CE204 Lab 3: Binary trees

David Richerby

Week 19, 2020–21

The lab exercises are not assessed and you are not required to complete all of them, though I recommend that you attempt them all. Feel free to work with others and talk about your answers.

Solutions will be released on Moodle, on the Friday after the labs.

1 Priority queues

Modify the priority queue class from lecture 3 so that it stores `Object`s along with the priorities. Thus, `insert()` will take two arguments – an object and an integer priority for it – and `next()` will return the object with the highest priority, as described on slide 99.

A simple way to test your code would be to call `insert (p, new Integer (p))` for several values of `p` and checking that `next()` returns the `Integer`s in increasing order.

2 Recursive algorithms on binary trees

Make a copy of the `BinaryTree` class from lecture 2.

a) Add a recursive method to `BinaryTree` that calculates the number of nodes in the tree.

b) Add a recursive method to `BinaryTree` that calculates the tree's height.

c) Add a recursive method to `BinaryTree` that returns a tree with left and right subtrees swapped.

d) Add a recursive method to `BinaryTree` that determines whether the tree is a binary search tree. Hint: the root can hold any number between `Integer.MIN_VALUE` and `Integer.MAX_VALUE`. If a node storing value v is in a subtree where all elements must have values between a and z, then what values can occur in v's left subtree? In v's right subtree?

3 Depth of random binary search trees

Implement a basic binary search tree class that includes an `insert()` method and a method to calculate the height of the tree. Use it to investigate the claim on slide 87 that a binary search tree generated by n random insertions has, on average height approximately $3\log_2 n \approx 4.311 \ln n$, where ln is the natural (base-e) logarithm.

Use `Math.Random.nextInt()` with no argument to generate random numbers in the full range that can be stored in type `int`, so there will be fairly few duplicates. (A lot of duplicates would reduce the expected height of the tree, because `insert` discards them.)

Reed's more detailed claim is that there is a constant k such that the average height tends to $4.311 \ln n - 1.953 \ln(\ln n) + k$ as n gets large. (B. Reed, *The Height of a Random Binary Search Tree*, Journal of the ACM, vol. 50, no. 3, pp.306–332, 2003.)