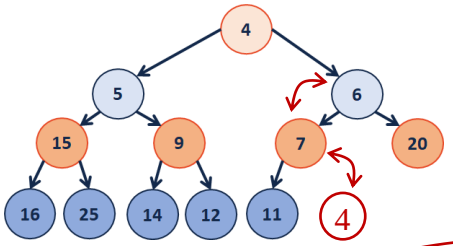
Formally, a binary tree T is a minHeap if:

- Heap-Shape Property: The heap is a complete binary tree
- Heap-Order Property: All nodes (except root) have a key
- value greater than or equal to their parent's key value.

Heap Operation: insert / bubbleUp:



–	4	5	6	15	9	7	20	16	25	14	12	11	4		

–	4	5	6	15	9	4	20	16	25	14	12	11	7		

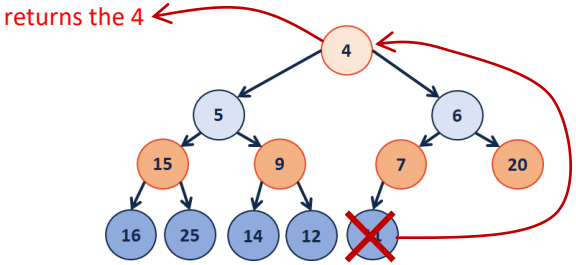
–	4	5	4	15	9	6	20	16	25	14	12	11	7		

Heap.java (partial)

```
1  void insert(int key) {
2    // Check to ensure there's space to insert an element
3    if (size==CAPACITY) { throw new FullHeapException() }
4
5    // Insert the new element at the end of the array
6    data[++size] = key;
7
8    bubbleUp(size); // Restore the heap property
9  }
```

```
31 void bubbleUp( int index ) {
32   if ( index > 1 ) {
33     if ( data[index] < data [ index / 2 ] ) {
34       swap(data, index, index/2);
35       bubbleUp( index / 2 );
36     }
37   }
38 }
```

Heap Operation: removeMin / bubbleDown



–	4	5	6	15	9	7	20	16	25	14	12	11			

–	11	5	6	15	9	7	20	16	25	14	12	4			

–	5	11	6	15	9	7	20	16	25	14	12				

–	5	9	6	15	11	7	20	16	25	14	12				

Heap.java (partial)

```
1  int removeMin() {
2    // Swap with the last value
3    int minValue = data[1];
4    data[1] = data[size];
5    data[size] = minValue;
6    size--;
7
8    bubbleDown(1); // Restore the heap property
9    return minValue; // Return the minimum value
10 }
```

```
1  void bubbleDown(int index) {
2    if ( !isLeaf(index) ) {
3      int minChildIndex = minChild(index);
4      if ( data[index] > data[minChildIndex] ) {
5        swap(data, index, minChildIndex);
6        bubbleDown( minChildIndex );
7      }
8    }
9  }
10
```