

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 package main
4
5 type MinHeap []int
6
7 func NewMinHeap(array []int) *MinHeap {
8     heap := MinHeap(array)
9     ptr := &heap
10    ptr.BuildHeap(array)
11    return ptr
12 }
13
14 // O(n) time | O(1) space
15 func (h *MinHeap) BuildHeap(array []int) {
16     first := (len(array) - 2) / 2
17     for currentIndex := first + 1; currentIndex >= 0; currentIndex-- {
18         h.siftDown(currentIndex, len(array)-1)
19     }
20 }
21
22 // O(log(n)) time | O(1) space
23 func (h *MinHeap) siftDown(currentIndex, endIndex int) {
24     childOneIdx := currentIndex*2 + 1
25     for childOneIdx <= endIndex {
26         childTwoIdx := -1
27         if currentIndex*2+2 <= endIndex {
28             childTwoIdx = currentIndex*2 + 2
29         }
30         indexToSwap := childOneIdx
31         if childTwoIdx > -1 && (*h)[childTwoIdx] < (*h)[childOneIdx] {
32             indexToSwap = childTwoIdx
33         }
34         if (*h)[indexToSwap] < (*h)[currentIndex] {
35             h.swap(currentIndex, indexToSwap)
36             currentIndex = indexToSwap
37             childOneIdx = currentIndex*2 + 1
38         } else {
39             return
40         }
41     }
42 }
43
44 // O(log(n)) time | O(1) space
45 func (h *MinHeap) siftUp() {
46     currentIndex := h.length() - 1
47     parentIndex := (currentIndex - 1) / 2
48     for currentIndex > 0 {
49         current, parent := (*h)[currentIndex], (*h)[parentIndex]
50         if current < parent {
51             h.swap(currentIndex, parentIndex)
52             currentIndex = parentIndex
53             parentIndex = (currentIndex - 1) / 2
54         } else {
55             return
56         }
57     }
58 }
59
60 // O(1) time | O(1) space
61 func (h MinHeap) Peek() int {
62     if len(h) == 0 {
63         return -1
64     }
65     return h[0]
66 }
67
68 // O(log(n)) time | O(1) space
69 func (h *MinHeap) Remove() int {
70     l := h.length()
71     h.swap(0, l-1)
72     peeked := (*h)[l-1]
73     *h = (*h)[0 : l-1]
74     h.siftDown(0, l-1)
75     return peeked
76 }
77
78 // O(log(n)) time | O(1) space
79 func (h *MinHeap) Insert(value int) {
80     *h = append(*h, value)
81     h.siftUp()
82 }
83
84 func (h MinHeap) swap(i, j int) {
85     h[i], h[j] = h[j], h[i]
86 }
87
88 func (h MinHeap) length() int {
89     return len(h)
90 }
```

Solution 1

Solution 2

Solution 3

```
1 package main
2
3 // Do not edit the class below except for the buildHeap,
4 // siftDown, siftUp, peek, remove, and insert methods.
5 // Feel free to add new properties and methods to the class.
6 type MinHeap []int
7
8 func NewMinHeap(array []int) *MinHeap {
9     // Do not edit the lines below.
10    heap := MinHeap(array)
11    ptr := &heap
12    ptr.BuildHeap(array)
13    return ptr
14 }
15
16 func (h *MinHeap) BuildHeap(array []int) {
17     // Write your code here.
18 }
19
20 func (h *MinHeap) siftDown(currentIndex, endIndex int) {
21     // Write your code here.
22 }
23
24 func (h *MinHeap) siftUp() {
25     // Write your code here.
26 }
27
28 func (h MinHeap) Peek() int {
29     // Write your code here.
30     return -1
31 }
32
33 func (h *MinHeap) Remove() int {
34     // Write your code here.
35     return -1
36 }
37
38 func (h *MinHeap) Insert(value int) {
39     // Write your code here.
40 }
41
```

Custom Output

Raw Output

Submit Code

Run or submit code when you're ready.