York University | EECS 4101/5101 | November 4, 2022

Homework Assignment #6
Due: November 11, 2022 at 11:59 p.m.

1. Suppose you have a set S of n elements (k, p), each of which has a key k and a priority p. The priorities of elements in S are distinct. The keys of elements in S are distinct. The keys are drawn from an ordered domain D.

We are interested in building a binary tree containing the n elements satisfying the following properties.

- The *keys* satisfy the binary search tree property (i.e., for every node x the keys in the left subtree of x are less than x's key and the keys in the right subtree of x are greater than x's key).
- The *priorities* satisfy the heap-ordering property (i.e., for every node x other than the root, x's priority is less than x's parent's priority).

[4] **(a)** Draw a tree containing the following key-priority pairs: $(1, 7), (3, 4), (5, 2), (7, 10), (10, 5), (12, 6), (14, 1)$.

[3] **(b)** Claim: For any S, there is *exactly one* tree having the two properties described above. Explain why this claim is true.

[6] **(c)** Suppose you have a tree T that satisfies the properties described above. Consider the following algorithm to insert a new pair (k, p) into the tree.

```
1  INSERT(T, k, p)
2      create a new node z and set z.key ← k and z.priority ← p
3      TREE-INSERT(T, z)
4      while (z.parent ≠ nil and z.priority > z.parent.priority)
5          if z is the left child of its parent then RIGHT-ROTATE(T, z.parent)
6          else LEFT-ROTATE(T, z.parent)
7          end if
8      end while
9  end INSERT
```

The TREE-INSERT on line 3 is the standard BST insertion algorithm to insert the node z into T (see page 321 of the textbook). The calls to the RIGHT-ROTATE and LEFT-ROTATE routines (which are described on page 336 of the textbook) perform rotations that move z up to its parent's position.

(i) Prove that at the beginning of each iteration of the loop, the *only* possible violation of the tree properties is that z might have a greater priority than its parent.

(ii) Explain why the loop terminates and show that when it terminates, the tree satisfies the required tree properties.

[3] **(d)** Suppose that you insert distinct keys $k_1, k_2, \ldots, k_n$ in that order, one-by-one into an initially empty tree. For each key, you choose a random priority. Assume the random choices are independent and uniform from a large enough range that the probability of choosing the same priority for two keys is negligible. (In other words, pretend the probability of choosing equal priorities is 0.) Give a good upper bound on the expected height of the resulting tree.