Week 8 Exercises Submit

Sandra Batista

1.1-1.2

Exercise1: Using a Priority Queue for Sorting

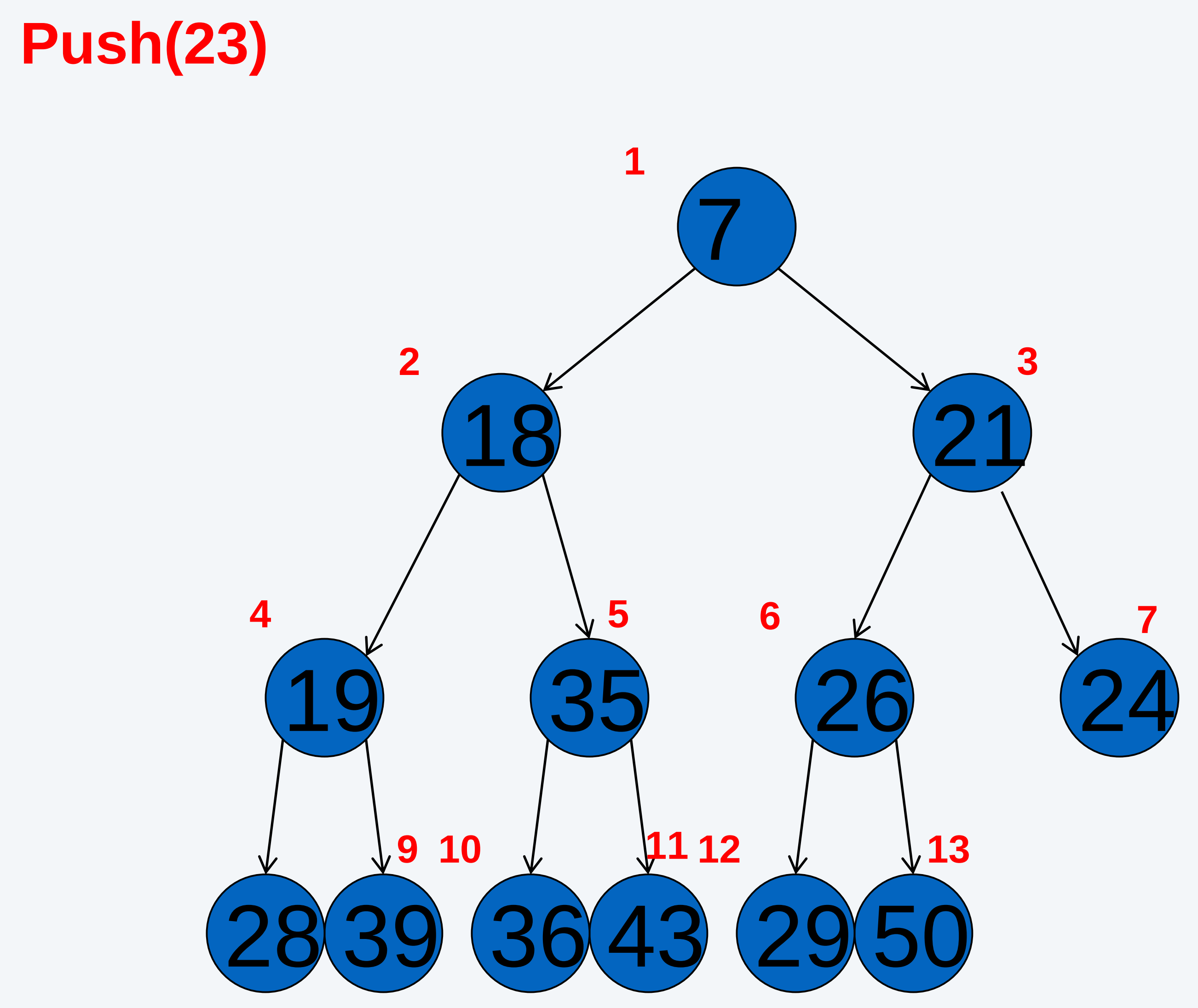
Starter code:

https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/heapsort.cpp

1. Instantiate a priority queue of integers that uses a min heap
2. Insert 10 random integers into the min heap
3. Then use the priority queue to print the integers in sorted order
4. Given a min heap with n integers, what is the runtime to print in sorted order?

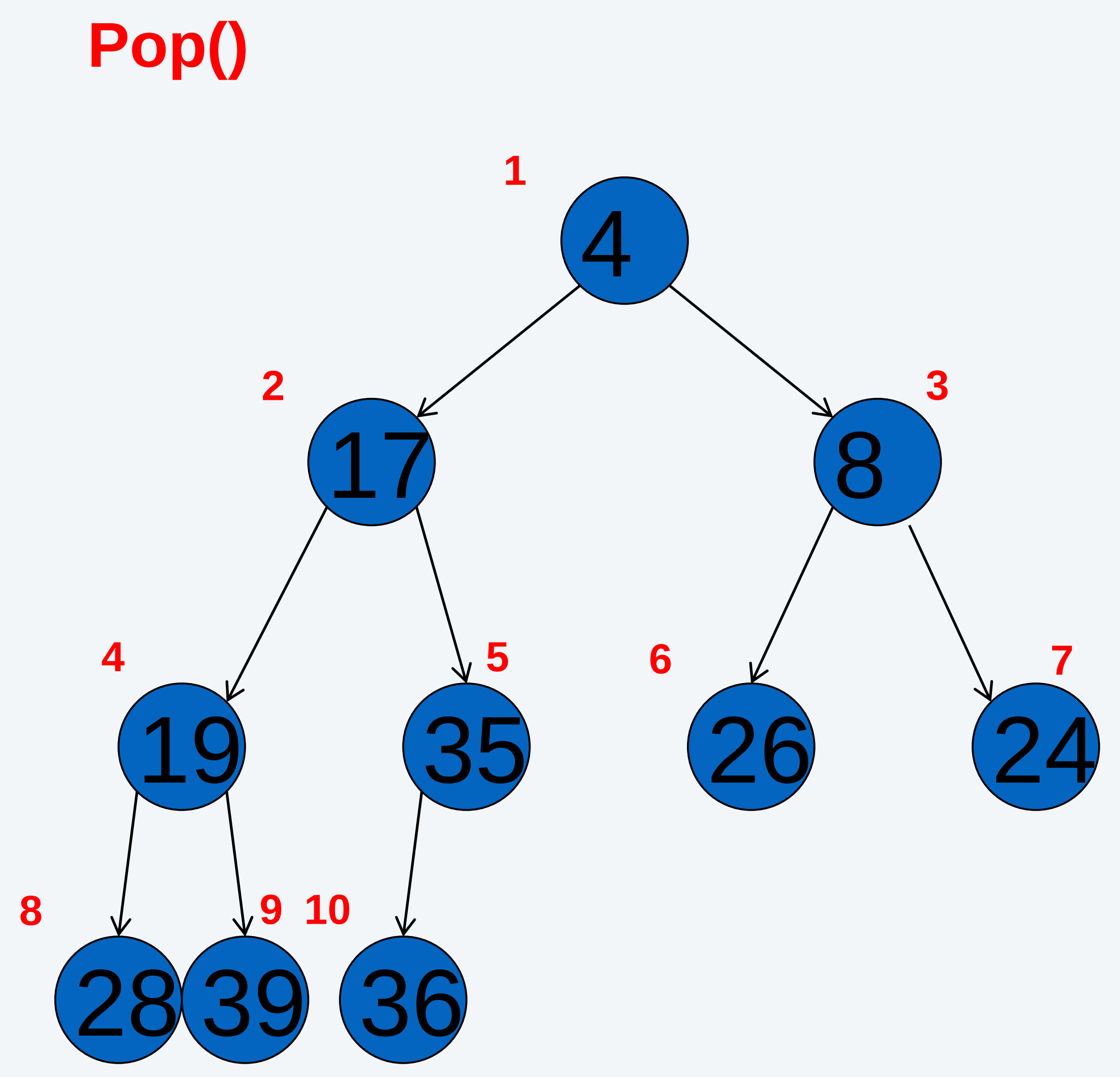
Exercise 2: Tracing Push

Draw the array and trace for min heap



Exercise 3: Trace Pop

Draw the array and trace for min heap



Exercise 4: Converting An Array to a Min Heap

0	1	2	3	4	5	6	7	8
em	28	9	18	10	35	14	7	19

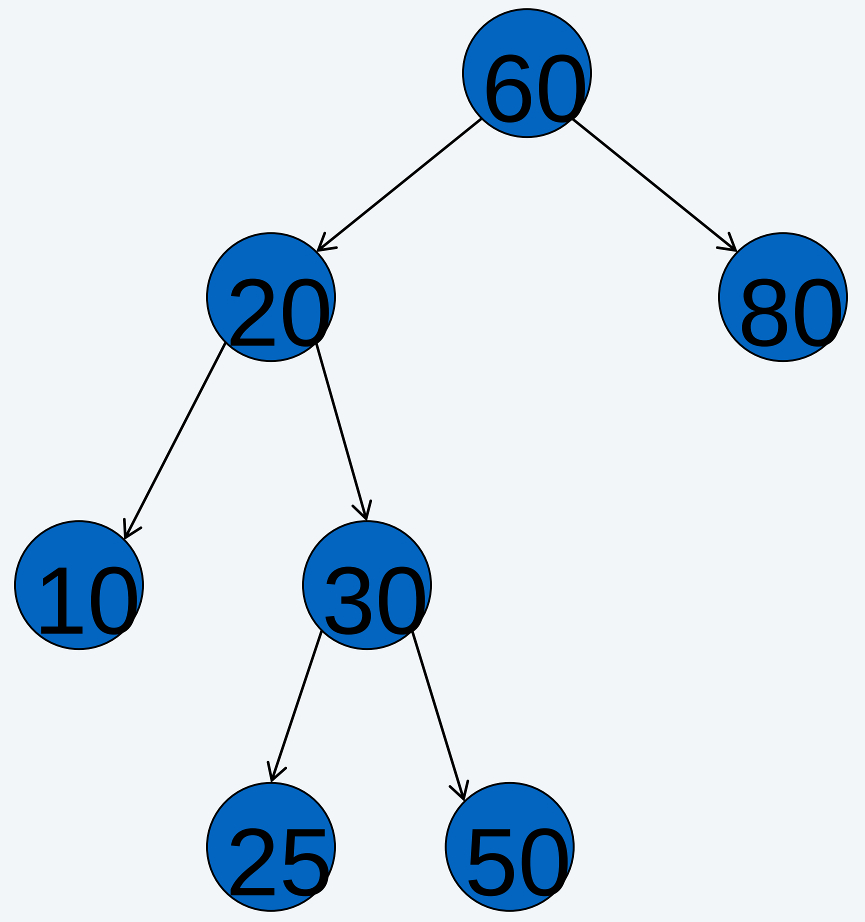
Original Array

1. Draw this array as a complete binary tree. Verify that it is not a min-heap.
2. Assume all leaf nodes are valid heaps
3. Then from first non-leaf node apply trickleDown or heapify. First non-leaf node is at index 4. (Why?)
4. Apply heapify on node at index 3.
5. Apply heapify on node at index 2.
6. Apply heapify on node at index 1.
7. Can you verify that this is a min-heap now?
8. Draw the min heap as an array again.
9. Can you show that this algorithm is O(n) where n is the number of elements in the array?

Exercise 5: Recursive Tree Traversals

Trace the following pseudocode for preorder, in order, and post-order traversal on the tree below.

```
// Node definition
struct TNode
{
  int val;
  shared_ptr<TNode> left;
  shared_ptr<TNode> right;
};
```



```
Preorder(shared_ptr<TNode> t)
{  if t == NULL return
   process(t) // print val.
   Preorder(t->left)
   Preorder(t->right)
}
```

```
Inorder(shared_ptr<Tnode> t)
{  if t == NULL return
   Inorder(t->left)
   process(t) // print val.
   Inorder(t->right)
}
```

```
Postorder(shared_ptr<Tnode> t)
{  if t == NULL return
   Postorder(t->left)
   Postorder(t->right)
   process(t) // print val.
}
```

Exercise 6: Recursive Tree Functions

Starter code:
https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/recursetree.cpp

1. Implement the count function recursively. The function should return the number of nodes in the tree.
2. Implement the height function recursively. The function should return the height of the tree.
3. Hint: You can use helper functions.