Lesson 8.01 — Thinking Recursively

Overview

Objectives — ***Students will be able to…***

- **Define** recursion.

Assessments — ***Students will…***

- **Describe** recursive methods
- **Compare** iterative and recursive methods during a class discussion

Homework — ***Students will…***

- **Read** the rest of BJP 12.1

Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Bookmarks** on each computer (or URL on projector) for (http://www.softschools.com/games/logic_games/tower_of_han

You should familiarize yourself with the Tower of Hanoi as needed. The (exhaustive) Wikipedia article covers the different algorithms that may be used to solve the problem (here: http://en.wikipedia.org/wiki/Tower_of_Hanoi). If you are pressed for time, you should read the introduction, origins, iterative and recursive solutions only. The story/legend associated with the puzzle is summarized here: (http://www.qef.com/oise/hanoi/hanoi01.html).

Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Activity: Tower of Hanoi Game	15min
Class Discussion & Solutions	20min
Lecture	10min

Procedure

Your hook today is to invite students to play the Tower of Hanoi game as soon as they enter the classroom. You should not offer any clues or hints as to why they are playing the game; only supply the information outlined in the "Activity" section below.

Bell-work and Attendance [5 minutes]

Activity: Tower of Hanoi Game [15 minutes]

1. Have students navigate to the Tower of Hanoi game at the link listed in "Materials & Prep".
2. Tell students that they have 15 minutes to win the game.
    - As they play the game, they should take notes on what strategies have worked and which ones have failed. (Point out that they are devising their own algorithms.)
    - They are not allowed to navigate to any other website. (If you do catch students cheating, it's actually a bonus! They're taking the initiative to learn and understand an algorithm that uses recursion J)
3. If students finish early:
    - Invite them to try to optimize their solution
    - Invite them to try the game using a larger stack of rings.

4. After 10–15 minutes of play (or sooner, if you have a particularly advanced class), call the class back together for a whole group discussion.

Class Discussion & Solutions [20 minutes]

1. Begin the group discussion with the following series of warm-up questions. Use these soft-ball questions to engage students who are usually shy or quiet in class.
   - Did you enjoy the game?
   - Did you win/solve the game?
   - How long did it take you to solve the game?
   - Have any of you ever played this game before? Have you heard of this game before?
2. If your class is interested in such things, explain to them the history of the game and its initial legend (this information can be found following the links in "Materials & Prep.")
3. Ask your students about their solutions (or non-solutions, if no one has correctly solved the problem in the time allotted).
   - What steps do you take to solve the problem? What does your algorithm look like?
     - If students suggest an iterative solution (for an example, refer to the Wikipedia page), point out that there is a faster/more efficient algorithm they can use.
     - If a student suggests a recursive solution (if you need an example, check the Wikipedia page), point out that this solution is the fastest solution to the problem. In the world of computing, that's a great thing! But in the story about the Buddhist monks, we probably wouldn't want to share this solution with them.
   - Did anyone use a different set of steps or rules (heuristics) to solve the problem?
   - As a design approach, what should be our first step in thinking about solving this solution? (Can we break the problem down into smaller problems that are easier to solve?)
4. Once you've completed a discussion about the algorithms' students used, watch the animation as a class, asking students to point out what patterns they see in the solution.
   - Link to animation is here: http://www.eisbox.net/blog/2009/04/06/tower-of-hanoi-animation/
   - A less attractive, but easier to follow animation can be found here: https://www.youtube.com/channel/UCMDuzeB8MqT_-AwFGT8qQ-g/search?query=Hanoi

Lecture [10 minutes]

1. Provide students with the following definitions:
   - **Iteration/Iterative**: A programming technique in which you describe actions to be repeated typically using a loop.
   - **Recursion/Recursive**: A programming technique in which you describe actions to be repeated using a method that calls itself.
2. Ask students which algorithms from your class discussion were iterative, and which ones were recursive. You should see if students can come up with a "rule of thumb" that helps them decide when a method is recursive instead of iterative. (The key is that the method will call itself in the method body!)
3. If time allows, watch the animation again, and ask students to narrate the animation using recursive pseudocode. (Correct answers will include rules like "To move a pile of height N from A to C, move the pile of height N-1 from A to B, move the Nth disc from A to C, then move the pile of height N-1 from B to C." This is a tough problem to articulate—the important part is that students understand that breaking the problem into smaller parts is the key to finding the simplest problem to solve (next lesson will discuss the "base case.")

Be sure to spot check students for understanding during the class discussion and note-taking session. If you would rather assess students formally, give students a quick ticket-to-leave assignment: have them hand you a piece of paper with their name and their definition of recursion as they walk out the door.

Accommodation and Differentiation

If students need additional modeling of iterative and recursive methods, review these examples outlined in the book chapter:

- The writeStars program takes an integer parameter n and produces a line of output with n stars on it. Here's the iterative solution (you know it's iterative because it uses a loop):

  ```
  public static void writeStars (int n) {
      for (int i = 1; i <=n; i++) {
          System.out.print("*");
      }
      System.out.println();
  }
  ```

- This same program can be written using a recursive version of the writeStars method (we know it's a recursive method because it calls itself in the method body):

  ```
  public static void writeStars (int n) {
      if (n==0) {
          System.out.println();
      } else {
          System.out.print("*");
          writeStars(n-1);
      }
  }
  ```

If your students are speeding through this lesson, ask them to try out the algorithms proposed during the class discussion. Have students track how many moves each approach takes, and have your students explain if a recursive method is faster.

For a broader discussion of self-reference in other contexts (language, art, and literature) get students thinking about terms like "never again" or play Carli Simon's "You're So Vain" (https://youtu.be/b6UAYGxiRwU).

Teacher Prior CS Knowledge

Many problems can be solved with either iteration or recursion. In general, iterative solutions run faster than recursive solutions because the recursive call has the added overhead of making a function call. In addition, there is a memory cost to each function call. Many compilers can optimize certain types of recursive algorithms like tail recursion to minimize the difference in execution time and memory usage.

Misconceptions

- When the base case is met execution of the recursive call ends and execution continues from the initial recursive call. This may be conceptually true for tail recursion but although desirable, not all recursive calls conform to this pattern.

Common Mistakes

Recursion common mistakes: http://interactivepython.org/runestone/static/JavaReview/Recursion/rMistakes.html

Video

- BJP 12-1, *Recursive Tracing* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c12-1
- CS Homework Bytes, *Recursion, with Maxine* https://www.youtube.com/watch?v=a2Op-yPcm-A
- CSE 143, *Recursion* (0:46–43:18) https://www.youtube.com/watch?v=cRjRrOvBonQ&start=46

Forum discussion

Lesson 8.01 Thinking Recursively (TEALS Discourse account required)