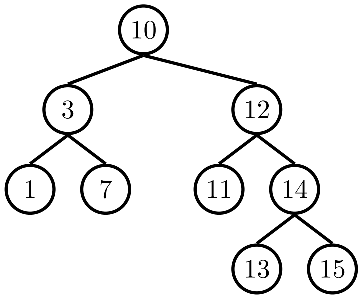
CS 61B Spring 2021

Traversals, Tries, Heaps

Discussion 9: March 15, 2020

1 Tree-versals



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

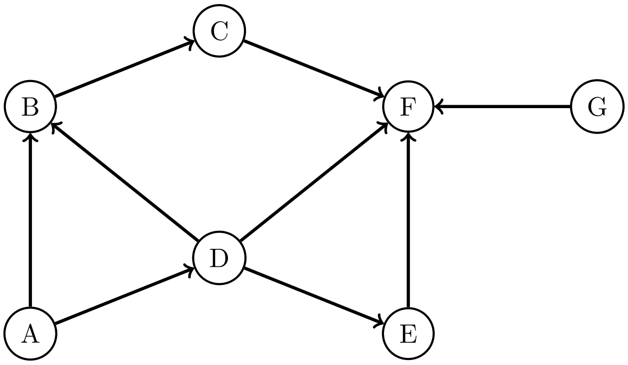
Pre-order: 10 3 1 7 12 11 14 13 15

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 Graph Representations



(a) Write the graph above as an adjacency matrix, then as an adjacency list. What would be different if the graph were undirected instead?

$$\begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ \hline A & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ D & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ F & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ G & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

A {B, D}
B {C}
C {F}
D {B, E, F}
E {F}
F { }
G {F}

$$\begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ \hline A & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ B & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ C & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ D & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ E & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ F & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ G & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

A {B, D}
B {A, C, D}
C {B, F}
D {A, B, E, F}
E {D, F}
F {C, D, E, G}
G {F}

(b) Write the order in which DFS pre-order graph traversal would visit nodes in the directed graph above, starting from vertex *A*. Break ties alphabetically. *Extra: Do the same for DFS post-order and BFS*

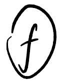
A B C F D E

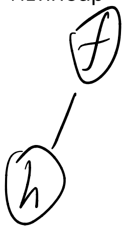
NO

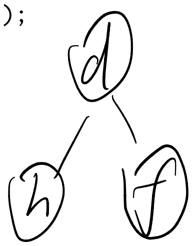
3 Heaps of Fun

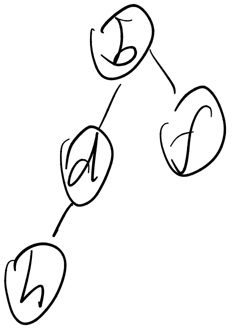
(a) Assume that we have a binary min-heap (smallest value on top) data structure called `MinHeap` that has properly implemented `insert` and `removeMin` methods. Draw the heap and its corresponding array representation after each of the operations below:

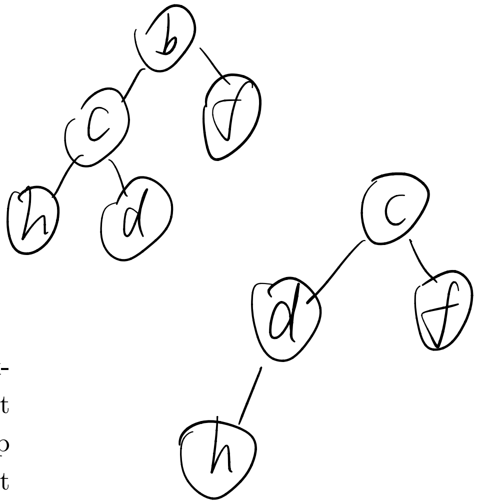
```
Heap<Character> h = new MinHeap<>();
h.insert('f');
h.insert('h');
h.insert('d');
h.insert('b');
h.insert('c');
h.removeMin();
h.removeMin();
```

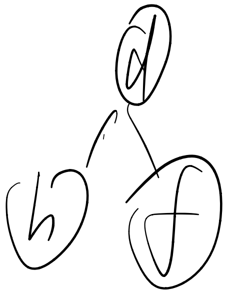












(b) Your friendly TA Tony challenges you to quickly implement an integer max-heap data structure. However, you already have your `MinHeap` and you don't feel like writing a whole second data structure. Can you use your min-heap to mimic the behavior of a max-heap? Specifically, we want to be able to get the largest item in the heap in constant time, and add things to the heap in $\Theta(\log n)$ time, as a normal max heap should.

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)