Solution Review: Convert Max-Heap to Min-Heap

We'll cover the following

Solution: Min Heapify all Parent Nodes

```
def minHeapify(heap, index):
    left = index * 2 + 1
    right = (index * 2) + 2
    smallest = index
    # check if left child exists and is less than smallest
    if len(heap) > left and heap[smallest] > heap[left]:
        smallest = left
    # check if right child exists and is less than smallest
    if len(heap) > right and heap[smallest] > heap[right]:
        smallest = right
    # check if current index is not the smallest
    if smallest != index:
        # swap current index value with smallest
        tmp = heap[smallest]
        heap[smallest] = heap[index]
        heap[index] = tmp
        # minHeapify the new node
        minHeapify(heap, smallest)
    return heap

```

```
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!

Back

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

Next

Challenge 2: Find k smallest elements ...

Mark as Completed

Report an Issue