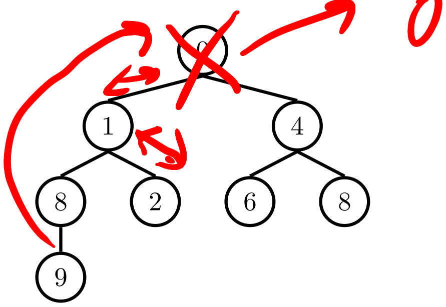
CS 61BL
Summer 2021

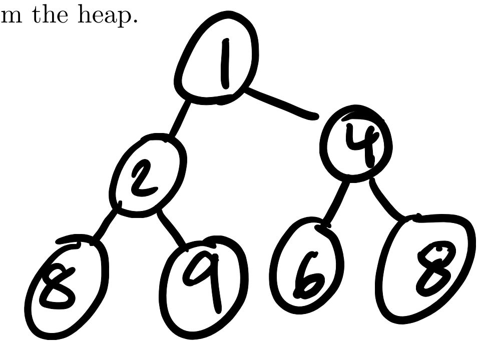
Heaps and Graphs

Recurring Section 10: Tuesday August 2, 2021

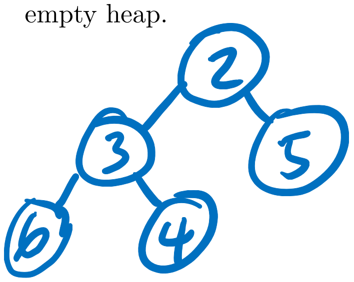
1 Heaps of Fun

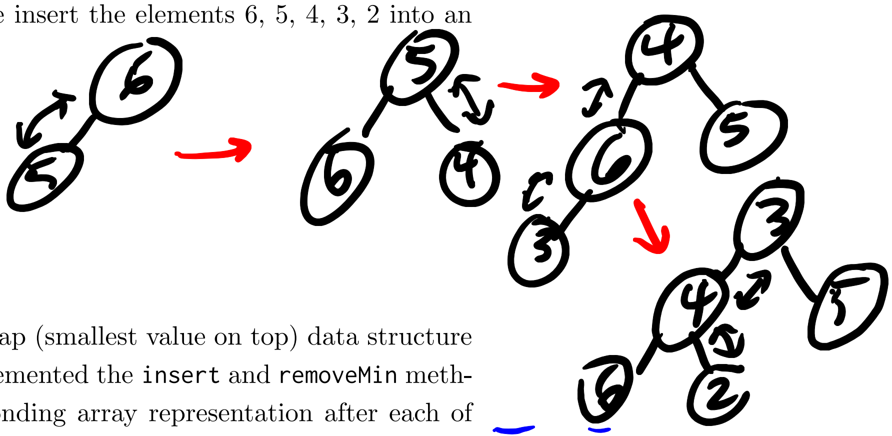
(a) Draw the Min Heap that results if we delete the smallest item from the heap.





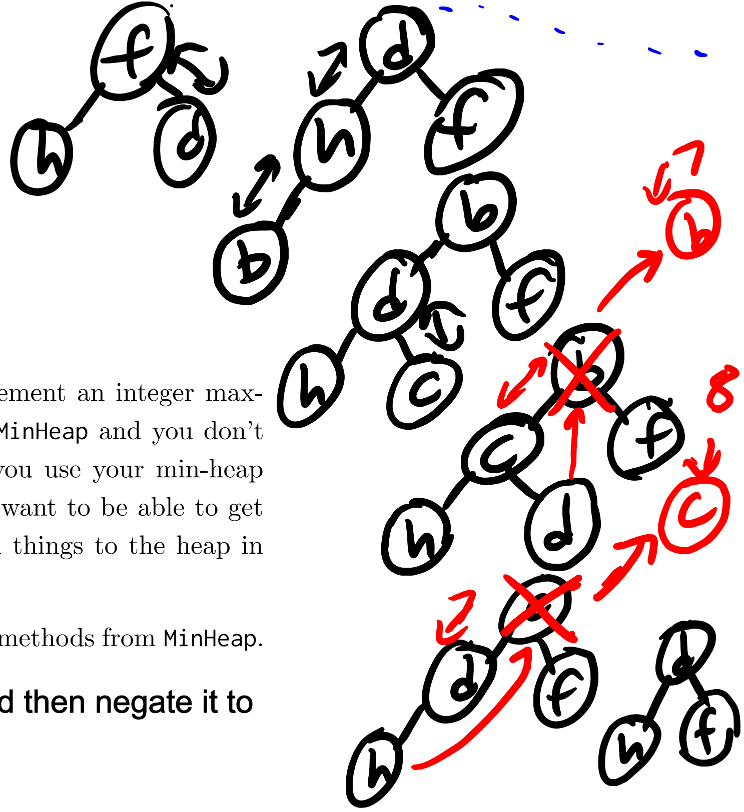
(b) Draw the Min Heap that results if we insert the elements 6, 5, 4, 3, 2 into an empty heap.





(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:

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

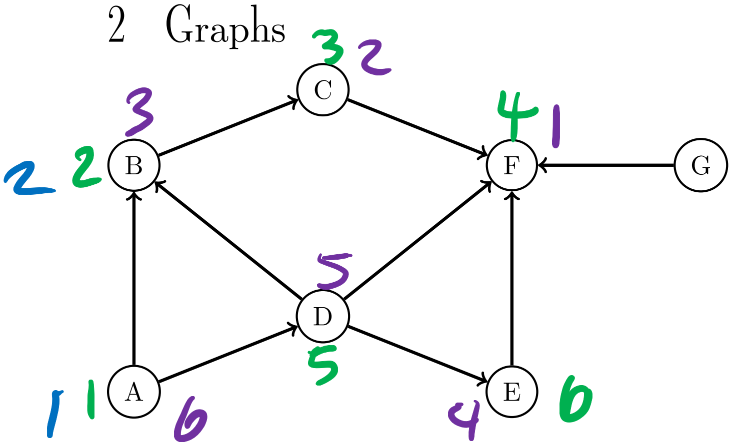


(d) Your friendly TA Sadia challenges you to quickly implement an integer max-heap data structure. However, you already have your `MinHeap` and you don't feel like writing a whole second data structure. Can you use your min-heap to mimic the behavior of a max-heap? Specifically, we want to be able to 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`.

When inserting, negate it. For removing it, remove it and then negate it to counteract the insert.

2 Graphs



Matrix:

	A	B	C	D	E	F	G - end node
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

Adjacency List:
A: {B, D}
B: {C}
C: {F}
D: {B, E, F}
E: {F}
F: {}
G: {F}

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

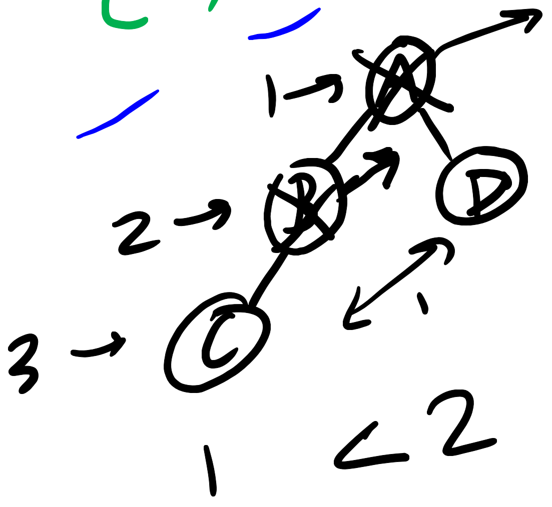
(b) Write the order in which DFS pre-order graph traversal would visit nodes in the ~~un~~directed graph above, starting from vertex *A*. Break ties alphabetically. Do the same for DFS post-order and BFS.

🟩 DFS pre-order: ABCFDE (G)
🟪 DFS post-order: FCBEDA (G)
🟦 BFS: ABDCEFG

DFS - FIFO
BFS - LIFO

visited
[~~A~~]
visited
[~~B~~, D] stack
[C, D]

visited
Queue [~~A~~]
[~~B~~, D]
[D, C]



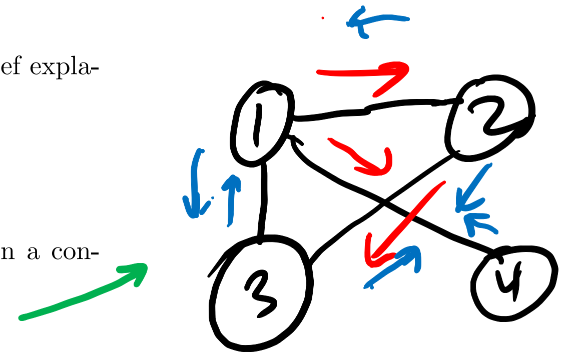
Things are added to the queue when stuff is visited; not possible to add to a level 3 without getting to level 2 first

↑
for 3.3

3 Graph Conceptuals

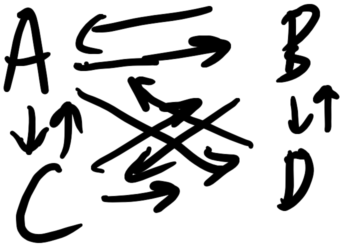
Answer the following questions as either **True** or **False** and provide a brief explanation:

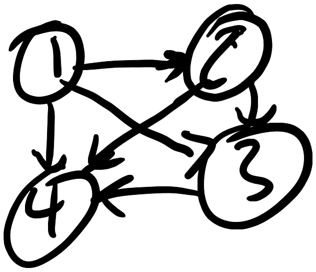
1. If a graph with n vertices has $n-1$ edges, it **must** be a tree.
   False - needs to be connected, could have a cycle
2. Every edge is looked at exactly twice in **every** iteration of DFS on a connected, undirected graph.
   False - both vertices the edge connects will look at it

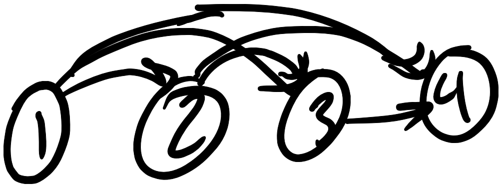


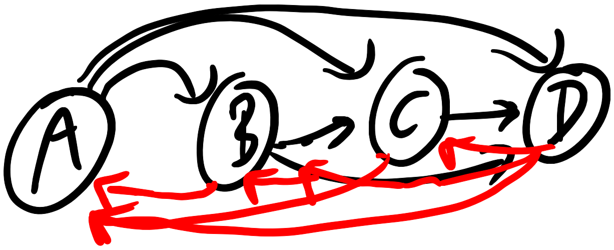
3. In BFS, let $d(v)$ be the minimum number of edges between a vertex v and the start vertex. For any two vertices u, v in the fringe, $|d(u) - d(v)|$ is **always less than** 2.
   True - see previous page
4. Given a fully connected, directed graph (a directed edge exists between every pair of vertices), a topological sort can never exist.

False - see counterexample









Connect the smaller valued node to the bigger valued node, for all nodes, ensuring all of them are connected and it is directed. Resulting topological sort is the nodes in the order from smallest to biggest valued.