

PromptScratchpadOur Solution(s)Video Explanation

Run Code

Solution 1

```
14     var firstParentIndex = Double((array.count - 2) / 2)
15     firstParentIndex = firstParentIndex.rounded(.down)
16
17     for var currentIndex in (0 ... Int(firstParentIndex)).reversed() {
18         var endIndex = array.count - 1
19         siftDown(currentIndex: &currentIndex, endIndex: &endIndex, heap: &array)
20     }
21
22     return array
23 }
24
25 // O(log(n)) time | O(1) space
26 func siftDown(currentIndex: inout Int, endIndex: inout Int, heap: inout [Int]) {
27     var firstChildIndex = (2 * currentIndex) + 1
28     while firstChildIndex < endIndex {
29         var secondChildIndex = -1
30         let potentialSecondChild = (2 * currentIndex) + 2
31         if potentialSecondChild <= endIndex {
32             secondChildIndex = potentialSecondChild
33         }
34
35         var indexToSwap = -1
36         if secondChildIndex != -1, heap[secondChildIndex] < heap[firstChildIndex] {
37             indexToSwap = secondChildIndex
38         } else {
39             indexToSwap = firstChildIndex
40         }
41
42         if heap[indexToSwap] < heap[currentIndex] {
43             swap(firstIndex: currentIndex, secondIndex: indexToSwap, heap: &heap)
44             currentIndex = indexToSwap
45             firstChildIndex = (2 * currentIndex) + 1
46         } else {
47             return
48         }
49     }
50 }
51
52 // O(log(n)) time | O(1) space
53 func siftUp(currentIndex: inout Int, heap: inout [Int]) {
54     var parentIndex = Double((currentIndex - 1) / 2)
55     parentIndex = parentIndex.rounded(.down)
56
57     while currentIndex > 0, heap[currentIndex] < heap[Int(parentIndex)] {
58         swap(firstIndex: currentIndex, secondIndex: Int(parentIndex), heap: &heap)
59         currentIndex = Int(parentIndex)
60         parentIndex = Double((currentIndex - 1) / 2)
61     }
62 }
63
64 // O(1) time | O(1) space
65 func peek() -> Int {
66     return heap[0]
67 }
68
69 // O(log(n)) time | O(1) space
70 func remove() -> Int {
71     swap(firstIndex: 0, secondIndex: heap.count - 1, heap: &heap)
72
73     if let valueToRemove = heap.popLast() {
74         var currentIndex = 0
75         var endIndex = heap.count - 1
76         siftDown(currentIndex: &currentIndex, endIndex: &endIndex, heap: &heap)
77
78         return valueToRemove
79     }
80
81     return -1
82 }
83
84 // O(log(n)) time | O(1) space
85 func insert(value: Int) {
86     heap.append(value)
87     var currentIndex = heap.count - 1
88     siftUp(currentIndex: &currentIndex, heap: &heap)
89 }
90
91 // Generic swap function
92 func swap(firstIndex: Int, secondIndex: Int, heap: inout [Int]) {
93     let temp = heap[firstIndex]
94     heap[firstIndex] = heap[secondIndex]
95     heap[secondIndex] = temp
96 }
97 }
98 }
99 }
```