***.println(**"The animal guessed is:\t"**+node.**question**);  
 **break**;*//Exit the loop* }  
  
 }  
 }  
}