IbraFSG™ 5 - Week 9; BSTs and Tree Traversals

Ibrahim Chehab

UTM RGASC

March 12, 2024

Table of Contents

Welcome back to IbraFSGs™

- Welcome back to IbraFSGs™! Hello to new people and welcome back to tenured members.
- This week we will be further discussing Trees and Recurisve Data Structures, and their relations to *Binary Search Trees* (BSTs) and *Tree Traversals*
- Trees are a main character in the remainder of the course, and will reappear in CSC236 and CSC263.
- There are several types of trees:
  - Binary Trees/Binary Search Trees (BSTs)
  - AVL Trees [Also known as a self-balancing BST]
  - Abstract Syntax Trees (ASTs) [also known as Expression Trees]
  - Heaps
- CSC148 Only Focuses on Binary Trees, BSTs, ASTs, and Traditional Trees
  - On my exam last year, we had a question about cycles
  - The idea is, since you have an introductory idea of basic graphs (through trees), cycles are a natural extension of that - And thus fair game*

**Will they give you a question about cycles? Probably not, but it's good to know the concept.*

FSG HouseKeeping

1. **Reminder that a new FSG Session has been added:** *Fridays 12:00-1:00 in IB210 Note: This session is not run by myself, rather another FSG Leader. It is still a very useful resource nonetheless*
2. **The Second Midterm is coming up!** That's right! The second midterm is approaching us quickly (March 25^{th} 2024) Better start studying, maybe with...
3. ... **The Second Midterm FSG Is Coming Soon!** Stay tuned for more details!
4. **Reminder that FSG Slides Are Posted!** That's right! All FSG Slides are posted on my GitHub page. You can find them here: https://github.com/IbraTech04/FSG-Slides

FSG HouseKeeping (Cont'd)

QR Code for FSG Slides:

https://github.com/IbraTech04/FSG-Slides

FSG HouseKeeping (Cont'd)

6. **New Study Session Drop Ins!**
   Feeling Stressed about exams? Don't know where to start? Drop by the MN3260 for a study skills session!
   - **When:** March 20^{th} 10-11AM, March 27^{th} 3-4PM, and April 3^{rd} 1-2PM
   - **Where:** MN3260

A Recap of the UltraSheet™

- An *UltraSheet™* is a "cheat sheet" that you compile for **yourself** to review course materials
  - Sharing UltraSheets™ is **counter-productive** and **will not help you learn the material**
  - However, reviewing content in a group and simultaneously updating your UltraSheets™ is a good idea
- It acts like your own personalized textbook chapter
  - It allows you to **regurgitate all the course information in a contigous, organized manner** and helps you **find gaps in your knowledge**
- UltraSheets™ help with type 1 and 2 questions
  - Can you remember what type 1 and 2 questions are?
  - Can you remember what a *type 3* question is? :troll:

Key Terms

Required Key Terms: The following key terms are required for this week's content. You should be able to define and explain these terms in your UltraSheets™:

- **Pre-order Traversal**
- **Post-order Traversal**
- **Level-order Traversal**
- **In-order Traversal**
  - **Note:** In-order Traversal is the most important traversal for BSTs
  - *Can In-Order Traversal be done on a non-BT? Why or why not?*
- **Binary Search Tree (BST):** A special kind of binary tree that has the following properties: **The four properties of a BST:**
  - The left subtree of a node contains only nodes with keys less than the node's key
  - The right subtree of a node contains only nodes with keys greater than the node's key
  - The left and right subtree each must also be a binary search tree - If the `root` is not `None`
  - Each subtree must have **at most** two children
- Note: Duplicates in Binary Trees usually depend on the implementation
  - In your case; You would need to check the question carefully to see if duplicates are allowed, and whether or not that will affect your traversal
  - Common Places to check: Preconditions, Rerepsentation Invariants, context-cues in the question

Key Terms (Cont'd)

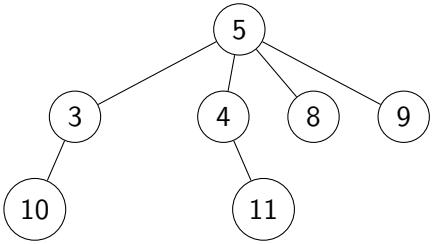
Suggested Key Terms: The following key terms are *recommended* for further studying this week's content. Most are out of the scope of the course and are **not** required, but are fun for extra learning:

- **AVL Tree:** A special kind of BST that is self-balancing - Guaranatees $\mathcal{O}(\log n)$ time complexity for all operations
  - **Why is this important?** Keep this in the back of your head for an upcoming activity ;D
- **Balancing Factor:** Useful in CSC263 when talking about *AVL Trees*. Denotes the difference in height between the left and right subtrees of a node, and is used to determine what *rotation* is required to re-balance the tree
  - Don't worry about this if it doesn't make sense;
  - Again, it's out of the scope of the course. Just a fun fact **if** you want to learn more
- **Breadth-first Search (BFS):** A graph traversal algorithm that visits all nodes at the current depth before moving to the next depth
- **Depth-first Search (DFS):** A graph traversal algorithm that visits all nodes in a branch before moving to the next branch

Practice Problem I: TraversalOverflow

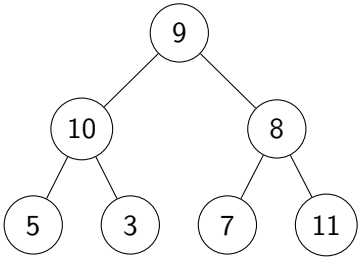
Dipal's computer was hit by a solar flare while he was working on his FSG Homework. As a result, all his trees were converted into their respective list representations. Unfortunately, he doesn't know what kind of traversals were used! He needs your help to match up the list representations with their respective trees.

1. `5, 3, 10, 7, 11, 8, 9.`
2. `5, 11, 10, 9, 8, 4, 3.`
3. `10, 3, 11, 7, 8, 9, 5.`
4. `10, 3, 11, 4, 8, 9, 5.`



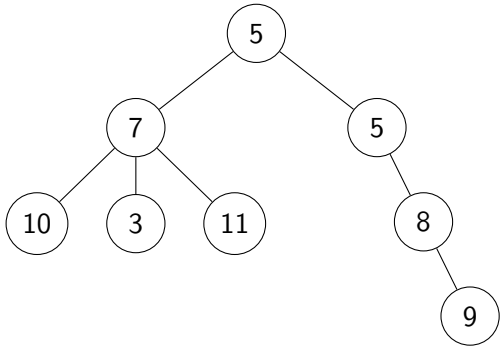
Practice Problem I: TraversalOverflow

1. `5, 3, 10, 7, 11, 8, 9.`
2. `9, 10, 8, 5, 3, 7, 11`
3. `10, 3, 11, 7, 8, 9, 5.`
4. `10, 3, 11, 4, 8, 9, 5.`



Practice Problem I: TraversalOverflow

1. `5, 3, 10, 7, 11, 8, 9.`
2. `9, 10, 8, 5, 3, 7, 11`
3. `10, 3, 11, 7, 8, 9, 5.`
4. `10, 3, 11, 4, 8, 9, 5.`



TraversalOverflow Debrief

What did you notice with the preoder and postorder traversals of each tree?

- Were they unique?
- Were they bijective?
- Was it possible to determine the tree from the traversal, or did you need to reference the tree first?

What does this tell you about the uniqueness of the pre-order and post-order traversals?

- Are they unique?
- Can you accurately reconstruct the tree from the traversal? Why or why not?
- Is there a bijection between the traversal and the tree?

Keep this in the back of your head for the next activity

Practice Problem II: TreeTheory

What kinds of questions have I been focusing on during the FSGs?
Type 1, 2, or 3?

Practice Problem II: TreeTheory

What kinds of questions have I been focusing on during the FSGs?
Type 1, 2, or 3?
I've mostly been drilling you on *Type 3* questions, since they've been the most relevant to the content. However, Trees have a lot of *Type 1 and Type 2* potential, so this next activity will focus on those.

Practice Problem II: TreeTheory

Question:
Consider an arbitrary binary tree of size n (i.e: n nodes). What is the maximum number of leaves that a binary tree of size n can have? What is the minimum number of leaves that a binary tree of size n can have?
Hint: Consider when the maximum amount of subtrees is achieved, and when the minimum amount of subtrees is achieved.

Practice Problem II: TreeTheory

Question:
Consider the `search` operation for a *Binary Search Tree*. What is the **worst case** time complexity for the `search` operation? When does it happen? Give an example of a tree that would cause this to happen.
Are BST operations *in general* always $\mathcal{O}(\log n)$? Why or why not?
Hint: Recall our discussion about the Balancing Factor and AVL Trees earlier in the slides.

Practice Problem II: TreeTheory

Question:
You are working on a new sorting algorithm known as *IbraSort*. *IbraSort* works by first inserting all the elements into a *Binary Search Tree*, and then performing an *In-Order Traversal* to retrieve the sorted elements. What is the time complexity of *IbraSort*? Is it ever possible for *IbraSort* to have a complexity of $\mathcal{O}(n \log n)$? Why or why not?