Buildheap:

- Sub-trees are made into heaps in a bottom up fashion starting with the largest index to the root.
- Loop Invariant: All subtrees rooted at indices heapsize to heapsize-i-1 (inclusive) are heaps after the i^{th} iteration.
    - Base Case: i = 1, node at heapsize is a leaf and remains a leaf after heapify(heapsize) so it is true.
    - Inductive Step: Subtrees rooted at indices heapsize to heapsize-i-1 are subtrees, during the i + 1 iteration: lchild and rchild of node[heapsize-i-2] are heaps as they belong to the above range, therefore after heapify subtree at index (heapsize-i-2) becomes a heap.
    - Termination: Loop terminates after heapsize iterations, when index 1 belongs to the range due to which tree at index 1 is a heap.

- In the Slides heapify being performed at each index is shown.

Heapsort:

- In each iteration of heapsort, the maximum element is swapped with the last element in the heap and removed by decreasing the heapsize by 1. As lchild and rchild of node1 are heaps before swapping, heapify(1) maintains the heap property.
- Removed nodes are shown in red color in the slides
- Loop Invariant: After i iterations, the first i greatest elements are in sorted order from n-i+1 to n, in the array
    - Base Case: After 1 iteration the greatest element(by max heap property) is at the heapsize+1 position in the array.
    - Inductive Step: During the $i+1^{th}$ iteration the greatest element (just lesser than all the ones not in the heap (because of I.H.)) is placed at its right position before all the elements not in the heap and greater than it (which are in sorted order) heapsize decreases by one, so i+1 greatest elements are in sorted order from heapsize+1 to n in the array after i+1 iterations

- Termination Step: The loop terminates after n - 1 iteration, therefore the first n-1 greatest elements are in 2 to n, therefore the array is sorted.

Heapify:

- The two nodes which are going to be swapped are shown in magenta color
- If the left and right subtrees are heaps then heapify(root) makes the tree at root a heap
- Loop Invariant: All the nodes in the levels above and below the current root satisfy the max heap property (i.e. lchild <= node and rchild <= node).

- After the current iteration the root is made to satisfy the heap property.
- Termination occurs when the current root satisfies the heap property.
- At this point all nodes below and above the current node and the node itself satisfy the heap property, therefore the subtree rooted at the initial root is a heap.