

PromptScratchpadOur Solution(s)Video Explanation

Run Code

Solution 1

```
...
45     constructor(comparisonFunc, array) {
46         this.heap = this.buildHeap(array);
47         this.comparisonFunc = comparisonFunc;
48         this.length = this.heap.length;
49     }
50
51     buildHeap(array) {
52         const firstParentIdx = Math.floor((array.length - 2) / 2);
53         for (let currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {
54             this.siftDown(currentIdx, array.length - 1, array);
55         }
56         return array;
57     }
58
59     siftDown(currentIdx, endIdx, heap) {
60         let childOneIdx = currentIdx * 2 + 1;
61         while (childOneIdx <= endIdx) {
62             const childTwoIdx = currentIdx * 2 + 2 <= endIdx ? currentIdx * 2 + 2 : -1;
63             let idxToSwap;
64             if (childTwoIdx !== -1) {
65                 if (this.comparisonFunc(heap[childTwoIdx], heap[childOneIdx])) {
66                     idxToSwap = childTwoIdx;
67                 } else {
68                     idxToSwap = childOneIdx;
69                 }
70             } else {
71                 idxToSwap = childOneIdx;
72             }
73             if (this.comparisonFunc(heap[idxToSwap], heap[currentIdx])) {
74                 this.swap(currentIdx, idxToSwap, heap);
75                 currentIdx = idxToSwap;
76                 childOneIdx = currentIdx * 2 + 1;
77             } else {
78                 return;
79             }
80         }
81     }
82
83     siftUp(currentIdx, heap) {
84         let parentIdx = Math.floor((currentIdx - 1) / 2);
85         while (currentIdx > 0) {
86             if (this.comparisonFunc(heap[currentIdx], heap[parentIdx])) {
87                 this.swap(currentIdx, parentIdx, heap);
88                 currentIdx = parentIdx;
89                 parentIdx = Math.floor((currentIdx - 1) / 2);
90             } else {
91                 return;
92             }
93         }
94     }
95
96     peek() {
97         return this.heap[0];
98     }
99
100    remove() {
101        this.swap(0, this.length - 1, this.heap);
102        const valueToRemove = this.heap.pop();
103        this.length--;
104        this.siftDown(0, this.length - 1, this.heap);
105        return valueToRemove;
106    }
107
108    insert(value) {
109        this.heap.push(value);
110        this.length++;
111        this.siftUp(this.length - 1, this.heap);
112    }
113
114    swap(i, j, heap) {
115        const temp = heap[j];
116        heap[j] = heap[i];
117        heap[i] = temp;
118    }
119 }
120
121 function MAX_HEAP_FUNC(a, b) {
122     return a > b;
123 }
124
125 function MIN_HEAP_FUNC(a, b) {
126     return a < b;
127 }
128
129 exports.ContinuousMedianHandler = ContinuousMedianHandler;
130
```