Trees

Updated: 7th December, 2021

Aims

- To implement a binary tree.
- To traverse a tree.
- To add file I/O and serialisation to the tree.

Before the Practical

- Read this practical sheet fully before starting.
- Ensure you have completed either Practical 3 or Practical 4.

Activities

1. UML Diagrams

Following the notes from the lecture slides as a guide, draw the UML diagrams for `DSATreeNode`, `DSABinarySearchTree` and their test harnesses.

Get feedback on the diagrams before you start coding.

2. Binary Search Tree Implementation

Following the lecture slides as a guide, implement a Binary Search Tree using a `DSATreeNode` and `DSABinarySearchTree` class. Write a test harness to test each operation thoroughly, be sure to test all cases.

Note:

- `DSATreeNode` has already been written for you, but you'll need to understand and test it.
- The code for `find()` was already implemented for you - `insert()` and `delete()` are very similar. *The methods must all use the **recursive** approaches and pseudocode from the lecture slides.*
- You may want to leave `delete()` until you finish the rest of the practical and then come back to it.

3. Implement Additional Methods

The lecture slides described the approach for doing `min()`, `max()` and `height()`. Implement each of these operations in `DSABinarySearchTree` and test them in your test harness.

Now consider how you would give a percentage score for how balanced the tree is. Implement this approach as a new method called `balance()`.

> **Note:** Approaches can include comparing left and right heights or comparing potential and actual leaf nodes.

4. Implement Traversal Methods

The lecture slides described the approach for doing `inorder()`, `preorder()` and `postorder()` traversals of a tree. Add recursive implementations of these algorithms inside `DSABinarySearchTree` to output the traversed tree.

> **Note:** You may want to export the output of each traversal method as a queue or linked list, which can then be iterated over to display the contents.

5. Interactive Menu and File I/O for DSABinarySearchTree

Setup an interactive menu system to explore building a binary tree from scratch or from file, displaying the tree using each of your traversals, saving it to a file and re-reading in those files to create new trees.
Include at least the following options:

(a) Read a .csv file

(b) Read a serialised file

(c) Display the tree - ask the user if they want inorder, preorder or postorder traversal

(d) Write a .csv file - ask the user if they want inorder, preorder or postorder traversal

(e) Write a serialised file

Consider what happens when you rebuild a tree using the output of different traversal methods.

Submission Deliverable

- Your code and UML diagrams are due 2 weeks from your current tutorial session.
  - You will demonstrate your work to your tutors during that session
  - If you have completed the practical earlier, you can demonstrate your work during the next session
- You must **submit** your code and any test data that you have been using **electronically via Blackboard** under the *Assessments* section before your demonstration.
  - Java students, please do not submit the *.class files

Marking Guide

Your submission will be marked as follows:

- [2] Your UML diagrams for all implemented classes. Include the methods (excluding basic getters and setters).
- [2] Your DSABinarySearchTree and DSATreeNode are implemented correctly - your test harness should test all functions in the binary tree.
- [2] You have implemented and tested your methods for min(), max(), height() and balance().
- [2] You have implemented and can demonstrate your methods for inorder(), preorder() and postorder().
- [2] You have implemented and can demonstrate your file I/O code.

End of Worksheet