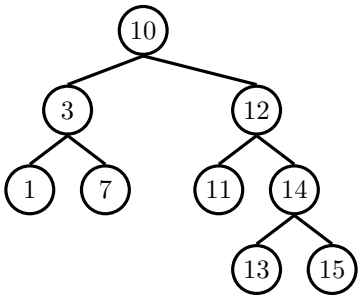
CS 61B Fall 2020

Traversals, Tries, Heaps

Discussion 9: October 19, 2020

1 Tree Traversals



Write the pre-order, in-order, post-order, and level-order traversals of the above binary search tree.

Pre-order: 10 3 1 7 12 11 14 13 15

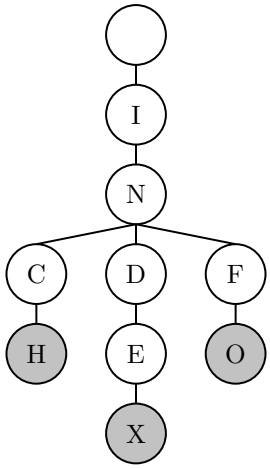
In-order: 1 3 7 10 11 12 13 14 15

Post-order: 1 7 3 11 13 15 14 12 10

Level-order (BFS): 10 3 12 1 7 11 14 13 15

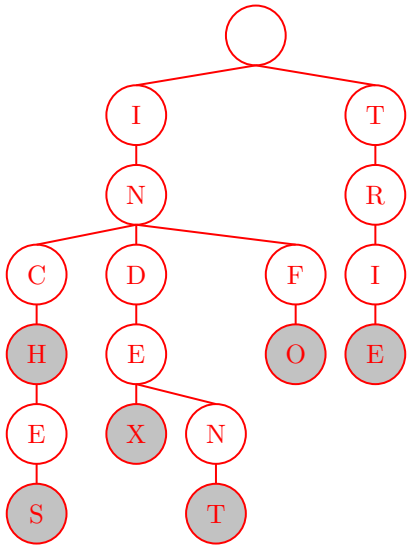
2 Tries

What strings are stored in the trie below? Now insert the strings *indent, inches, and trie* into the trie. *Extra:* How could you modify a trie so that you can efficiently determine the number of words with a specific prefix in the trie?



The strings originally contained in the trie are *inch, index, and info.*

The trie after inserting *indent, inches, and trie.*

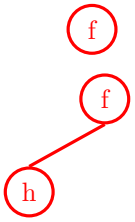


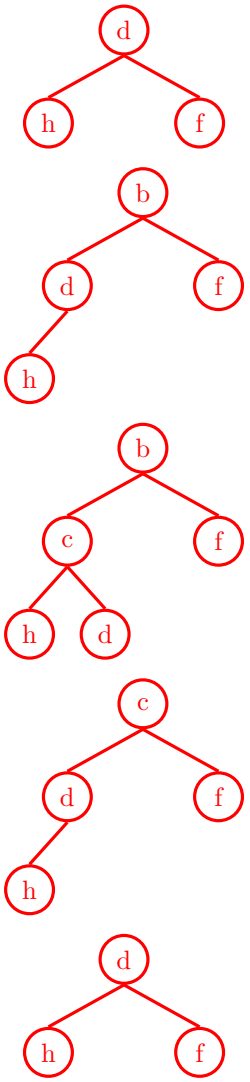
Extra:
We can add a `numWordsBelow` variable to each of the nodes in our trie. When we insert we will increment this variable for all nodes on the path to insertion. When we delete, we decrement this variable for all nodes on the path to the word. In order to determine the number of words that start with a specific prefix, we can traverse the trie following the letters in the prefix. Once we reach the end of the prefix, we return `numWordsBelow` of the last character in the prefix, or 0 if the entrie prefix is not contained in the tree. If the length of the prefix is k then this code will run in $\Theta(k)$ in the worst case. If we assume the lengths of the strings are constant, then this runtime of $\Theta(k)$ will actually be $\Theta(1)$ as we drop the constant coefficients.

3 Heaps of Fun

(a) Assume that we have a binary min-heap (smallest value on top) data structure called `Heap` that stores integers, and has properly implemented the `insert` and `removeMin` methods. Draw the heap and its corresponding array representation after each of the operations below:

```
1 MinHeap<Character> h = new MinHeap<>();
2 h.insert('f');
3 h.insert('h');
4 h.insert('d');
5 h.insert('b');
6 h.insert('c');
7 h.removeMin();
8 h.removeMin();
```





(b) Your friendly TA Anjali challenges you to quickly implement an integer max-heap data structure. However, you already have written a min-heap and you don't feel like writing a whole second data structure. Can you use your min-heap to mimic the behavior of a max-heap?

Hint: Although you cannot alter them, you can still use methods from `MinHeap`.

Yes. For every insert operation, negate the number and add it to the min-heap.

For a `removeMax` operation call `removeMin` on the min-heap and negate the number returned. Any number negated twice is itself (with one exception in Java, -2^{-31}), and since we store the negation of numbers, the order is now reversed (what used to be the max is now the min).