

PromptScratchpadOur Solution(s)Video Explanation

Run Code

Solution 1

```
16         self.updateMedian()
17
18     def rebalanceHeaps(self):
19         if self.lower.length - self.greater.length == 2:
20             self.greater.insert(self.lower.remove())
21         elif self.greater.length - self.lower.length == 2:
22             self.lower.insert(self.greater.remove())
23
24     def updateMedian(self):
25         if self.lower.length == self.greater.length:
26             self.median = (self.lower.peak() + self.greater.peak()) / 2
27         elif self.lower.length > self.greater.length:
28             self.median = self.lower.peak()
29         else:
30             self.median = self.greater.peak()
31
32     def getMedian(self):
33         return self.median
34
35
36 class Heap:
37     def __init__(self, comparisonFunc, array):
38         self.heap = self.buildHeap(array)
39         self.comparisonFunc = comparisonFunc
40         self.length = len(self.heap)
41
42     def buildHeap(self, array):
43         firstParentIdx = (len(array) - 2) // 2
44         for currentIdx in reversed(range(firstParentIdx + 1)):
45             self.siftDown(currentIdx, len(array) - 1, array)
46         return array
47
48     def siftDown(self, currentIdx, endIdx, heap):
49         childOneIdx = currentIdx * 2 + 1
50         while childOneIdx <= endIdx:
51             childTwoIdx = currentIdx * 2 + 2 if currentIdx * 2 + 2 <= endIdx else -1
52             if childTwoIdx != -1:
53                 if self.comparisonFunc(heap[childTwoIdx], heap[childOneIdx]):
54                     idxToSwap = childTwoIdx
55                 else:
56                     idxToSwap = childOneIdx
57             else:
58                 idxToSwap = childOneIdx
59             if self.comparisonFunc(heap[idxToSwap], heap[currentIdx]):
60                 self.swap(currentIdx, idxToSwap, heap)
61                 currentIdx = idxToSwap
62                 childOneIdx = currentIdx * 2 + 1
63             else:
64                 return
65
66     def siftUp(self, currentIdx, heap):
67         parentIdx = (currentIdx - 1) // 2
68         while currentIdx > 0:
69             if self.comparisonFunc(heap[currentIdx], heap[parentIdx]):
70                 self.swap(currentIdx, parentIdx, heap)
71                 currentIdx = parentIdx
72                 parentIdx = (currentIdx - 1) // 2
73             else:
74                 return
75
76     def peek(self):
77         return self.heap[0]
78
79     def remove(self):
80         self.swap(0, self.length - 1, self.heap)
81         valueToRemove = self.heap.pop()
82         self.length -= 1
83         self.siftDown(0, self.length - 1, self.heap)
84         return valueToRemove
85
86     def insert(self, value):
87         self.heap.append(value)
88         self.length += 1
89         self.siftUp(self.length - 1, self.heap)
90
91     def swap(self, i, j, array):
92         array[i], array[j] = array[j], array[i]
93
94
95 def MAX_HEAP_FUNC(a, b):
96     return a > b
97
98
99 def MIN_HEAP_FUNC(a, b):
100     return a < b
101
```