Recursive vs. Iterative Traversal

Introduction

3 min

In this lesson, you will learn how to implement a recursive solution to a linked list search. The

Preview: Docs Loading link description

method

accepts a value as input and recursively checks each node in the linked list, until the node of interest is found. If it is found, the method should return the node. Otherwise, it should return null.

Before you begin, let's take a look at how we can search for an element in a linked list using an iterative approach. The code below is taken from the LinkedList() class in **LinkedList.js**.

```
findNodeIteratively(data) {
  let currentNode = this.head;
  while (currentNode !== null) {
    if (currentNode.data === data) {
      return currentNode;
    }
    currentNode = currentNode.next;
  }
  return null;
}
```

The method starts at the head of the linked list and checks if the input data is equal to the data

Preview: Docs Loading link description

parameter

at the head. The method continues to iterate through the linked list until the node is found or the end of the list is reached.

Instructions

1. Checkpoint 1 Passed

1.

In **index.js**, use .findNodeIteratively() to find the Node in myList with data equal to 'Node 3'.

Save the returned value to foundNode, then log it to the console.

Hint

From **index.js**, use the following syntax:

myList.findNodeIteratively('Node');

After saving the value to foundNode, log it to the console with the following syntax:

console.log(foundNode);

index.js

```
const Node = require('./Node');
const LinkedList = require('./LinkedList');

const myList = new LinkedList();

myList.addToHead('Node 1');
myList.addToHead('Node 2');
myList.addToHead('Node 3');
myList.addToHead('Node 4');

// Add code below
const foundNode = myList.findNodeIteratively('Node 3');

console.log(foundNode);
```

Node.js

```
class Node {
 constructor(data) {
  this.data = data;
  this.next = null;
 }

 setNextNode(node) {
  if (!(node instanceof Node)) {
   throw new Error('Next node must be a member of the Node class');
  }
  this.next = node;
 }

 setNext(data) {
```

```
    this.next = data;
  }

  getNextNode() {
    return this.next;
  }
}

module.exports = Node;
```

LinkedList.js

```
const Node = require('./Node');

class LinkedList {
  constructor() {
    this.head = null;
  }

  addToHead(data) {
    const nextNode = new Node(data);
    const currentHead = this.head;
    this.head = nextNode;
    if (currentHead) {
      this.head.setNextNode(currentHead);
    }
  }

  addToTail(data) {
    let lastNode = this.head;
    if (!lastNode) {
      this.head = new Node(data);
```

```
  } else {
    let temp = this.head;
    while (temp.getNextNode() !== null) {
      temp = temp.getNextNode();
    }
    temp.setNextNode(new Node(data));
  }
}

removeHead() {
  const removedHead = this.head;
  if (!removedHead) {
    return;
  }
  if (removedHead.next) {
    this.head = removedHead.next;
  }
  return removedHead.data;
}

printList() {
  let currentNode = this.head;
  let output = '<head> ';
  while (currentNode !== null) {
    output += currentNode.data + ' ';
    currentNode = currentNode.next;
  }
  output = output.concat("<tail>");
  console.log(output);
}

findNodeIteratively(data) {
```

```
    let currentNode = this.head;
    while (currentNode !== null) {
      if (currentNode.data === data) {
        return currentNode;
      }
      currentNode = currentNode.next;
    }
    return null;
  }

  findNodeRecursively(data, currentNode = this.head) {

  }

}

module.exports = LinkedList;
```