**DEERWALK INSTITUTE OF TECHNOLOGY**

**Lab 1**

**Artificial Intelligence**

**Submitted to: Birod Rijal**

**Submitted By: Bipin Paudel**

**0327**

# 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, its associated id and it's left and right node.

Class Node{

String question

int idNumber

Node leftChild;

Node rightChild;

}

There are different nodes which should be kept in a tree according the rule of 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 addYesBySearching() and addNoBySearching() 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.

**public void** addYesBySearching(Node root, **int** parentNumber, **int** number, String question) {

**if** (root.**number** == parentNumber) {

Node newNode = **new** Node(question, number);

root.**leftChild** = newNode;

**return**;

}

**else** {

**if** (root.**leftChild** != **null**) {

addYesBySearching(root.**leftChild**, parentNumber, number, question);

}

**if** (root.**rightChild** != **null**) {

addYesBySearching(root.**rightChild**, parentNumber, number, question);

}

}

}

**public void** addNoBySearching(Node root, **int** parentNumber, **int** number, String question) {

**if** (root.**number** == parentNumber) {

root.**rightChild** = **new** Node(question, number);

**return**;

} **else** {

**if** (root.**leftChild** != **null**) {

addNoBySearching(root.**leftChild**, parentNumber, number, question);

}

**if** (root.**rightChild** != **null**)

addNoBySearching(root.**rightChild**, parentNumber, number, question);

}

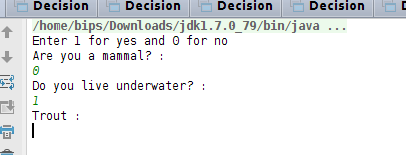
}

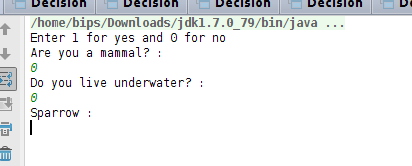
# 4. Output

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

# 

# 





# 4. Analysis

# 

As said earlier, 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. It 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, ie our answer.

# 5. Annex

Implementation of this problem using Java is given here.

**import** java.util.Scanner;

*/\*\**

*\* Created by bips on 3/26/16.*

*\*/*

**public class** Decision {

*//function to add node to yes branch ie, left branch*

**public void** addYesBySearching(Node root, **int** parentNumber, **int** number, String question) {

**if** (root.**number** == parentNumber) {

Node newNode = **new** Node(question, number);

root.**leftChild** = newNode;

**return**;

}

**else** {

**if** (root.**leftChild** != **null**) {

addYesBySearching(root.**leftChild**, parentNumber, number, question);

}

**if** (root.**rightChild** != **null**) {

addYesBySearching(root.**rightChild**, parentNumber, number, question);

}

}

}

*//function to add node to no branch ie, right branch*

**public void** addNoBySearching(Node root, **int** parentNumber, **int** number, String question) {

**if** (root.**number** == parentNumber) {

root.**rightChild** = **new** Node(question, number);

**return**;

} **else** {

**if** (root.**leftChild** != **null**) {

addNoBySearching(root.**leftChild**, parentNumber, number, question);

}

**if** (root.**rightChild** != **null**)

addNoBySearching(root.**rightChild**, parentNumber, number, question);

}

}

**public static void** main(String[] args) {

System.***out***.println(**"Enter 1 for yes and 0 for no"**);

Node rootNode = **new** Node(**"Are you a mammal?"**, 1);*//creation of root node*

Decision addItem = **new** Decision();

*//now nodes are added to the decision tree*

addItem.addYesBySearching(rootNode, 1, 2, **"Are you bigger than a cat?"**);

addItem.addNoBySearching(rootNode, 1, 3, **"Do you live underwater?"**);

addItem.addYesBySearching(rootNode, 3, 6, **"Trout"**);

addItem.addNoBySearching(rootNode, 3, 3, **"Sparrow"**);

addItem.addYesBySearching(rootNode, 2, 4, **"Dog"**);

addItem.addNoBySearching(rootNode, 2, 5, **"Mouse"**);

*//all the nodes has been addes. Now it's time to play the game*

*questionSession*(rootNode);*// this function asks user various questions from the node.*

}

**public static void** questionSession(Node rootNode) {

Scanner input = **new** Scanner(System.***in***);

System.***out***.println(rootNode.toString());

**int** answer = input.nextInt();

**if** (answer == 1) {

**if** (rootNode.**leftChild** == **null**) {

**return**;

}

*questionSession*(rootNode.**leftChild**);

} **else if** (answer == 0) {

**if** (rootNode.**rightChild** == **null**) **return**;

*questionSession*(rootNode.**rightChild**);

}

**else**{

System.***out***.println(**"Enter only 1 or 0, Try again"**);

*questionSession*(rootNode);

}

}

}

*// Node is a class that contains information about a node.*

**class** Node {

String **question**;*//it is a string value associated with each node*

**int number**;*//it is id number for that node*

Node **leftChild**;

Node **rightChild**;

**public** Node(String question, **int** number) {

**this**.**question** = question;

**this**.**number** = number;

}

@Override

**public** String toString() {

**return question** + **" :"**;

}

}

DEERWALK INSTITUTE OF TECHNOLOGY

Simulation and modeling

Lab

Submitted to: Binod Sitaula

Submitted by: Bipin Paudel

0327