### Algorithm: Heap Sort

Heap Sort is a comparison-based sorting technique based on a Binary Heap data structure. It is similar to the selection sort where we first find the maximum element and place it at the end. We repeat the same process for the remaining elements.

#### Build a Max Heap:

* Rearrange the array to satisfy the heap property.

#### Extract Elements:

* Swap the root of the heap with the last element and reduce the heap size.
* Heapify the root to maintain the heap property.

#### Implementation:

| **def** heapify(arr: List[int], n: int, i: int):  largest = i  left = 2 \* i + 1  right = 2 \* i + 2  **if** left < n **and** arr[i] < arr[left]:  largest = left  **if** right < n **and** arr[largest] < arr[right]:  largest = right  **if** largest != i:  arr[i], arr[largest] = arr[largest], arr[i]  heapify(arr, n, largest)  **def** heap\_sort(arr: List[int]) -> List[int]:  n = len(arr)  **for** i **in** range(n // 2 - 1, -1, -1):  heapify(arr, n, i)  **for** i **in** range(n - 1, 0, -1):  arr[i], arr[0] = arr[0], arr[i]  heapify(arr, i, 0)  **return** arr  *# Example usage:*  arr = [12, 11, 13, 5, 6, 7]  print(heap\_sort(arr)) *# Output: [5, 6, 7, 11, 12, 13]* |
| --- |

#### Explanation:

Build a Max Heap:

* Rearrange the array to satisfy the heap property.

Extract Elements:

* Swap the root of the heap with the last element and reduce the heap size.

### Heapify the root to maintain the heap property.