Day 25 – Tree Variations

Chapter 07, 7.10 – Tree Variations , Review Problem(s): ***57***

It is fun teaching this course, CS 240 – Computer Science II in the same semester as CS 360 – Programming Languages. There is a lot of overlap.

Computer Scientists work to understand how a high-level language, like Java, is transformed into machine code, which is then executed by the computer.

Two models for this conversion are interpreted languages and compiled languages.

In an interpreted language, each line of a program is read, and then immediately converted and executed. You may have had some experience with a Python Notebook, which is interpreted.

In a compiled language, like C++, the program, or source code, is converted ("compiled") into an executable, which can then be run.

Java is a hybrid language, a blend of interpreted and compiled. Java source code is converted to Java byte-code, which when it is run is compiled "just-in-time."

One of the key phases of the translations is analyzing the syntax, checking if the source code follows the rules of the language.

The syntax analysis phase has two parts, the lexical analysis phase and the syntax analysis (***yes, syntax analysis is overloaded, but seriously, it isn't that confusing***).

In the lexical analysis phase, the compiler or interpreter will identify the basic building blocks, called ***tokens***, such as variables, keywords, and operators. Then, the expressions are parsed and an expression tree is attempted to be constructed.

It's this phase, constructing the parse tree, when syntax errors are reported.

When first learning the syntax analysis, mathematical expressions are studied because they are relatively simple and have a well-defined precedence.

Consider a few mathematical expressions and related parse trees. Find one (expression or tree), given the other.

Another variation on a tree is a ***trie***. It is used in many search engines, like Google. It's from re***TRIE***val. The characters in a word makeup a tree, and the leaf nodes are links to all the URLs containing those words. See Chapter 07 PPT, pg 85.

Next, create several different trees, using the normal alphabetical ordering ("lexicographic" from MA 116), for the letters: ***A, C, F, J, M, P, R, S, X, Z***

One of the trees should have the maximum height for this set of letters, and one should have the minimum height.

Our tree algorithms depend on the height of a tree, so keeping a balanced tree helps the performance of the operations on the trees.

Note that for each tree you create, the Inorder traversal will be the original list of characters, in alphabetical order.

In CS 315 – Algorithms and Analysis, you will study algorithms used to balance trees.

Chapter 07 PPT, pg 81-90.