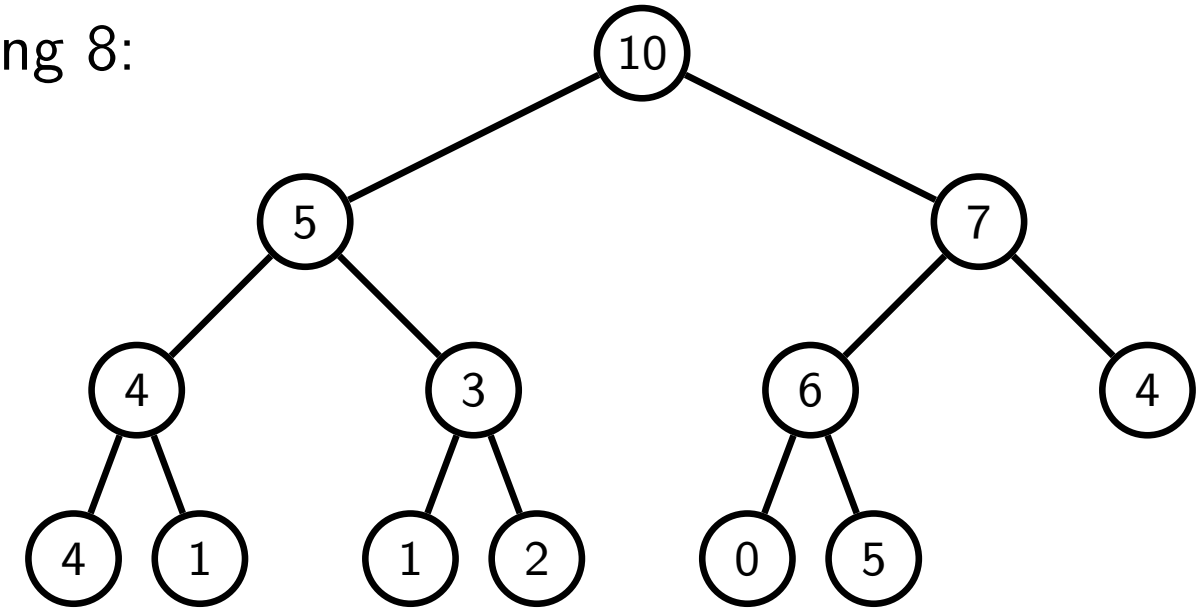
Binary Heap Tree Insertion

Insertion

Idea: Insert the value at the end of the last level and then keep bubbling it up as long as it is larger than its parent.

Inserting 8:

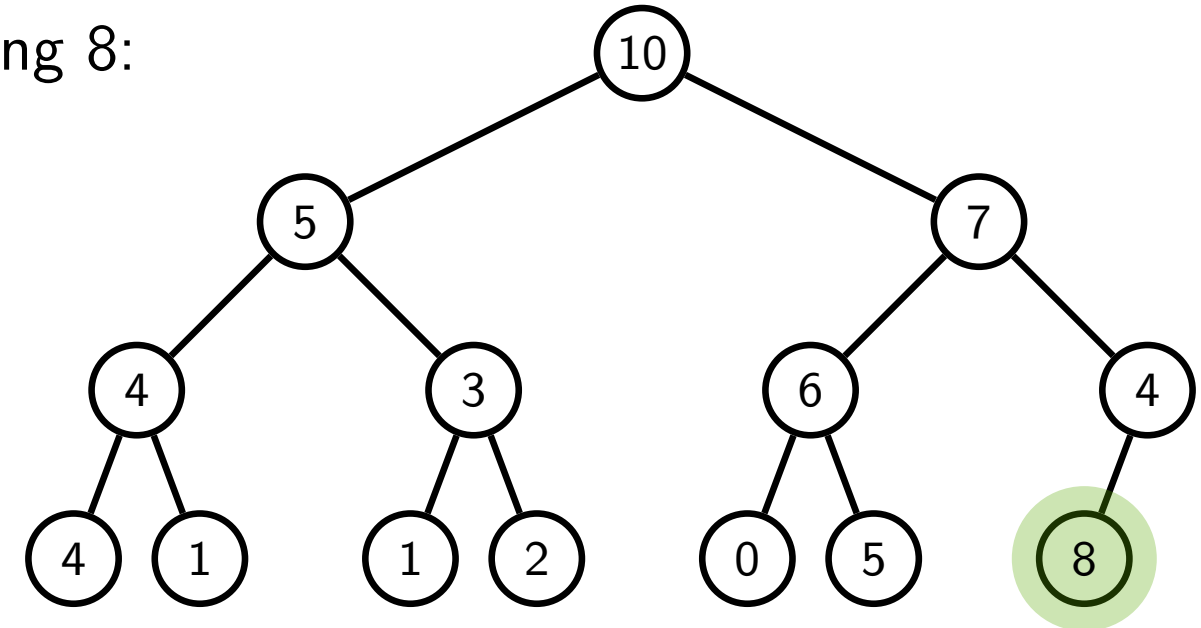


As we bubble up, when we swap the value of a node i with that of its parent, we don't have to compare i with its sibling, because if the value of i is greater than that of its parent, then it must be greater than that of its sibling because of the Binary Heap Tree property.

Insertion

Idea: Insert the value at the end of the last level and then keep bubbling it up as long as it is larger than its parent.

Inserting 8:

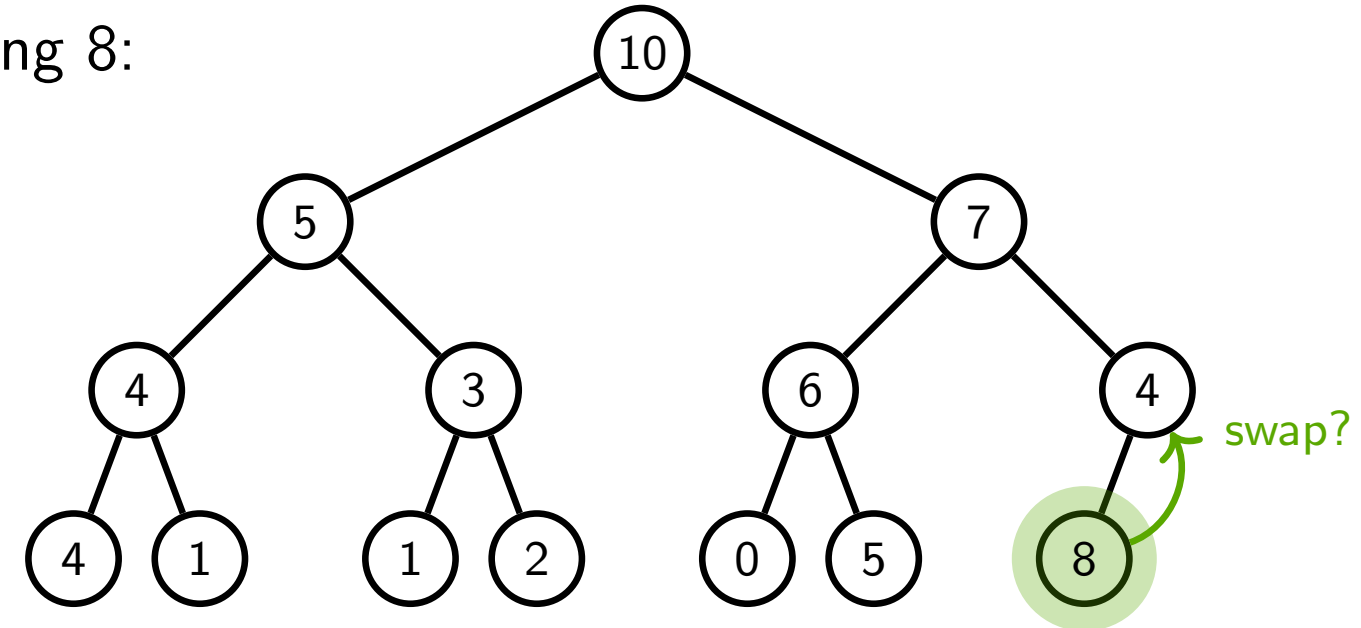


As we bubble up, when we swap the value of a node i with that of its parent, we don't have to compare i with its sibling, because if the value of i is greater than that of its parent, then it must be greater than that of its sibling because of the Binary Heap Tree property.

Insertion

Idea: Insert the value at the end of the last level and then keep bubbling it up as long as it is larger than its parent.

Inserting 8:

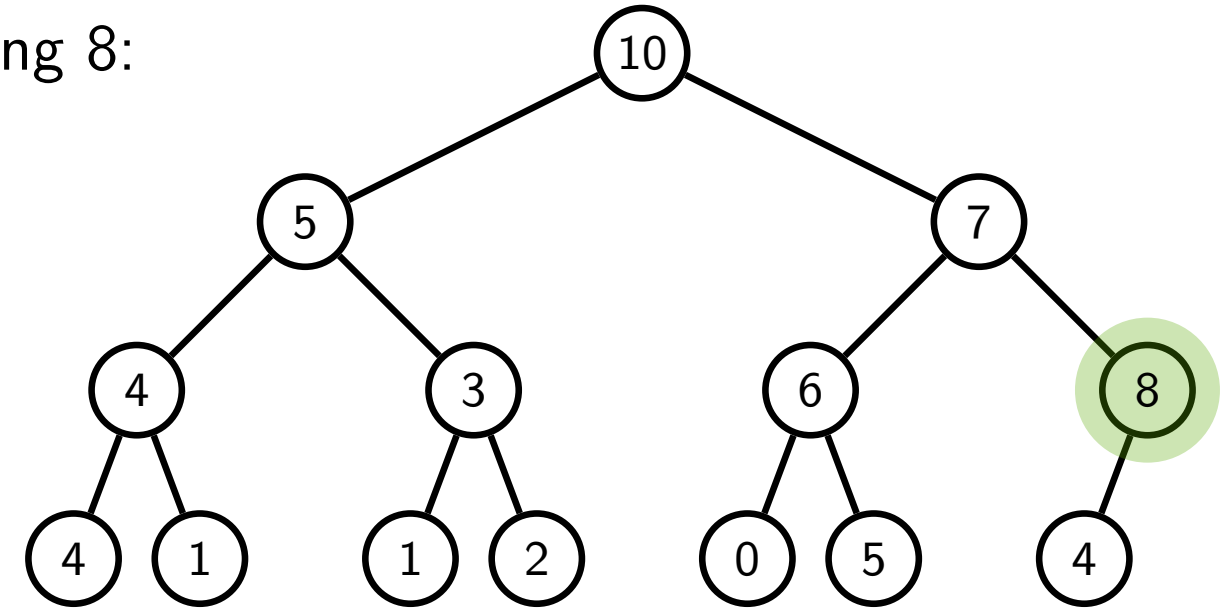


As we bubble up, when we swap the value of a node i with that of its parent, we don't have to compare i with its sibling, because if the value of i is greater than that of its parent, then it must be greater than that of its sibling because of the Binary Heap Tree property.

Insertion

Idea: Insert the value at the end of the last level and then keep bubbling it up as long as it is larger than its parent.

Inserting 8:

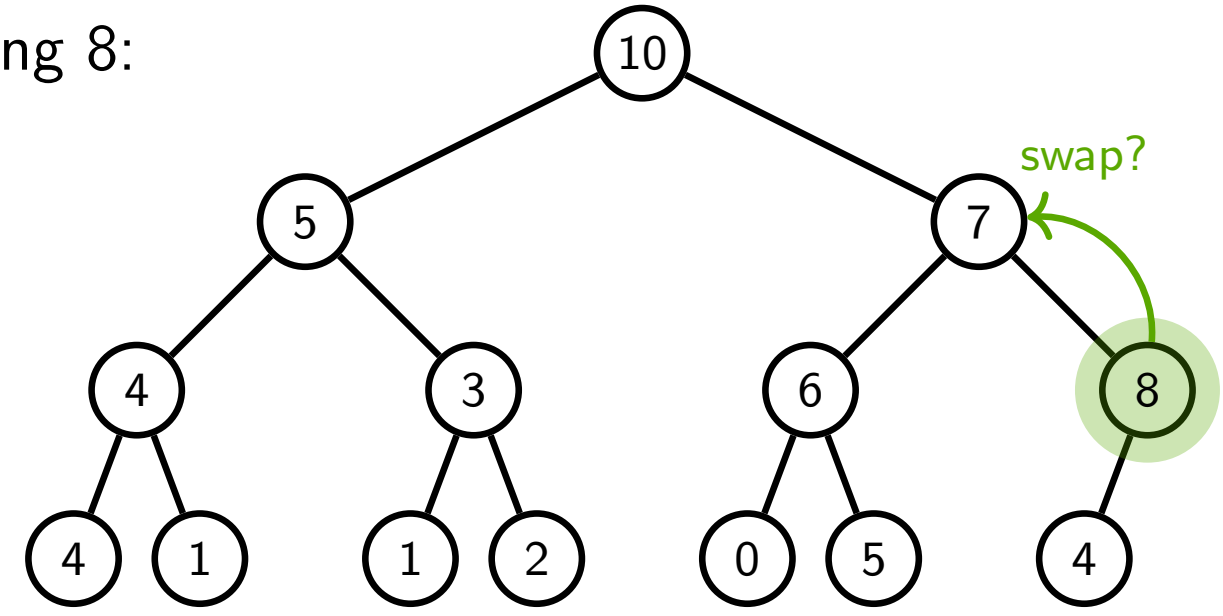


As we bubble up, when we swap the value of a node i with that of its parent, we don't have to compare i with its sibling, because if the value of i is greater than that of its parent, then it must be greater than that of its sibling because of the Binary Heap Tree property.

Insertion

Idea: Insert the value at the end of the last level and then keep bubbling it up as long as it is larger than its parent.

Inserting 8:

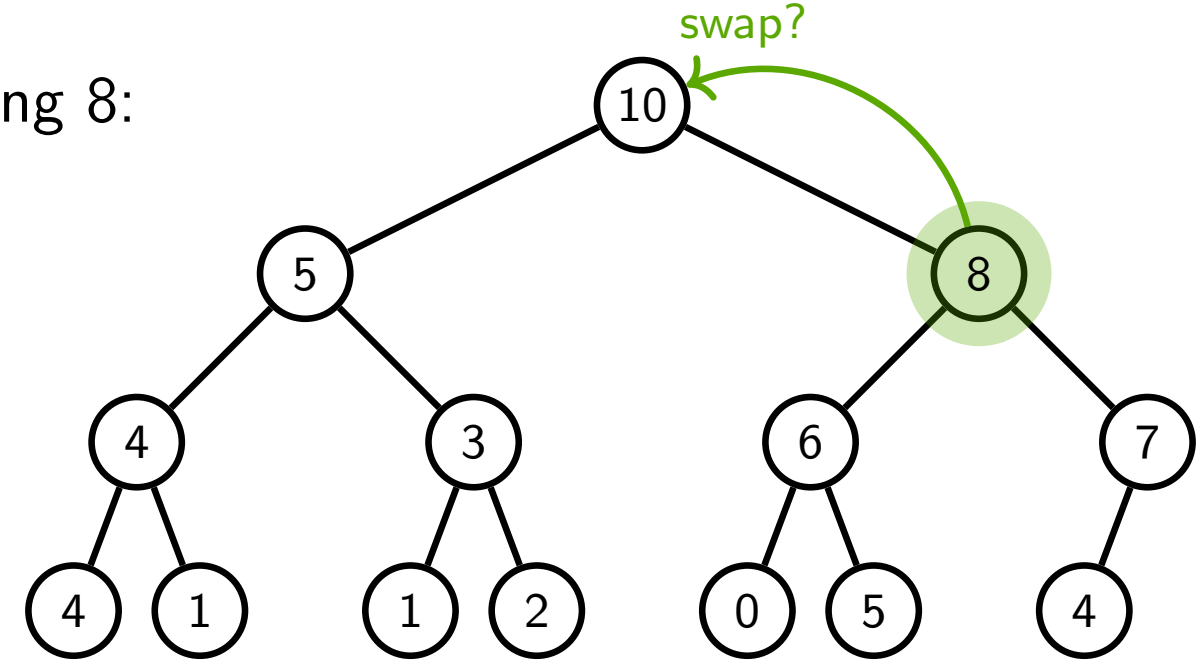


As we bubble up, when we swap the value of a node i with that of its parent, we don't have to compare i with its sibling, because if the value of i is greater than that of its parent, then it must be greater than that of its sibling because of the Binary Heap Tree property.

Insertion

Idea: Insert the value at the end of the last level and then keep bubbling it up as long as it is larger than its parent.

Inserting 8:

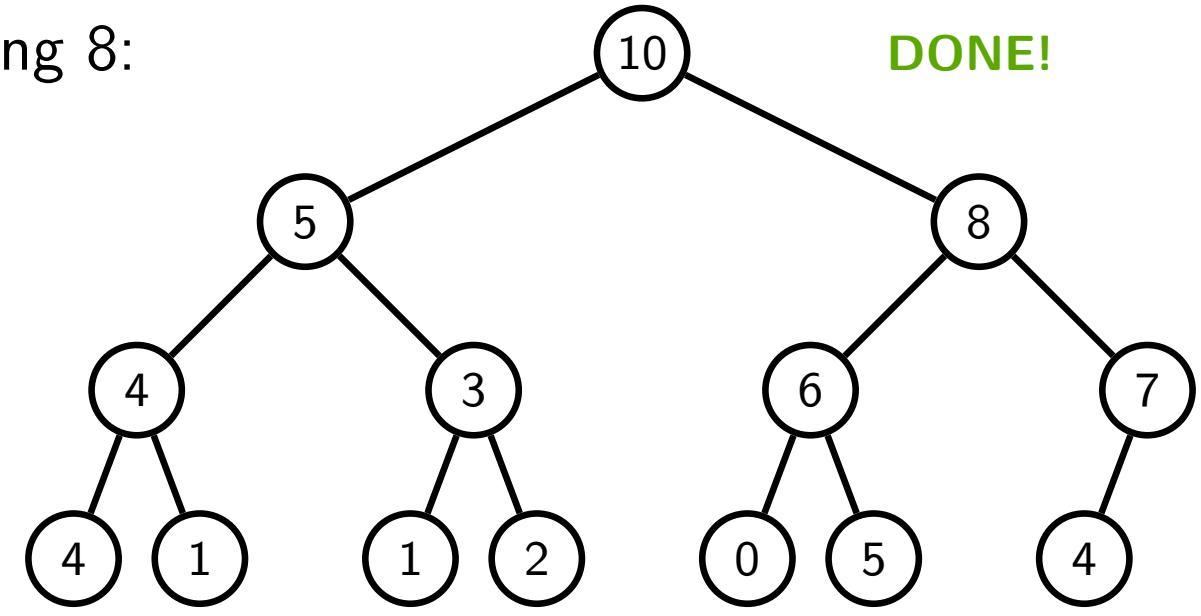


As we bubble up, when we swap the value of a node i with that of its parent, we don't have to compare i with its sibling, because if the value of i is greater than that of its parent, then it must be greater than that of its sibling because of the Binary Heap Tree property.

Insertion

Idea: Insert the value at the end of the last level and then keep bubbling it up as long as it is larger than its parent.

Inserting 8:



As we bubble up, when we swap the value of a node i with that of its parent, we don't have to compare i with its sibling, because if the value of i is greater than that of its parent, then it must be greater than that of its sibling because of the Binary Heap Tree property.

Insert (pseudocode)

```
1  public void insert(int p) {
2    if (n == MAXSIZE)
3      throw HeapFullException;
4    n = n + 1;
5    heap[n] = p;  // insert the new value as the last
6                  // node of the last level
7    bubbleUp(n);  // and bubble it up
8  }
```

```
1  private void bubbleUp(int i) {
2    if (i == 1) return;  // i is the root
3
4    if (heap[i] > heap[parent(i)]) {
5      swap heap[i] and heap[parent(i)];
6      bubbleUp(parent(i));
7    }
8  }
```