

# CSC148 Exercise 6: Binary Search Trees

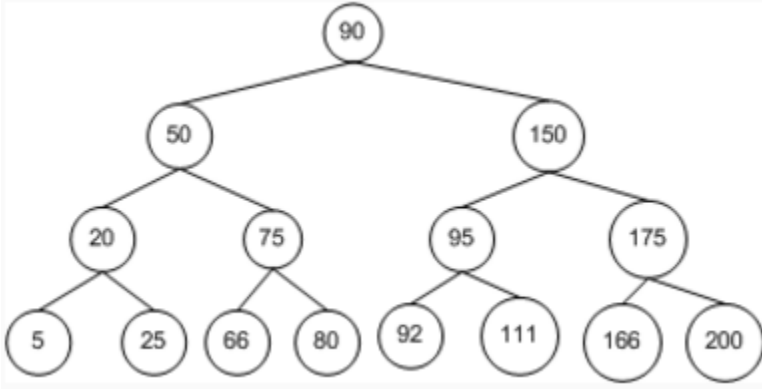
(Due: July 5, 10:00 pm on Markus)

## Task 1: BST methods

In this task, you will write three BST methods that operate on binary search tree (although the latter two do not use the BST property).

The first method is `num_less_than`, which takes an item, and returns the number of items in the BST that are less than the given item. You should think carefully about how to use the BST property to avoid making unnecessary recursive calls.

To prepare for the other two methods, first consider the following binary search tree:



A binary search tree.

We say that the *depth* of an item is equal to the distance between it and the root, counting items. So the root of a BST always has **depth 1**, and in the above tree, the items at depth 3 are 20, 75, 95, and 175. Note that the maximum possible depth of a node is equal to the height of the tree.

Your first task is to implement the BST method `items_at_depth`, which takes a positive integer `d`, and returns a list of all items at depth `d` in the BST (this list might be empty). The returned list should be sorted in non-decreasing order, but note that you should not need to explicitly sort the list yourself (e.g., by calling the built-in `sort` or `sorted`). Use the BST property here!

Then, implement the BST method `levels`, which returns a list of tuples of the form `(d, items)`, where `d` is a positive integer and `items` is a sorted list of items in the tree at depth `d`. The list should be sorted in ascending order of `d` values, and should have exactly `h` elements, where `h` is the height of the tree.

For example, calling `levels` on the above BST should produce the list

```
[(1, [90]), (2, [50, 150]), (3, [20, 75, 95, 175]), (4, [5, 25, 66, 80, 92, 111, 166, 200])]
```

You will find it easier to use `items_at_depth` as a helper method for `levels`; this is a valid approach, but is a bit inefficient. We recommend starting with this approach so that you have a working solution, and then doing some more careful thinking to make `levels` itself recursive, without using `items_at_depth` as a helper.

For all three methods, you are free to define your other helpers.