Recursive Tracing

Hitesh Boinpally
Summer 2023
Agenda

• Recursion Intro
• Practice
• Visualizations
• Reminders
Agenda

• Recursion Intro
• Practice
• Visualizations
• Reminders
Road Map - Recursion

- **Friday (Today)**
  - Introduce idea of "recursion"
  - Goal: Understand idea of recursion and read recursive code
- **Tuesday**
  - Practice reading recursive code
- **Wednesday**
  - More complex recursive examples
  - Goal: Identify recursive structure in problems and write recursive code
- **Thursday**
  - Practice writing recursive code
Road Map - Recursion

- **Friday (Today)**
  - Introduce idea of “recursion”
  - Goal: Understand idea of recursion and read recursive code

- **Tuesday**
  - Practice reading recursive code

- **Wednesday**
  - More complex recursive examples
  - Goal: Identify recursive structure in problems and write recursive code

- **Thursday**
  - Practice writing recursive code
Recursion Intro
Recursion Intro

- **Recursion**: Defining something in terms of itself
- **Recursive Programming**: Writing methods that call themselves to solve problems
  - Helpful to solve specific problems
  - Equally as powerful as iterative solutions
Recursive Programming

Two cases to always keep in mind:

1. **Base Case:**
   - Stopping point, how to know we’re “done”
   - Easiest / smallest thing to calculate

2. **Recursive Case:**
   - Do “one step” of the problem
   - Pass on the work to the next method call

• Some problems may have multiple base / recursive cases!
Agenda

• Recursion Intro
• Practice
• Visualizations
• Reminders
public static int recursiveMystery(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * recursiveMystery(n - 1);
    }
}
Recursive Tracing Practice

```java
public static int recursiveMystery(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * recursiveMystery(n - 1);
    }
}
```

Think of some example calls. What do they execute?

- `recursiveMystery(0)`
- `recursiveMystery(2)`
- `recursiveMystery(4)`
Recursive Tracing Practice

public static int recursiveMystery(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * recursiveMystery(n - 1);
    }
}

recursiveMystery(4):
Agenda

• Recursion Intro
• Practice
• Visualizations
• Reminders
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
reverse Visualization

```java
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“1. Suits”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place
reverse Visualization

```java
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“1. Suits”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“2. Barry”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place
reverse Visualization

```java
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“2. Barry”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place
reverse Visualization

```
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“3. Modern Family”</td>
</tr>
</tbody>
</table>
reverse Visualization

```
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“4. The Good Place”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place
reverse Visualization

```java
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place
reverse Visualization

```java
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“4. The Good Place”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place

Console output:
4. The Good Place
reverse Visualization

```java
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“3. Modern Family”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place

Console output:
4. The Good Place
3. Modern Family
reverse Visualization

```java
public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}
```

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“2. Barry”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place

Console output:
4. The Good Place
3. Modern Family
2. Barry
reverse Visualization

public static void reverse(Scanner file) {
    if (file.hasNextLine()) {
        String text = file.nextLine();
        reverse(file);
        System.out.println(text);
    }
}

<table>
<thead>
<tr>
<th>file</th>
<th>Reference to Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>“1. Suits”</td>
</tr>
</tbody>
</table>

Contents of file:
1. Suits
2. Barry
3. Modern Family
4. The Good Place

Console output:
4. The Good Place
3. Modern Family
2. Barry
1. Suits
Agenda

• Recursion Intro
• Practice
• Visualizations
• Reminders
Reminders

• Resub 0 Due tonight
  • Updated to allow for C0 submissions
Reminders

• Resub 0 Due tonight
  • Updated to allow for C0 submissions
• Quiz 0 **Monday (7/10)**
  • Logistic details to be posted today
  • Topics: Linked Nodes, Linked Lists
  • Take-home, open 8:30am – 11:59pm
  • Will get instant feedback
  • Open collaboration (with a caveat), Open note
  • Reach out ASAP with any extenuating circumstances