Problem: Implement a min heap with extract_min and insert methods.

Constraints

- Can we assume the inputs are ints?
    - Yes
- Can we assume this fits memory?
    - Yes

Test Cases

- Extract min of an empty tree
- Extract min general case
- Insert into an empty tree
- Insert general case (left and right insert)
-          _5_
-         /   \
-        20   15
-       / \   / \
-      22 40 25
-
- extract_min(): 5
-
-           _15_
-         /   \
-        20   25
-       / \   / \
-      22 40
-
- insert(2):
-
-           _2_
-         /   \
-        20   5
-       / \   / \
-      22 40 25 15

Algorithm

A heap is a complete binary tree where each node is smaller than its children.

extract_min

- _5_
- /   \
- 20   15
- / \   / \
- 22 40 25
-
- **Save the root as the value to be returned: 5**
- **Move the right most element to the root: 25**
-
- _25_
- /   \
- 20   15
- / \   / \
- 22 40
-
- **Bubble down 25: Swap 25 and 15 (the smaller child)**
-
- _15_
- /   \
- 20   25
- / \   / \
- 22 40
-
- **Return 5**

We'll use an array to represent the tree, here are the indices:

- _0_
- /   \
- 1   2
- / \   / \
- 3   4

To get to a child, we take 2 *index + 1 (left child) or 2* index + 2 (right child).

For example, the right child of index 1 is 2 * 1 + 2 = 4.

Complexity:

- **Time: O(lg(n))**

- **Space: O(lg(n)) for the recursion depth (tree height), or O(1) if using an iterative approach**

insert

- **_5_**
- **/ **
- **20 15**
- **/ \ / **
- **22 40 25**
-
- **insert(2):**
- **Insert at the right-most spot to maintain the heap property.**
-
- **_5_**
- **/ **
- **20 15**
- **/ \ / **
- **22 40 25 2**
-
- **Bubble up 2: Swap 2 and 15**
-
- **_5_**
- **/ **
- **20 2**
- **/ \ / **
- **22 40 25 15**
-
- **Bubble up 2: Swap 2 and 5**
-
- **_2_**
- **/ **
- **20 5**
- **/ \ / **
- **22 40 25 15**

We'll use an array to represent the tree, here are the indices:

- **_0_**
- **/ **
- **1 2**
- **/ \ / **
- **3 4 5 6**

To get to a parent, we take (index - 1) // 2.

For example, the parent of index 6 is (6 - 1) // 2 = 2.

Complexity:

- **Time: O(lg(n))**
- **Space: O(lg(n)) for the recursion depth (tree height), or O(1) if using an iterative approach**
-