CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 5

For: Sept. 12, 2019

Problem 1. Ranking and Selecting

In the previous Discussion Group, we figured out a data structure for the contestants on planet Kronos. In today's DG, consider that all the contestants are stored in a Binary Search Tree (BST) and not a heap. Your employers now want two other operations to be included:

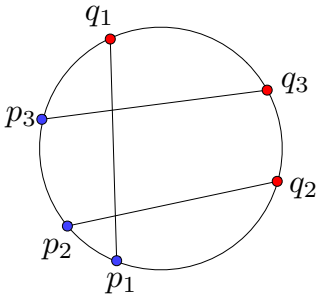
- `select(int i)` which finds the i'th smallest element.
- `rank(contestant x)` which returns the rank of element `x`, i.e., its index in the sorted list of all the elements.

Problem 1.a. Discuss how you can augment the BST to efficiently perform `select(int i)` and `rank(contestant x)` queries.
Hint: *In AVL trees, we stored the balance factor at each node to help us rebalance the tree. What information can you store at each node that can help you quickly determine a node's rank? Can the information also be useful to perform a select?*

Problem 1.b. Give pseudocode for `select(int i)` and `rank(contestant x)` and state their asymptotic time complexities.

Problem 2. Circle Intersections (*Optional*)

Suppose you are given two sets of 2-dimensional points $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$. Connect each point p_i to the corresponding q_i. What is the most efficient algorithm you can think of for determining how many pairs of these line segments intersect? In the example below, there are two intersections; the line formed by (p_1, q_1) intersects with (p_2, q_2) and (p_3, q_3), but the lines formed by (p_2, q_2) and (p_3, q_3) do not intersect with one another.
Hint: *Can you find a $O(n \log n)$ algorithm? Perhaps you can re-use the solution in Problem 1?*



Problem 3. Tree Insertion (*Optional*)

Next, we will attempt the *Tree Insertion* problem on Kattis:

`https://open.kattis.com/problems/insert`

Note: For this problem, the numbers involved can get very large and you'll need `BigInteger`: `https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html`