CSC148: Binary Search Trees

Daniel Zingaro
University of Toronto

Question 1

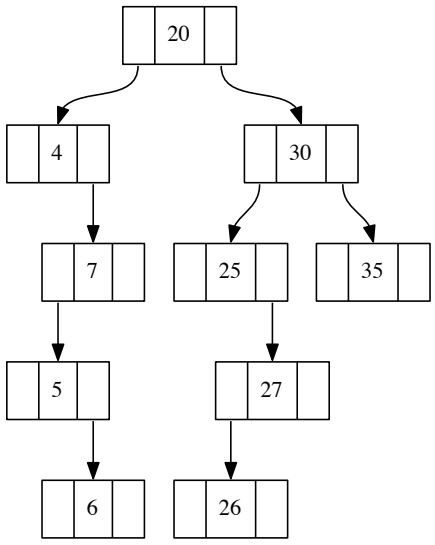
(a)
Consider each of the following orders of insertion into an empty BST.
Which one results in the tree where searches are fastest?

```
A. 1, 2, 3, 4, 5, 6, 7
B. 7, 6, 5, 4, 3, 2, 1
C. 3, 2, 1, 7, 6, 5, 4
```

(b)
What is the **maximum** height of a BST that contains 32 elements?
What is the **minimum** height of a BST that contains 32 elements?

Question 2

Consider this BST.



Show the BST after deleting node 7. Show the BST after deleting node 20.

Question 3

The algorithm for deleting a node with two children that we discussed during lecture involved replacing the unwanted value with **the maximum value from the left subtree**.

(a) What is the algorithm to find the maximum value of a BST?

(b) Your friend proposes an alternative delete algorithm as follows: replace the unwanted value with the maximum value from the **right** subtree.
Does this still work? If yes, explain why. If not, explain why not and fix the algorithm.

Question 4

Here is a Binary Tree Node class for this question. The empty tree will be represented as `None`.

```
class BTNode:
  """Binary Tree node."""

  def __init__(self: 'BTNode', data: object,
               left: 'BTNode'=None, right: 'BTNode'=None) -> None:
    """Create BT node with data and children left and right."""
    self.data, self.left, self.right = data, left, right
```

Complete the following function as specified by the docstring.

```
def is_bst(t: BTNode) -> bool:
  """Return True iff binary tree rooted at t has the BST property."""
```