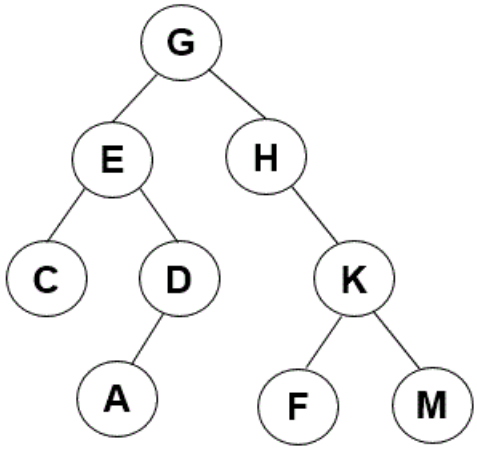
Lab 04: Trees

Task 1: Binary-tree basics

Consider the following binary tree in which the keys are letters, ordered alphabetically. Answer the following questions.



1. Is this a binary-search tree? Why or why not?
2. What is the height of the tree?
3. What is the depth of node E? Of node A?
4. If a *preorder* traversal were used to print the keys, what would the output be?
5. What would be the output of an *inorder* traversal?
6. What would be the output of a *reverse postorder* traversal?
7. What would be the output of a *level-order* traversal?

Task 2: Algorithms for binary search trees

1. Insert the following sequence of keys into an initially empty binary search tree:

```
15, 23, 20, 10, 13, 6, 18, 35, 9, 24
```

2. What does the tree look like after each of the following deletions, which are carried out in sequence?
    - delete 6
    - delete 15
    - delete 20

Task 3: Balanced Trees: 2-3 Trees

Recall that a binary tree is *balanced* if, for each of its nodes, the node's subtrees have the same height or have heights that differ by 1; another way to think of this is that the depth of all leaves can differ by at most 1. This is important because a balanced tree has a height of `O(log n)`. Therefore, for a balanced binary search tree, the worst case for *search*, *insert*, and *delete* is O(h) = `O(log n)`. Giving us the the *best* worst-case time complexity!

Is the search tree displayed in Task 1 a balanced tree? How about the resulting search tree from Task 2, question 1? You should be able to answer why or why not.

1. Create a 2-3 tree from the following sequence

```
45, 1, 25, 12, 26, 44, 4, 50, 43, 21
```

2. Draw a new tree for each of the following operations
    - insert 5

- insert 42

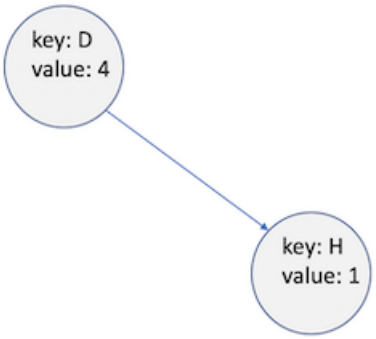
3. How many comparisons between keys are performed if I'm searching for
   - 44?
   - 18?

Task 4: Index Trees and Inverted Index Trees

1. Suppose you have an empty Index Tree (or Dictionary) where each key is a letter which has a single integer as its value. Suppose you want to insert the following key-value pairs into the tree:

   ```
   [  (D,4), (H,1), (T,3), (B,2), (F,8), (A,5) ]
   ```

   Here is the result after the first two insertions; complete the diagram:



2. Now draw the inverted index tree corresponding to this index tree; it is created by exchanging the values and keys, as if creating an index tree from [(4,D), (1,H), ...], where the keys and values have been exchanged.

What to Hand In

As proof that you did the exercises in this lab, please hand in a file Lab04Question.txt (or .pdf, or Word) containing your **answers to Task 4** only. Any reasonable attempt to show the structure of the tree is fine. You should write up the other answers as well, but you do not need to hand them in.
Last modified on Aug 4 2020.