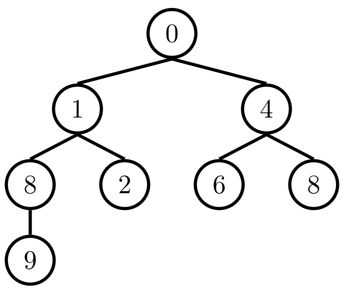
CS 61BL — Summer 2021

Heaps and Graphs

Recurring Section 10: Tuesday August 2, 2021

1 Heaps of Fun

(a) Draw the Min Heap that results if we delete the smallest item from the heap.



(b) Draw the Min Heap that results if we insert the elements 6, 5, 4, 3, 2 into an empty heap.

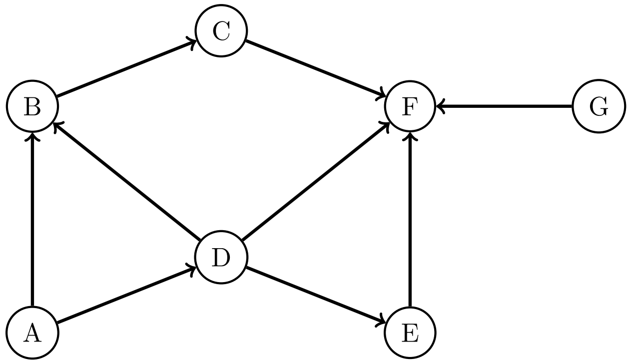
(c) Assume that we have a binary min-heap (smallest value on top) data structure called `MinHeap` that has properly implemented the `insert` and `removeMin` methods. Draw the heap and its corresponding array representation after each of the operations below:

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

(d) Your friendly TA Sadia 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`.

2 Graphs



(a) Write the directed 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 graph above, starting from vertex A. Break ties alphabetically. Do the same for DFS post-order and BFS.

3 Graph Conceptuals

Answer the following questions as either **True** or **False** and provide a brief explanation:

1. If a graph with n vertices has $n-1$ edges, it **must** be a tree.

2. Every edge is looked at exactly twice in **every** iteration of DFS on a connected, undirected graph.

3. In BFS, let $d(v)$ be the minimum number of edges between a vertex v and the start vertex. For any two vertices u, v in the fringe, $|d(u) - d(v)|$ is **always less than** 2.

4. Given a fully connected, directed graph (a directed edge exists between every pair of vertices), a topological sort can never exist.