CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean — Fall 2021
Webpage: `http://www.cs.clemson.edu/~bcdean/` — MWF 9:05-9:55
Handout 8: Lab #6 — Powers 208

1 Making Grading Easier

Deciding who passes 2120 is always a challenge. To simplify this process, one possibility is to employ a so-called *Josephus permutation*: starting with n students numbered $0 \ldots n-1$ standing in a circle (student 0 following student $n-1$), we begin with student 0 and take 2120 steps forward. The student we land on is failed and removed from the circle, and we then take 2120 more steps around the circle (the first step being the student immediately following the student who just failed), again failing and removing the student we land on, and so on. The final student who remains passes the class[1].

A straightforward simulation of the mechanism above takes $\Theta(n^2)$ time. In this lab, we'll use binary search trees to achieve a faster running time of $O(n \log n)$. This will be an excellent change to practice using binary search trees to represent an arbitrary sequence (the sequence of students standing around the circle, in this case).

2 BST Updates

Starting with the BST code from lab 5, our goal with this lab is to re-tool the BST so it represents an arbitrary sequence. This means:

- The *find* function will no longer serve any useful purpose, as all access will be based on index/rank instead of value.
- We'll implement *select* at the beginning of lab, which selects the element of a specific rank (in this case, a student's index standing around the circle).
- You'll need to implement *insert* and *remove* based on rank instead of value.
- Finally, to make things run quickly, you'll want to modify *split* so it splits on a given rank instead of on a value, using this to update the faster version of *insert* that keeps the tree balanced by making sure its shape stays "as if just build randomly from scratch".

[1]The course staff have suggested to me that this mechanism might need some fine tuning before being deployed as an actual grading mechanism...

3 Finding the Student Who Passes

Now that you have an updated BST capable of representing a sequence, you'll want to read an integer n from standard input, build a BST representing the sequence $0, 1, \ldots, n-1$, and simulate the process above until only 1 student remains. Please output the ID of the remaining student.

Your program should run in $O(n \log n)$ time with high probability for full credit. In particular, this means you should only be using the "fast" version of insert that keeps the tree balanced, rather than the slower version (although the slower version might still be useful for your own testing).

4 Submission and Grading

Code should be submitted electronically on on `handin.cs.clemson.edu`. we will be turning in all of our work this semester. 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 Sunday, October 3. No late submissions will be accepted.