IbraFSG™ 6 - Week 10; Expression Trees and Midterm Review

Ibrahim Chehab

UTM RGASC

March 18, 2024

Table of Contents

FSG HouseKeeping

1. **Reminder that a new FSG Session has been added:**
   - **When?** *Fridays, 12:00-1:00PM*
   - **Where?** *IB210*

FSG HouseKeeping

1. **Reminder that a new FSG Session has been added:**
   - **When?** *Fridays, 12:00-1:00PM*
   - **Where?** *IB210*
2. **Assignment 2 Deadline Coming Up!** Assignment 2 is due on April 3rd! Make sure you're actively working on it!

FSG HouseKeeping

1. **Reminder that a new FSG Session has been added:**
   - **When?** *Fridays, 12:00-1:00PM*
   - **Where?** *IB210*
2. **Assignment 2 Deadline Coming Up!** Assignment 2 is due on April 3rd! Make sure you're actively working on it!
3. **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

FSG HouseKeeping

1. **Reminder that a new FSG Session has been added:**
   - **When?** *Fridays, 12:00-1:00PM*
   - **Where?** *IB210*
2. **Assignment 2 Deadline Coming Up!** Assignment 2 is due on April 3rd! Make sure you're actively working on it!
3. **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
4. **Join the UTM CS Discord Server!** https://discord.gg/utmcs

Welcome back to IbraFSGs™

- Welcome back to IbraFSGs™! Hello to new people and welcome back to tenured members.
- This week we will be quickly going over expression trees, and then going over content for your midterms
- There are several use cases for expression trees:
  - Evaluating expressions (duh...)
  - Integral/Derivative calclators (Symbolab and Wolfram Alpha are just big expression trees)
  - Rendering out a canvas/timeline in creative applications (Photoshop, Premiere, etc.)
  - Engineering simulations
    - Electrical Circuit Analysis
    - Fluid Dynamics
    - Computational Chemistry and Physics
    - Structural Analysis
  - Compiling and optimizing source code

Welcome back to IbraFSGs™

- Welcome back to IbraFSGs™! Hello to new people and welcome back to tenured members.
- This week we will be quickly going over expression trees, and then going over content for your midterms
- There are several use cases for expression trees:
  - Evaluating expressions (duh...)
  - Integral/Derivative calclators (Symbolab and Wolfram Alpha are just big expression trees)
  - Rendering out a canvas/timeline in creative applications (Photoshop, Premiere, etc.)
  - Engineering simulations
    - Electrical Circuit Analysis
    - Fluid Dynamics
    - Computational Chemistry and Physics
    - Structural Analysis
  - Compiling and optimizing source code
- **Note:** Memorize at least one of the above use cases for the midterm - Oftentimes a question is *What is a use case for expression trees?*

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 contiguous, organized manner** and helps you **find gaps in your knowledge**
  - You should **not** be copying the textbook or lecture slides verbatim; You should be **summarizing** the content in your own words while tying in examples and analogies
- UltraSheets™ help with type 1 and 2 questions
- With your midterm coming up, you should have at least one UltraSheet™ for each week of content, or at least the content you're struggling with

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™:

- **Expression**
- **Expression Tree**
- **In-Order Traversal**
  - **Note:** Not all expression trees are binary; However most expression trees in this course will be binary
- **Polymorphism**
  - **Why?** Expression Trees abuse Polymorphism to exist

Key Concepts - My Thoughts

The ultimate take-aways from Weeks 5-10 are as follows:

1. **Recursion**

Key Concepts - My Thoughts

The ultimate take-aways from Weeks 5-10 are as follows:

1. **Recursion**
2. **List Comprehensions**

Key Concepts - My Thoughts

The ultimate take-aways from Weeks 5-10 are as follows:

1. **Recursion**
2. **List Comprehensions**
3. **Trees**

Key Concepts - My Thoughts

The ultimate take-aways from Weeks 5-10 are as follows:

1. **Recursion**
2. **List Comprehensions**
3. **Trees**
4. **Efficiency and Complexity**

Key Concepts - My Thoughts (Cont'd)

The ultimate take-aways from Weeks 5-10 are as follows:

1. **Recursion**
   - Base Case
   - Recursive Case
   - My Recursion Analogy
2. **List Comprehensions**
   - Syntax
   - Use Cases
   - Chaining/Composition
3. **Trees**
   - Traversals
   - BSTs vs BTs
   - Efficiency and Complexity
4. **Efficiency and Complexity**
   - How do we measure efficiency?
   - Big-O Notation
   - Recursive Efficiency

Type 1, 2, and 3 Questions

You've often heard me refer to *Type 1, 2, and 3* questions. What do I mean by this?

Type 1, 2, and 3 Questions

You've often heard me refer to *Type 1, 2, and 3* questions. What do I mean by this?

- **Type 1:** *Theory-based questions*
  - These questions are typically pulled straight out of notes/lecture slides
  - These questions are typically multiple choice or short answer
  - Study for them using your UltraSheets™

Type 1, 2, and 3 Questions

You've often heard me refer to *Type 1, 2, and 3* questions. What do I mean by this?

- **Type 1:** *Theory-based questions*
  - These questions are typically pulled straight out of notes/lecture slides
  - These questions are typically multiple choice or short answer
  - Study for them using your UltraSheets™
- **Type 2:** *Knowledge-based questions*
  - These questions typically build off of Type 1 questions, and usually require you to have a strong understanding of the material. They require you to reflect on what you learned in lecture + preps
  - These questions are typically short answer or long answer
  - Study for them using your UltraSheets™

Type 1, 2, and 3 Questions

You've often heard me refer to *Type 1, 2, and 3* questions. What do I mean by this?

- **Type 1:** *Theory-based questions*
    - These questions are typically pulled straight out of notes/lecture slides
    - These questions are typically multiple choice or short answer
    - Study for them using your UltraSheets™
- **Type 2:** *Knowledge-based questions*
    - These questions typically build off of Type 1 questions, and usually require you to have a strong understanding of the material. They require you to reflect on what you learned in lecture + preps
    - These questions are typically short answer or long answer
    - Study for them using your UltraSheets™
- **Type 3:** *Application-based questions*
    - These questions typically require you to apply your knowledge to a new problem
    - These questions are typically long answer
    - Study for them by practicing problems

Midterm-taking Strategies

1. **Preflight Analysis:**
   - Quckly skim through the entire exam to see what you're up against. Make mental notes of the questions you know you can answer, and the questions you're unsure about.

Midterm-taking Strategies

1. **Preflight Analysis:**
   - Quckly skim through the entire exam to see what you're up against. Make mental notes of the questions you know you can answer, and the questions you're unsure about.
2. **Butterfly Method:**
   - If you get stuck on a question, go to the next question you can solve (see above). This will help you get into a rhythm and build confidence.
   - Sometimes swapping between questions rapidly in succession can help you solve the question you were stuck on.

Midterm-taking Strategies

1. **Preflight Analysis:**
   - Quckly skim through the entire exam to see what you're up against. Make mental notes of the questions you know you can answer, and the questions you're unsure about.
2. **Butterfly Method:**
   - If you get stuck on a question, go to the next question you can solve (see above). This will help you get into a rhythm and build confidence.
   - Sometimes swapping between questions rapidly in succession can help you solve the question you were stuck on.
3. **Time Management:**
   - Allocate time for each question based on the number of marks it's worth. If you're stuck on a question, move on and come back to it later.
   - If you're stuck on a question that isn't worth much, skip it and come back to it later. Chances are you're overthinking it.
   - Even if you don't end up figuring it out, if it was only worth 1-2 marks, it's not the end of the world. Prioritize the questions worth more marks.
   - A question worth more marks $\implies$ it's harder

Jamboard Link

https://tinyurl.com/ibrafsg0320

0320 for March 20th

Note: Don't fool around with the Jamboard, it's for your benefit.

Practice Problem 1.1

Which of the following statements is true? Select all that apply:

1. The in-order traversal of a binary tree is always sorted
2. The post-order traversal of a binary search tree *might* be sorted
3. The pre-order traversal of a binary tree *might* be sorted
4. The pre-order traversal of a binary search tree is always sorted
5. **All** ancestors of a node in a binary search tree are less than the node
6. **All** ancesors of a node in a binary tree *might* be less than the node

Practice Problem 1.1

Which of the following statements is true? Select all that apply:

1. The in-order traversal of a binary tree is always sorted
2. The post-order traversal of a binary search tree *might* be sorted
3. The pre-order traversal of a binary tree *might* be sorted
4. The pre-order traversal of a binary search tree is always sorted
5. **All** ancestors of a node in a binary search tree are less than the node
6. **All** ancesors of a node in a binary tree *might* be less than the node

Pay close attention to the usage of Binary Search Tree and Binary Tree in questions like these

Obviously, we know they are **NOT** interchangeable, but the question might be trying to trick you.

Practice Problem 1.2

Given the in-order traversal of an arbitrary binary *search* tree, is it possible to reconstruct the original tree? If so, how? If not, why not?

Hint: Recall our discussion on AVL Trees from last week

Hint 2: Recall your lab on BST rotations

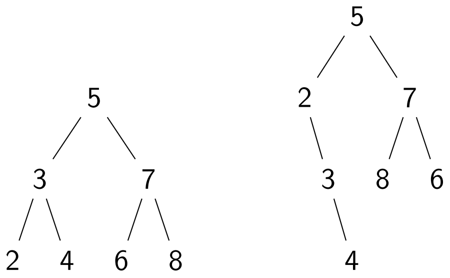
Practice Problem 1.2

Given the in-order traversal of an arbitrary binary *search* tree, is it possible to reconstruct the original tree? If so, how? If not, why not?

Hint: Recall our discussion on AVL Trees from last week

Hint 2: Recall your lab on BST rotations

Consider the two following trees:



Practice Problem 1.3

Given a sorted list of integers from 1 to n, $n \in \mathbb{N} \setminus \{0\}$, how do you construct a balanced binary search tree (i.e: height $\log(n)$) from this list? Outline the order of insertion, and explain why this works.

Practice Problem 1.3

Given a sorted list of integers from 1 to n, $n \in \mathbb{N} \setminus \{0\}$, how do you construct a balanced binary search tree (i.e: height $\log(n)$) from this list? Outline the order of insertion, and explain why this works.
Hint: Consider what is required for a BST to be balanced

Practice Problem II: Itr2Recur

Convert the following iterative function to a recursive function:

```
def ibranatchi_iterative(n: int) -> int:
  if n == 0:
    return 0
  elif n == 1:
    return 1
  elif n == 2:
    return 5

  sequence = [0, 1, 5]
  for i in range(3, n + 1):
    next_value = sequence[i - 1] * 2 * sequence[i - 2] - 5 *
   sequence[i - 3]
    sequence.append(next_value)

  return sequence[-1]

def ibranatchi_recursive(n: int) -> int:
  # TODO: Implement this recursively:
```

Practice Problem III: List Comprehensions

What is the output of the following list comprehension. If it throws an error, explain why:

```
[[x*y for x in range(4)] for y in range(7)]
```

what if we switch the `x` and `y` iterables?

Thank you for coming!

Figure 1: Image courtsey of Looney Tunes™ and Warner Bros.™

Good luck, everyone!