



Solution Review: Convert Max-Heap to Min-Heap

We'll cover the following



- Solution: Min Heapify all Parent Nodes
- Time Complexity

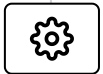
Solution: Min Heapify all Parent Nodes

#

```
1 def minHeapify(heap, index):
2     left = index * 2 + 1
3     right = (index * 2) + 2
4     smallest = index
5     # check if left child exists and is less than smallest
6     if len(heap) > left and heap[smallest] > heap[left]:
7         smallest = left
8     # check if right child exists and is less than smallest
9     if len(heap) > right and heap[smallest] > heap[right]:
10        smallest = right
11    # check if current index is not the smallest
12    if smallest != index:
13        # swap current index value with smallest
14        tmp = heap[smallest]
15        heap[smallest] = heap[index]
16        heap[index] = tmp
17        # minHeapify the new node
18        minHeapify(heap, smallest)
19    return heap
20
```



```
21
22 def convertMax(maxHeap):
23     # iterate from middle to first element
24     # middle to first indices contain all parent nodes
25     for i in range((len(maxHeap))//2, -1, -1):
26         # call minHeapify on all parent nodes
27         maxHeap = minHeapify(maxHeap, i)
28     return maxHeap
```



Remember that we can consider the given **maxHeap** to be a regular list of elements and reorder it so that it represents a min heap accurately. We do exactly that in this solution. The **convertMax()** function restores the heap property on all the nodes from the lowest parent node by calling the **minHeapify()** function on each.

Time Complexity#

As discussed [here](#), the time complexity of building a heap is **O(n)**.

Now, let's move on to another challenge regarding heaps in the next lesson!

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#) ⓘ



← Back

Next →

Challenge 1: Convert Max-Heap to Mi...

Challenge 2: Find k smallest elements ...

✓ Mark as Completed



Report an Issue

