Computer Science 2530
April 16, 2020

Happy Thursday, April 16.

Assignments 6 and 7

I have been going through the assignment 6 submissions that do not work. Disappointinly, that is close to half of them. Even more disappointingly, many of the submissions that do not work are plagiarized. For some reason, a lot of people got hold of the same faulty code, put their name on it and submitted it.

The provisions in the syllabus against plagiarism remain in effect. The current situation is not an excuse to cheat. If you put your name on someone else's work, expect to receive a negative grade.

That brings us to assignment 7. You will want to do your own work on that. Do not submit a plagiarized solution. You can recover from a bad grade on assignment 6, but it is very unlikely that you can recover from submitting a plagiarized assignment 7.

Do not write code for assignment 7 by sitting in front of a computer screen and typing. Draw pictures and work out function definitions. If you try to code without planning, you will end up with a solution that is hopelessly incorrect and beyond salvage. You will end up frustrated, and you will be tempted to plagiarize. That is bad.

If you draw pictures and work things out on paper, you will probably end up in a good place where your solution works. That is good. Do that.

Keeping binary search trees height-balanced

Pages **42A** to **42C** describe how to use rotations to keep a binary search tree height balanced. You can safely skip page **42B**. You can also safely skip the section titled **Do single rotations suffice to rebalance?** on page **42A**. Spoiler: single rotations don't always work. Sometimes you need to do a double rotation.

Concentrate on page **42C**, which gives examples of single rotations and double rotations. The red links show two steps downward from a node that is not height-balanced after the insertion, requiring a rotation. Each red link beneath a node v is toward the higher subtree of v. If the red links are both left links or are both right links, we call it a **zig-zig**, and a single rotation is called for. If the red links are one to the left and the other to the right, we call it a **zig-zag**, and a double rotation is called for.

Notice that you don't always do a rotation at the root. After doing an insertion, work your way back up the path toward the root. Check each node to see if it is height-balanced after the insertion. When you encounter one that is not height-balanced, do a rotation at that node.

You will need to be able to do an insertion with rotations on the next test. Make sure that you understand how they work.