Work on project. Stage 3/6: Guessing game

Project: Guess the Animal

Guessing game

Hard 10 minutes **93** users solved this stage. Latest completion was **about 20 hours ago**.

Description

It's time to play! The program prompts the user to think of an animal and then tries to guess what it is. If the computer fails, the program should ask the user what this animal is and what statement can help distinguish it from another one. This knowledge is supposed to be stored in a form of a binary tree and kept in memory only for the duration of the game session.

4 / 4 Prerequisites

- Graph 4
- Recursion basics 4
- Tree 4
- Binary search tree 4

Join a study group for the project Guess the Animal

Discuss your current project with fellow learners and help each other.

Objectives

The program should ask the user about their favorite animal. This animal name will be the root node in our knowledge tree. When the computer starts the game, it will ask questions starting from the top, that is, the root node. If the computer makes a wrong guess, it should ask the user two questions: first, what animal the user had in mind, and second, what statement can help the computer distinguish the animal it guessed (old) from the animal that the person actually thought of (new). The program should clarify whether that fact is correct for the new animal. After that, the name of the "old" animal in the tree is replaced with the new statement, and two new leaves are added to this node: one with the "old" animal and another with the "new" animal.

Let's look at an example so that you can better visualize the game process.

When the program starts and prompts the user for their favorite animal, the user replies that it is a cat.

```
    |
 ( Cat )
   | |
null null
```

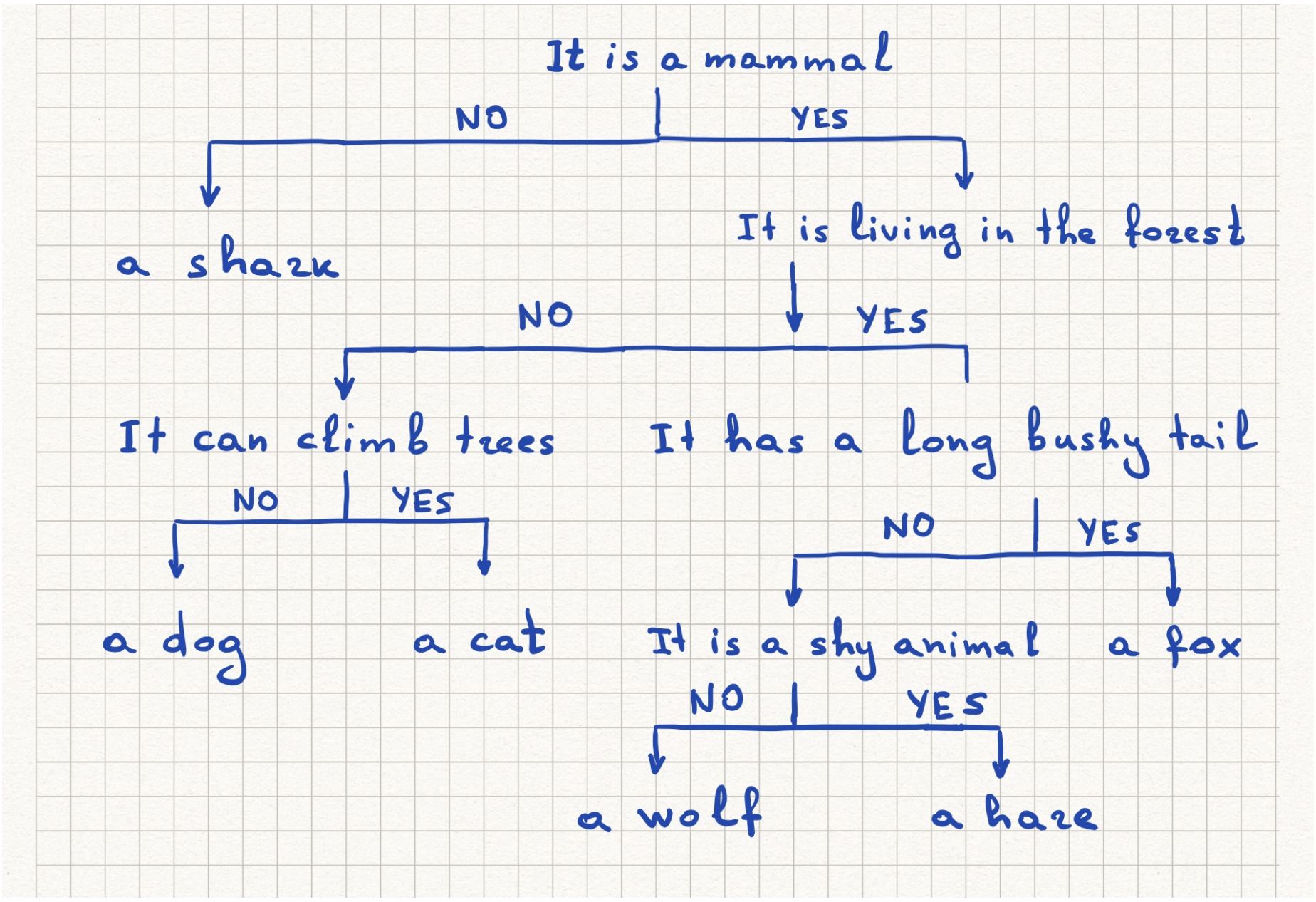
We get a binary tree where the root node is unique and has no children. Such a node is called a **leaf**.

Suppose that then the user has thought of a dog. To the computer's question "Is it a cat?", they answer negatively. The user then enters the name of the intended animal, "dog", and the statement "It can climb trees". The correct answer for a dog is "no". Knowing this, the computer builds a new tree:

```
  ( Can it climbs trees? )
     |                 |
    yes                no
     |                 |
  ( Cat )           ( Dog )
    | |               | |
 null null         null null
```

In the new tree, the root element is the statement, and the node has two children: the names of the animals.

So, if the program does not guess the animal correctly, two items are added to the knowledge tree: the name of the new animal and a fact that distinguishes one animal from the other. After some game time, the knowledge tree could look something like this:



At the end of each session, the program should ask the user if they want to keep playing or quit the game. In this stage, we are not yet saving the knowledge tree to the disk, so when the user decides to finish the game, the built knowledge tree will disappear. Don't worry: in the next step, we will learn how to save it.

Example

The greater-than symbol followed by a space `> ` represents the user input. Note that it's not part of the input.

```
Hi, Early Bird!

I want to learn about animals.
Which animal do you like most?
> cat
Wonderful! I've learned so much about animals!
Let's play a game!
You think of an animal, and I guess it.
Press enter when you're ready.
>
Is it a cat?
> No
I give up. What animal do you have in mind?
> a shark
Specify a fact that distinguishes a cat from a shark.
The sentence should satisfy one of the following templates:
- It can ...
- It has ...
- It is a/an ...

> It is a mammal
Is the statement correct for a shark?
> No
I have learned the following facts about animals:
 - The cat is a mammal.
 - The shark isn't a mammal.
I can distinguish these animals by asking the question:
 - Is it a mammal?
Nice! I've learned so much about animals!

Would you like to play again?
> No

Have a nice day!
```

Report a typo

Write a program

Code Editor IDE

CONNECTION STATUS

IDE is responding IntelliJ IDEA 2022.1.1

EduTools plugin is responding 2022.5-2022.1-343

Solve in IDE Synchronizing IDE may take a while

src/animals/Main.java

```
package animals;

import java.time.LocalTime;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.Scanner;
import java.util.regex.Pattern;

public class Main {
    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        Game game = new Game();
        System.out.println(game.checkTimeOfDay() + "\n");

        System.out.println("I want to learn about animals.\n" +
                "Which animal do you like most?");
        String firstAnimal = game.getAnimal(scanner.nextLine().toLowerCase().trim());
        Node root = new Node(firstAnimal);
        game.addNode(root);
        System.out.println("Wonderful! I've learned so much about animals!");
        do {
            System.out.println("Let's play a game!\n" +
                    "You think of an animal, and I guess it.\n" +
                    "Press enter when you're ready.");
            scanner.nextLine();
            root.check();
            System.out.println("Would you like to play again?");
        } while (game.askYesNo());
        System.out.println("Bye");
    }
}

class Node {
    private static final Scanner scanner = new Scanner(System.in);

    private static final Pattern STATEMENT = Pattern.compile("it (can|has|is) .+");
    Game game = new Game();
    private boolean isLeaf;
    String content;
    Node left;
    Node right;

    Node(String content) {
        this.content = content;
        left = null;
        right = null;
        isLeaf = true;
    }

    public String getQuestion(String statement) {
        String question = statement.startsWith("it has")
                ? statement.replaceFirst("it has", "does it have")
                : statement.replaceFirst("it (can|is)", "$1 it");
        return capitalize(question) + "?";
    }

    public String askQ() {
        if (isLeaf) {
            return "Is it " + content + "?";
        }
        return content;
    }

    private static String capitalize(String sentence) {
        return sentence.substring(0, 1).toUpperCase() + sentence.substring(1).toLowerCase();
    }
```

```
String getContent() {
    return content;
}

void check() {

    System.out.println(askQ());
    if (!this.isLeaf) {
        if (game.askYesNo()) {
            this.left.check();
        } else {
            this.right.check();
        }
    } else {
        if (!game.askYesNo()) {
            String firstAnimal = this.

                    getContent();
            System.out.println("I give up. What animal do you have in mind?");
            String userNewAnimal = game.getAnimal(scanner.nextLine().strip().toLowerCase());

            String positive = getStatement(firstAnimal, userNewAnimal);
            System.out.println("Is it correct for " + userNewAnimal + "?");

            String learn1 = toAnimalFact(positive, firstAnimal);
            String learn2 = toAnimalFact(positive, userNewAnimal);
            Node secondNode = new Node(userNewAnimal);
            if (!game.askYesNo()) {
                this.left = new Node(this.content);
                this.content = getQuestion(positive);
                this.right = secondNode;
                this.isLeaf = false;
                System.out.println("I have learned the following facts about animals:\n" +
                        "- " + learn1 + "\n" +
                        "- " +
                        learn2.replace("can", "can't")
                                .replace("has", "doesn't have")
                                .replace("is", "isn't")
                        + "\n" +
                        "I can distinguish these animals by asking the question:\n" +
                        "- " + getQuestion(positive));
            } else {
                this.right = new Node(this.content);
                this.content = getQuestion(positive);
                this.left = secondNode;
                this.isLeaf = false;
                System.out.println("I have learned the following facts about animals:\n" +
                        "- " + learn1.replace("can", "can't")
                        .replace("has", "doesn't have")
                        .replace("is", "isn't") + "\n" +
                        "- " +
                        learn2
                        + "\n" +
                        "I can distinguish these animals by asking the question:\n" +
                        "- " + getQuestion(positive));
            }
        }
    }
}


private static String getStatement(String first, String second) {
    while (true) {
        System.out.printf("Specify a fact that distinguishes %s from %s.%n" +
                "The sentence should satisfy one of the following templates:%n" +
                "- It can ...%n- It has ...%n- It is a/an ...%n", first, second);
        String response = scanner.nextLine().toLowerCase().trim();
        if (STATEMENT.matcher(response).matches()) {
            return response.replaceFirst("(.+)\\.+", "$1");
        }
        System.out.printf("The examples of a statement:%n" +
```

```
                    " - It can fly%n - It has horn%n - It is a mammal%n");
        }
    }

    public String toAnimalFact(String fact, String animal) {
        return fact.replaceFirst("it", animal.replaceFirst("an?", "The")) + ".";
    }
}

class Game {
    private static final Scanner scanner = new Scanner(System.in);
    private static final Pattern POSITIVE = Pattern.compile(
            "(y|yes|yeah|yep|sure|right|affirmative|correct|indeed|you bet|exactly|you said it)[.!]?",
Pattern.CASE_INSENSITIVE);
    private static final Pattern NEGATIVE = Pattern.compile(
            "(n|no( way)?|nah|nope|negative|i don't think so|yeah no)[.!]?", Pattern.CASE_INSENSITIVE);
    private static final LocalTime MORNING_START = LocalTime.of(5, 1);
    private static final LocalTime MORNING_END = LocalTime.of(12, 0);
    private static final LocalTime DAY_START = LocalTime.of(12, 1);
    private static final LocalTime DAY_END = LocalTime.of(18, 0);
    private static final Map<String, String> POS_TO_NEG = Map.of("it can", "it can't",
            "it has", "it doesn't have", "it is", "it isn't");

    private final String[] phrasesBye = {"Have a nice day!", "See you soon!", "Bye!"};

    private final List<Node> nodes = new ArrayList<>();

    public void addNode(Node node) {
        nodes.add(node);
    }

    public String getAnimal(String animal) {
        if (!animal.matches("an? .+")) {
            var isVowel = animal.matches("(the )?[aeiou].+");
            var article = isVowel ? "an " : "a ";
            animal = animal.replaceFirst("(the )?", article);
        }
        return animal;
    }

    public String checkTimeOfDay() {
        LocalTime now = LocalTime.now();
        if (now.isAfter(MORNING_START) && now.isBefore(MORNING_END)) {
            return "Good morning!";
        } else if (now.isAfter(DAY_START) && now.isBefore(DAY_END)) {
            return "Good day!";
        } else {
            return "Good evening!";
        }
    }

    public boolean askYesNo() {
        while (true) {
            String answer = scanner.nextLine().toLowerCase().trim();
            if (POSITIVE.matcher(answer).matches()) {
                return true;
            }
            if (NEGATIVE.matcher(answer).matches()) {
                return false;
            }
            System.out.println("Come on, yes or no?");
        }
    }
}
```

✓ **Correct.**

Great job!

9 users liked this problem. 0 didn't like it. **What about you?**

Continue

Solutions (22)

Comments (13) Hints (0) Useful links (0) Solutions (22) Show discussion