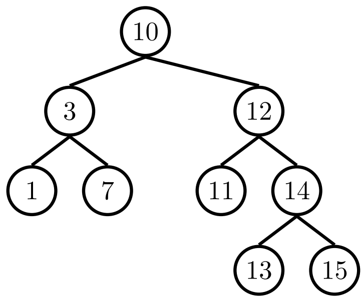
CS 61B Traversals, Tries, Heaps

Spring 2021 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.

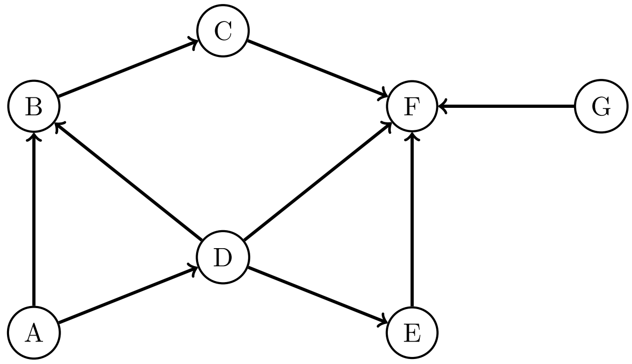
Pre-order:

In-order:

Post-order:

Level-order (BFS):

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?

(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*

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`.