Required Algorithms For Heap Sort

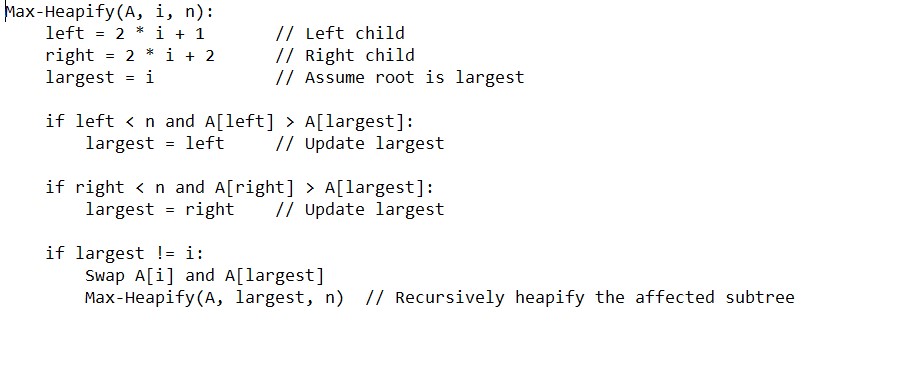
1-Max-Heapify:

-Ensures that a subtree rooted at index i satisfies the heap property.

-Assume the heap is stored as an array.

# Input:

* A: Array
* i: index
* n: size of heap



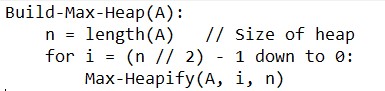
2-Build Max-Heap:

-Build a max heap from an unsorted array.

-We start heapifying from the last non-leaf node up to the root.

# Input:

-A: Array

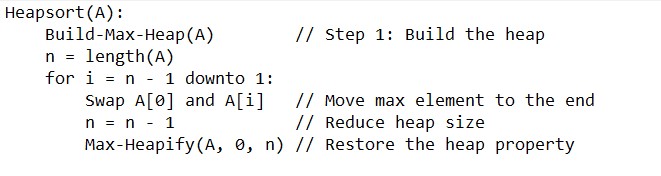


3-Heapsort:

-Sorts the array using the max-heap property.

# Input:

-A: Array



==================================================

Analysis of Algorithms (Time Complexity)

1-Max-Heapify:

-Time Complexity: O(log n)

-Justification: At each level of the tree, the number of operations decreases by half, and the tree height is log n.

2-Build Max-Heap:

-Time Complexity: O(n)

-Justification: Although it appears each node requires O(logn) the overall cost is linear because the number of nodes decreases exponentially at lower levels.

3-Heapsort:

-Time Complexity: O(n log n)

-Justification: The main loop runs n times, and each iteration calls Max-Heapify (O(log n)