

Section 05: Heaps + Midterm Review

Heap Problems

1. Ternary Heaps

Consider the following sequence of numbers:

5, 20, 10, 6, 7, 3, 1, 2

- (a) Insert these numbers into a min-heap where each node has up to *three* children, instead of two. (So, instead of inserting into a binary heap, we're inserting into a ternary heap.)
Draw out the tree representation of your completed ternary heap.
- (b) Draw out the array representation of the above tree. In your array representation, you should start at index 0 (not index 1).
- (c) Given a node at index i , write a formula to find the index of the parent.
- (d) Given a node at index i , write a formula to find the j -th child. Assume that $0 \leq j < 3$.

2. Heaps – More Basics

- (a) Insert the following sequence of numbers into a *min heap*:

[10, 7, 15, 17, 12, 20, 6, 32]

- (b) Now, insert the same values into a *max heap*.
- (c) Now, insert the same values into a *min heap*, but use Floyd's buildHeap algorithm.
- (d) Insert 1, 0, 1, 1, 0 into a *min heap*.
- (e) Call removeMin on the min heap stored as the following array: [2, 5, 7, 8, 10, 9]

3. Sorting and Reversing (with Heaps)

- (a) Suppose you have an array representation of a heap. Must the array be sorted?

- (b) Suppose you have a sorted array (in increasing order). Must it be the array representation of a valid min-heap?
- (c) You have an array representation of a min-heap. If you reverse the array, does it become an array representation of a max-heap?
- (d) Describe the most efficient algorithm you can think of to convert the array representation of a min-heap into a max-heap. What is its running time?

4. Project Prep: Contains

You just finished implementing your heap of ints when your boss tells you to add a new method called `contains`. Your solution should not, in general, examine every element in the heap (do it recursively!)

- (a) How efficient do you think you can make this method?
- (b) Write code for `contains`. Remember that `heapArray` starts at index 0!

5. Challenge: Debugging Heaps of Problems

For this problem, we will consider a hypothetical hash table that uses linear probing and implements the `IDictionary` interface. Specifically, we will focus on analyzing and testing one potential implementation of the `remove` method.

- (a) Come up with at least 4 different test cases to test this `remove(...)` method. For each test case, describe what the expected outcome is (assuming the method is implemented correctly).

Try and construct test cases that check if the `remove(...)` method is correctly using the key's hash code. (You may assume that you can construct custom key objects that let you customize the behavior of the `equals(...)` and `hashCode()` method.)

- (b) Now, consider the following (buggy) implementation of the `remove(...)` method. List all the bugs you can find.
- (c) Briefly describe how you would fix these bug(s).