CMSC 420:Fall 2019 Dave Mount

Homework 2: Search Trees

Handed out Sun, Oct 6. Deadline: **6pm on Sun, Oct 13**. This is a **hard deadline**, since solutions will be posted at 6:30pm the same day.

Problem 1. Consider the 2-3 tree shown in Fig. 1.

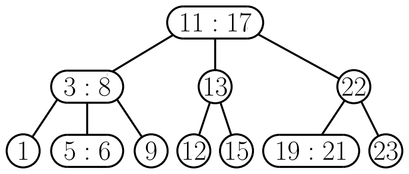

Figure 1: Problem 1.

(a) Show the AA tree corresponding to this tree. (Because Gradescope is not very good with colors, indicate each red node by having a dashed line coming in from its parent.)

(b) Show the 2-3 tree that results by inserting the key 7. (You need only show the final result for full credit, but intermediate results can be given to help with partial credit.)

Problem 2. In our simple binary-search tree implementations, we assumed that each node stores just a key-value pair (`p.key` and `p.value`) and pointers to the node's left and right children (`p.left` and `p.right`). In practice, it is often useful to store additional information including the following:

- `p.parent`: p's parent, or `null` if p is the root
- `p.min`: The smallest key in the subtree rooted at p
- `p.max`: The largest key in the subtree rooted at p
- `p.size`: The total number of nodes (including p) in p's subtree

Modify the *right rotation* pseudo-code (Lecture 5, bottom of page 3) so that (in addition to the rotation) all of the above associated values are updated. (**Note**: Remember that the return value is significant and should remain q.)

Problem 3. Define a new treap operation, `expose(Key x)`. It find the key `x` in the tree (throwing an exception if not found), sets its priority to $-\infty$ (or more practically `Integer.MIN_VALUE`), and then restores the treap's priority order through rotations.

Present pseudo-code for this operation. If you like, you may modify the template code below (which finds the key, but does not rebalance the structure). You may also assume that the code for performing left and right rotations (Lecture 5, bottom of page 3) is given to you.

```
TreapNode expose(Key x, TreapNode p) {
    if (p == null) {                        // error - key not in tree
        throw Exception("Key not found");
    }
    else if (x < p.key) {                   // x is smaller - search left
        p.left = expose(x, p.left);
    }
    else if (x > p.key) {                   // x is larger - search right
        p.right = expose(x, p.right);
    }
    else {                                  // found it
        p.priority = Integer.MIN_VALUE;     // set priority to -infinity
    }
    return p;
}
```

General note regarding coding in homeworks: Remember, when asked to present an algorithm or data structure, do **not** give complete Java code.

Submission Instructions: Because we will use GradeScope for grading, please write your solutions on the solution template, which can be downloaded from the class Handouts page. (If there is insufficient space to write your answer, please note this within the space provided.) We will deduct points for submissions that are not in proper format.

Please submit your assignment as a pdf file through Gradescope. Here is a link to an explanatory video on How to Submit Your Assignment. Please attempt a test submission well prior to the due date, even if it is empty, just to verify that everything is working properly.

You can either type your answers directly onto the template and convert to pdf document, or you can write your answer by hand and submit a scanned version. If you do the latter, please *do not* just submit the raw scanned image. Instead, use an image-enhancing app like CamScanner or Genius Scan. Submit your homework as a *single pdf document*, not multiple image files. Poorly formatted or illegible homeworks will be returned to you, and this may result in loss of points.