Copilot

Sure, let's go through the operations step by step.

Heap after each insert operation

1. Insert 20:

```
   20
```

1. Insert 4:

```
   20
  /
 4
```

1. Insert 24:

```
   24
  /  \
 4   20
```

1. Insert 100:

```
     100
    /   \
  24    20
 /
4
```

1. Insert 0:

```
     100
    /   \
  24    20
 / \
4   0
```

1. Insert 50:

```
       100
      /   \
    50    20
   / \   /
24  0  4
```

Heap after each deleteMax operation

1. DeleteMax (remove 100):

```
    50
   /  \
 24   20
/ \
0   4
```

1. DeleteMax (remove 50):

```
   24
```

```
  /  \
 4   20
/
0
```

1. DeleteMax (remove 24):

```
  20
 /  \
4    0
```

This is the final state of the heap after three deleteMax operations. The heap property is maintained at each step. The root is always the maximum element, and each parent node is greater than or equal to its child nodes. The shape property is also maintained, as the heap is always a complete binary tree. The operations are performed in O(log n) time, where n is the number of elements in the heap. This is because we only need to sift up or sift down along a path from the root to a leaf. The space complexity is O(1), as we only use a constant amount of space to store indices and temporary variables. The correctness of the operations is justified by the maintenance of the heap and shape properties. The heap property ensures that the root is always the maximum element, and the shape property ensures that the heap is a complete binary tree. Therefore, the operations are correct and efficient. I hope this helps! Let me know if you have any other questions. 😊