**Lab 1**

WAP to guess the animal name

**Background:**

This game is played between program and user where the user is asked question about the different nature and behavior of the animal and the computer has to guess the animal that the user is thinking of. Since it is an AI and machine learning based program, the program has to learn about various different characteristics of the animal which it doesn’t have the knowledge about.

The program is implemented using binary search tree. The binary search tree is used to search the correct answer in the leaf of the tree. The non-leaf nodes are the question and the leaf gives the answer of the question. The questions that are added to train the program are stored in the leaf.

**Methodology:**

This program is written in Java and is implemented using different classes and methods. Various techniques of data structure and algorithm is also implemented in this program. The questions are stored in this program. If the question asked are answered correctly other different questions are asked as well to make sure the guess is right and other question are asked to be added if the guesses are wrong till the program is terminated. The questions are in the form of yes/no questions. The program thinks that the guess are right if the questions asked are answered with yes and if the user answers no then the program asks several other questions and at last after compiling all the answers, the program guesses the answers the guess and if it is unable to do so its ask the user to enter several other question for the guess.

For example

If a user has a guess of lion there are pre fitted questions in the program. If the questions asked meet the requirement for the guess to be a lion, the guess is right else other questions are asked and also other questions are to be input by the user if the guess is wrong.

**Program:**

import java.util.Scanner;

public class labOne {

private static Scanner stdin;

public labOne() {

}

public static void main(String[] args) {

instruct();

labOne.BTNode root = beginningTree();

do {

play(root);

} while(query("Lets Play A Game"));

System.out.println("Thank You Senpai");

}

public static void instruct() {

System.out.println("Guess an Animal");

System.out.println("Yes/No Question will be asked");

System.out.println("to recognize the Animal");

}

public static void play(labOne.BTNode<String> current) {

while(!current.isLeaf()) {

if(query((String)current.getData())) {

current = current.getLeft();

} else {

current = current.getRight();

}

}

System.out.print("My guesses Are " + (String)current.getData() + ". ");

if(!query("Is it correct?")) {

learn(current);

} else {

System.out.println("Hah! I am always correct");

}

}

public static labOne.BTNode<String> beginningTree() {

String ROOT\_QUESTION = "Do you have whiskers";

String LEFT\_QUESTION = "Can you fly?";

String RIGHT\_QUESTION = "Do you live underwater?";

String ANIMAL1 = "Cat";

String ANIMAL2 = "Pigeon";

String ANIMAL3 = "Fish";

String ANIMAL4 = "Lion";

labOne.BTNode root = new labOne.BTNode("Do you have whiskers", (labOne.BTNode)null, (labOne.BTNode)null);

labOne.BTNode child = new labOne.BTNode("Can you fly?", (labOne.BTNode)null, (labOne.BTNode)null);

child.setLeft(new labOne.BTNode("Cat", (labOne.BTNode)null, (labOne.BTNode)null));

child.setRight(new labOne.BTNode("Pigeon", (labOne.BTNode)null, (labOne.BTNode)null));

root.setLeft(child);

child = new labOne.BTNode("Do you live underwater?", (labOne.BTNode)null, (labOne.BTNode)null);

child.setLeft(new labOne.BTNode("Fish", (labOne.BTNode)null, (labOne.BTNode)null));

child.setRight(new labOne.BTNode("Lion", (labOne.BTNode)null, (labOne.BTNode)null));

root.setRight(child);

return root;

}

public static void learn(labOne.BTNode<String> current) {

String guessAnimal = (String)current.getData();

System.out.println("Senpai i cannot guess it,can you help me? ");

String correctAnimal = stdin.nextLine();

System.out.println("Please type a yes/no question that will distinguish a");

System.out.println(correctAnimal + " from a " + guessAnimal + ".");

String newQuestion = stdin.nextLine();

current.setData(newQuestion);

System.out.println("As a " + correctAnimal + ", " + newQuestion);

if(query("Please answer")) {

current.setLeft(new labOne.BTNode(correctAnimal, (labOne.BTNode)null, (labOne.BTNode)null));

current.setRight(new labOne.BTNode(guessAnimal, (labOne.BTNode)null, (labOne.BTNode)null));

} else {

current.setLeft(new labOne.BTNode(guessAnimal, (labOne.BTNode)null, (labOne.BTNode)null));

current.setRight(new labOne.BTNode(correctAnimal, (labOne.BTNode)null, (labOne.BTNode)null));

}

}

public static boolean query(String prompt) {

System.out.print(prompt + " [Y or N]: ");

String answer;

for(answer = stdin.nextLine().toUpperCase(); !answer.startsWith("Y") && !answer.startsWith("N"); answer = stdin.nextLine().toUpperCase()) {

System.out.print("Invalid response. Please type Y or N: ");

}

return answer.startsWith("Y");

}

static {

stdin = new Scanner(System.in);

}

public static class BTNode<E> {

private E data;

private labOne.BTNode<E> left;

private labOne.BTNode<E> right;

public BTNode(E initialData, labOne.BTNode<E> initialLeft, labOne.BTNode<E> initialRight) {

this.data = initialData;

this.left = initialLeft;

this.right = initialRight;

}

public E getData() {

return this.data;

}

public labOne.BTNode<E> getLeft() {

return this.left;

}

public E getLeftmostData() {

return this.left == null?this.data:this.left.getLeftmostData();

}

public labOne.BTNode<E> getRight() {

return this.right;

}

public E getRightmostData() {

return this.left == null?this.data:this.left.getRightmostData();

}

public void inorderPrint() {

if(this.left != null) {

this.left.inorderPrint();

}

System.out.println(this.data);

if(this.right != null) {

this.right.inorderPrint();

}

}

public boolean isLeaf() {

return this.left == null && this.right == null;

}

public void preorderPrint() {

System.out.println(this.data);

if(this.left != null) {

this.left.preorderPrint();

}

if(this.right != null) {

this.right.preorderPrint();

}

}

public void postorderPrint() {

if(this.left != null) {

this.left.postorderPrint();

}

if(this.right != null) {

this.right.postorderPrint();

}

System.out.println(this.data);

}

public void print(int depth) {

int i;

for(i = 1; i <= depth; ++i) {

System.out.print(" ");

}

System.out.println(this.data);

if(this.left != null) {

this.left.print(depth + 1);

} else if(this.right != null) {

for(i = 1; i <= depth + 1; ++i) {

System.out.print(" ");

}

System.out.println("--");

}

if(this.right != null) {

this.right.print(depth + 1);

} else if(this.left != null) {

for(i = 1; i <= depth + 1; ++i) {

System.out.print(" ");

}

System.out.println("--");

}

}

public labOne.BTNode<E> removeLeftmost() {

if(this.left == null) {

return this.right;

} else {

this.left = this.left.removeLeftmost();

return this;

}

}

public labOne.BTNode<E> removeRightmost() {

if(this.right == null) {

return this.left;

} else {

this.right = this.right.removeRightmost();

return this;

}

}

public void setData(E newData) {

this.data = newData;

}

public void setLeft(labOne.BTNode<E> newLeft) {

this.left = newLeft;

}

public void setRight(labOne.BTNode<E> newRight) {

this.right = newRight;

}

public static <E> labOne.BTNode<E> treeCopy(labOne.BTNode<E> source) {

if(source == null) {

return null;

} else {

labOne.BTNode leftCopy = treeCopy(source.left);

labOne.BTNode rightCopy = treeCopy(source.right);

return new labOne.BTNode(source.data, leftCopy, rightCopy);

}

}

public static <E> long treeSize(labOne.BTNode<E> root) {

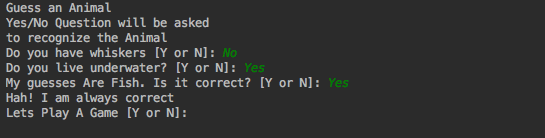
return root == null?0L:1L + treeSize(root.left) + treeSize(root.right);

}

}

}

**Output:**

****