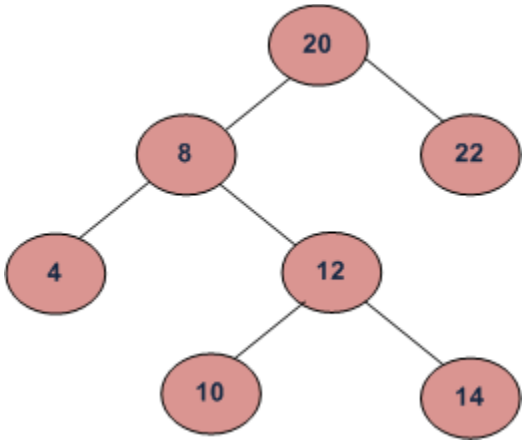
Alexander Lizzo
5/6/2022
CMPT 435 Algorithm Analysis and Design

Assignment 10

Section 1: Pen-and-paper Exercises

1. Given a BST and a positive integer k, find the k_th smallest element in the BST.

   For example, in the following BST, if k = 3, then output should be 10, and if k = 5, then output should be 14.



Assume the tree is balanced, and the tree height is O(log n). Design an algorithm to solve this problem.
(i) describe the idea behind your algorithm in English;
(ii) provide pseudocode;
(iii) analyze its running time.

(i) describe the idea behind your algorithm in English;

The Binary Search Trees (BST) preserves an accessible structure when inserting new values. Values that are less than the root are places on the left sub-node and values greater than or equal to the root are placed on the right sub-node. Since the binary search tree preserves the order in which elements are stored, it is possible to visit each node in ascending order using an In-order traversal. An In-order traversal can be used to sort the values in a balanced BST into a two-dimensional array. Making it an easy task to find the 'k_th' value or the position value of 'k.' The In-order traversal works by recursively visiting the left subtree; when no further left subtree exists, then visit the root; followed by visiting the right subtree. To find the kth position, simply call this recursive function 'k' times.

I.e.

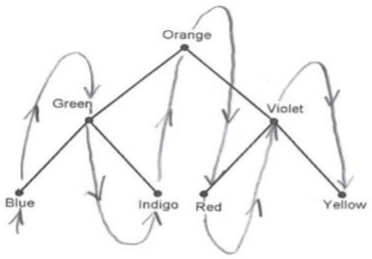
Inorder - visits the colors in order. order: (Blue, Green, Indigo, orange, red, violet, yellow)

Inorder Traversal:

Preconditions: $\mathcal{T}$ is a binary tree.

Postconditions: Every node of $\mathcal{T}$ was visited exactly once.

```
function Inorder(T ∈ {binary trees})
   if T ≠ ∅ then
      Inorder(T's left subtree)
      visit the root of T
      Intorder(T's right subtree)
   return
```



(ii) provide pseudocode;

Class Node:

```
        int data;
        Node left, right;
Node(int data, Node l, Node r)  //constructor
      left = l; right = r;
      this.data = data;

Node(int data)     //over write constructor
      this(data, null, null);

End of Class Node


 static int count = 0;
  public static Node ReturnKthSmallestElement(int k, Node p)
     if (p == null)
        return null;
     Node left = ReturnKthSmallestElement(k, p.left);
      if (left != null)
        return left;

     count++;
     if (count == k)
        return p;

     return ReturnKthSmallestElement(k, p.right);
```

(iii) analyze its running time.

The time complexity to find the kth smallest element in the BST is O(h), where h is the size of the balanced BST. The height is proportional to the number of levels in the BST. O(h) is approximately equal to O(log n), where n is the number of nodes in the tree.