Algorithms & Data Structures, Class № 9

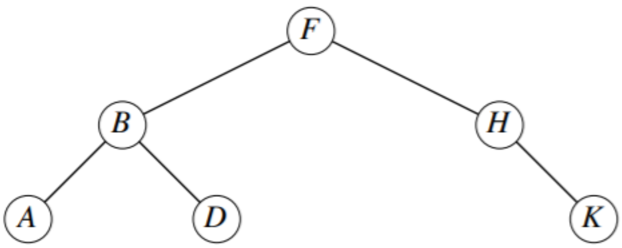
Binary Search Trees

Binary search trees are an important data structure for dynamic sets.

- Accomplish many dynamic-set operations in $O(h)$ time, where h = height of tree.
- $root[T]$ points to the root of tree T.
- Each node contains the fields
  - key (and possibly other satellite data).
  - $left$: points to left child.
  - $right$: points to right child.
  - p: points to parent. $p[root[T]] = NIL$.
- Stored keys must satisfy the binary-search-tree property.
  - If y is in left subtree of x, then $key[y] \leq key[x]$.
  - If y is in right subtree of x, then $key[y] \geq key[x]$.

Problem 1

Write down the nodes of the tree in:



- pre-order:
- in-order:
- post-order:

Write the pseudocode of the traversal procedures (whose argument is node).

Problem 2 (optional)

Implement the in-order traversal on Hackerrank:
`https://www.hackerrank.com/challenges/tree-inorder-traversal/problem`.

Problem 3

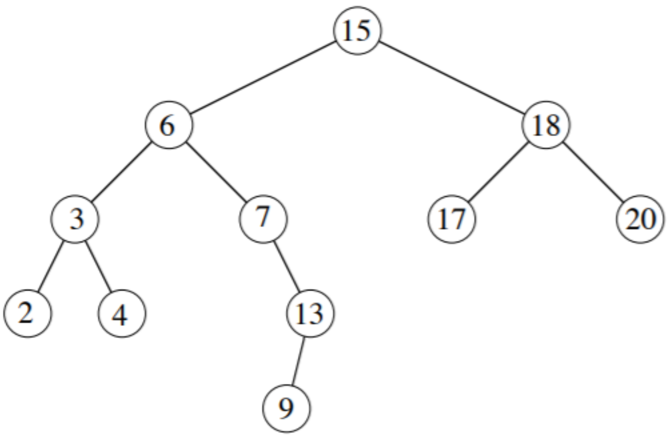
Write pseudocode for:

- searching for a value x
- finding minimum
- finding maximum

in a Binary Search Tree. What is the asymptotic complexity of these operations ($O(...)$)?

Problem 4

In the given tree:



1. Find the successor of the node with key value 15
2. Find the successor of the node with key value 6.
3. Find the successor of the node with key value 4.
4. Find the predecessor of the node with key value 6.

Write a pseudocode for a `TREE-SUCCESSOR(x)` function.

Problem 5

Show that if a node in a binary search tree has two children, then its successor has no left child and its predecessor has no right child.

Problem 6

Implement the solution for the challenge on Hackerrank:
`https://www.hackerrank.com/challenges/tree-height-of-a-binary-tree/problem`.

Problem 7

Implement the solution for the challenge on Hackerrank:
`https://www.hackerrank.com/challenges/binary-search-tree-lowest-common-ancestor/problem`.

Problem 8

Write a pseudocode for `TREE-INSERT(T,z)` procedure, where z is a new node with some value v.

Hints

To insert value v into the binary search tree, the procedure is given node z, with $key[z] = v$, $left[z] = NIL$, and $right[z] = NIL$.

1. Beginning at root of the tree, trace a downward path, maintaining two pointers.
   (a) Pointer x: traces the downward path.
   (b) Pointer y: "trailing pointer" to keep track of parent of x.
2. Traverse the tree downward by comparing the value of node at x with v, and move to the left or right child accordingly.
3. When x is NIL, it is at the correct position for node z.
4. Compare z's value with y's value, and insert z at either y's left or right, appropriately.

Problem 9

We can sort a given set of n numbers by first building a binary search tree containing these numbers (using TREE-INSERT repeatedly to insert the numbers one by one) and then printing the numbers by an in-order tree walk. What are the worst-case and best-case running times for this sorting algorithm?

Problem 10 (Homework)

Write a pseudocode for `TREE-DELETE(T,z)` procedure, where z is a node to delete. Consider 3 cases:

1. z has no children.
    - Delete z by making the parent of z point to NIL, instead of to z.
2. z has one child.
    - Delete z by making the parent of z point to z's child, instead of to z.
3. z has two children.
    - z's successor y has either no children or one child. (y is the minimum node –with no left child– in z's right subtree.)
    - Delete y from the tree (via Case 1 or 2).
    - Replace z's key and satellite data with y's