

PromptScratchpadOur Solution(s)Video Explanation

Run Code

Solution 1

```
--
58     }
59
60     public class Heap {
61         public List<int> heap = new List<int>();
62         public Func<int, int, bool> comparisonFunc;
63         public int length;
64
65         public Heap(Func<int, int, bool> func, List<int> array) {
66             this.heap = buildHeap(array);
67             this.comparisonFunc = func;
68             this.length = heap.Count;
69         }
70
71         public int peek() {
72             return heap[0];
73         }
74
75         public int remove() {
76             this.swap(0, heap.Count - 1);
77             int valueToRemove = heap[heap.Count - 1];
78             this.heap.RemoveAt(heap.Count - 1);
79             this.length -= 1;
80             this.siftDown(0, heap.Count - 1, heap);
81             return valueToRemove;
82         }
83
84         public void Insert(int value) {
85             this.heap.Add(value);
86             this.length += 1;
87             this.siftUp(heap.Count - 1, heap);
88         }
89
90         public List<int> buildHeap(List<int> array) {
91             int firstParentIdx = (array.Count - 2) / 2;
92             for (int currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {
93                 this.siftDown(currentIdx, array.Count - 1, array);
94             }
95             return array;
96         }
97
98         public void siftDown(int currentIdx, int endIdx, List<int> heap) {
99             int childOneIdx = currentIdx * 2 + 1;
100             while (childOneIdx <= endIdx) {
101                 int childTwoIdx = currentIdx * 2 + 2 <=
102                     endIdx ? currentIdx * 2 + 2 : -1;
103                 int idxToSwap;
104                 if (childTwoIdx != -1) {
105                     if (comparisonFunc(heap[childTwoIdx], heap[childOneIdx])) {
106                         idxToSwap = childTwoIdx;
107                     } else {
108                         idxToSwap = childOneIdx;
109                     }
110                 } else {
111                     idxToSwap = childOneIdx;
112                 }
113                 if (comparisonFunc(heap[idxToSwap], heap[currentIdx])) {
114                     swap(currentIdx, idxToSwap);
115                     currentIdx = idxToSwap;
116                     childOneIdx = currentIdx * 2 + 1;
117                 } else {
118                     return;
119                 }
120             }
121         }
122
123         public void siftUp(int currentIdx, List<int> heap) {
124             int parentIdx = (currentIdx - 1) / 2;
125             while (currentIdx > 0) {
126                 if (comparisonFunc(heap[currentIdx], heap[parentIdx])) {
127                     swap(currentIdx, parentIdx);
128                     currentIdx = parentIdx;
129                     parentIdx = (currentIdx - 1) / 2;
130                 } else {
131                     return;
132                 }
133             }
134         }
135
136         public void swap(int i, int j) {
137             int temp = this.heap[j];
138             this.heap[j] = this.heap[i];
139             this.heap[i] = temp;
140         }
141     }
142 }
143
```