

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 using namespace std;
5
6 void buildMaxHeap(vector<int> &array);
7 void siftDown(int currentIdx, int endIdx, vector<int> &heap);
8
9 // Best: O(nlog(n)) time | O(1) space
10 // Average: O(nlog(n)) time | O(1) space
11 // Worst: O(nlog(n)) time | O(1) space
12 vector<int> heapSort(vector<int> array) {
13     buildMaxHeap(array);
14     for (int endIdx = array.size() - 1; endIdx > 0; endIdx--) {
15         swap(array[0], array[endIdx]);
16         siftDown(0, endIdx - 1, array);
17     }
18     return array;
19 }
20
21 void buildMaxHeap(vector<int> &array) {
22     int firstParentIdx = (array.size() - 2) / 2;
23     for (int currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {
24         siftDown(currentIdx, array.size() - 1, array);
25     }
26 }
27
28 void siftDown(int currentIdx, int endIdx, vector<int> &heap) {
29     int childOneIdx = currentIdx * 2 + 1;
30     while (childOneIdx <= endIdx) {
31         int childTwoIdx = currentIdx * 2 + 2 <= endIdx ? currentIdx * 2 +
32         int idxToSwap;
33         if (childTwoIdx != -1 && heap.at(childTwoIdx) > heap.at(childOneId
```

Solution 1

Solution 2

Solution 3

```
1 #include <vector>
2 using namespace std;
3
4 vector<int> heapSort(vector<int> array) {
5     // Write your code here.
6     return {};
7 }
8
```

Our Tests

Custom Output

Submit Code

1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.

2

3 #include <vector>

4 using namespace std;

5

6 void buildMaxHeap(vector<int> &array);

7 void siftDown(int currentIdx, int endIdx, vector<int> &heap);

8

9 // Best: O(nlog(n)) time | O(1) space

10 // Average: O(nlog(n)) time | O(1) space

11 // Worst: O(nlog(n)) time | O(1) space

12 vector<int> heapSort(vector<int> array) {

13 buildMaxHeap(array);

14 for (int endIdx = array.size() - 1; endIdx > 0; endIdx--) {

15 swap(array[0], array[endIdx]);

16 siftDown(0, endIdx - 1, array);

17 }

18 return array;

19 }

20

21 void buildMaxHeap(vector<int> &array) {

22 int firstParentIdx = (array.size() - 2) / 2;

23 for (int currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {

24 siftDown(currentIdx, array.size() - 1, array);

25 }

26 }

27

28 void siftDown(int currentIdx, int endIdx, vector<int> &heap) {

29 int childOneIdx = currentIdx * 2 + 1;

30 while (childOneIdx <= endIdx) {

31 int childTwoIdx = currentIdx * 2 + 2 <= endIdx ? currentIdx * 2 +

32 int idxToSwap;

33 if (childTwoIdx != -1 && heap.at(childTwoIdx) > heap.at(childOneId

1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.

2

3 #include <vector>

4 using namespace std;

5

6 void buildMaxHeap(vector<int> &array);

7 void siftDown(int currentIdx, int endIdx, vector<int> &heap);

8

9 // Best: O(nlog(n)) time | O(1) space

10 // Average: O(nlog(n)) time | O(1) space

11 // Worst: O(nlog(n)) time | O(1) space

12 vector<int> heapSort(vector<int> array) {

13 buildMaxHeap(array);

14 for (int endIdx = array.size() - 1; endIdx > 0; endIdx--) {

15 swap(array[0], array[endIdx]);

16 siftDown(0, endIdx - 1, array);

17 }

18 return array;

19 }

20

21 void buildMaxHeap(vector<int> &array) {

22 int firstParentIdx = (array.size() - 2) / 2;

23 for (int currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {

24 siftDown(currentIdx, array.size() - 1, array);

25 }

26 }

27

28 void siftDown(int currentIdx, int endIdx, vector<int> &heap) {

29 int childOneIdx = currentIdx * 2 + 1;

30 while (childOneIdx <= endIdx) {

31 int childTwoIdx = currentIdx * 2 + 2 <= endIdx ? currentIdx * 2 +

32 int idxToSwap;

33 if (childTwoIdx != -1 && heap.at(childTwoIdx) > heap.at(childOneId

```
10
11
12 def test_case_07, 1000, 0
13   return test_expected = [0, 0]
14   assert_equal(test, 0) == expected
15 end
16
17 def test_case_07, 1000, 0
18   return test_expected = [0, 0]
19   assert_equal(test, 0) == expected
20 end
21
22 def test_case_07, 1000, 0
23   return test_expected = [0, 0, 0]
24   assert_equal(test, 0, 0) == expected
25 end
```

Run or submit code when you're ready.