CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean — Fall 2017
Webpage: `http://www.cs.clemson.edu/~bcdean/` — TTh 12:30-1:45
Handout 6: Lab #4 — McAdams 119

1 Binary Search Tree Practice

The goal of this lab is to gain familiarity with binary search trees. You will code a number of primitive operations on a BST, ultimately building the code base for a randomly-balanced BST – as described in lecture, this form of balancing ensures that the BST stays in a state which is always as if it was just built from scratch by inserting its elements in random order. It's height is always $O(\log n)$ with high probability, so every major BST operation therefore runs in $O(\log n)$ time with high probability.

You will start with the following file:

`/group/course/cpsc212/f17/lab04/bst.cpp`

In this file you will find the structure defining a node in a BST, which contains an integer key, pointers to its left and right children, and a size field that should contain a count of the number of nodes in its subtree. For convenience, there is also a constructor that lets you easily allocate a new node given its integer key; for example:

```
Node *n = new Node(7);
```

There are a number of functions to fill in for this lab. All of them start with a comment describing what the function should do, and all of them have been discussed in class during the past week (in fact, we have already written code for some of them during class, so for example `insert` and `print_inorder` should be quite easy as a warm-up). A few notes on the more advanced functions:

- You should `remove` a node from a tree by replacing it by the join of its two subtrees. Don't forget to de-allocate the memory used by the node you remove.
- When you `join` two subtrees L and R together, you'll make a random choice between the root of L and the root of R for the root of the merged tree. You want to choose with probabilities proportional to the sizes of L and R, so you will choose the root of L with probability $\frac{|L|}{|L|+|R|}$, or the root of R with probability $\frac{|R|}{|L|+|R|}$ (here $|T|$ denotes the size of T). To help with making your random choice: the function `rand()` (defined in `cstdio`) returns a random integer, so the expression `rand() % N` returns a random integer in the range $0 \ldots N-1$. For example, "`if (rand() % N == 0) A; else B;`" would execute statement A with probability $1/N$ (only if the random number we choose comes out to zero), and otherwise statement B with probability $1 - 1/N$.

- The `split` function needs to return pointers to two trees, and since functions can generally only return one value, we have therefore achieved this by passing pointers to these pointers – these are the `Node **L` and `Node **R` arguments to `split`. Be careful here in keeping track of what your variables really mean – for example, `L` is a pointer to a pointer, so `*L = ...` can be used to change the value of the pointer that `L` points to.

For convenience, the `main` function contains some pre-built testing code, which can help you determine if your implementations are working correctly.

2 Grading

For this lab, you will receive 8 points for correctness and 2 points for having well-organized, readable code. Zero points will be awarded for code that does not compile, so make sure your code compiles on the lab machines before submitting!

Final submissions are due by 11:59pm on the evening of Friday, September 29. No late submissions will be accepted.