**1.10 Given an array of integers nums sort the array in ascending order and return it. You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.**

**Aim:**

To sort a given array of integers in ascending order without using any built-in sorting functions, achieving O(nlog⁡n)*O*(*n*log*n*) time complexity and minimal space usage.

**Algorithm**:

1. Build a max heap from the input array.
2. Swap the root (maximum element) with the last element of the heap.
3. Reduce the heap size by 1 (exclude the last element which is now sorted).
4. Heapify the root element to maintain the max heap property.
5. Repeat steps 2-4 until the heap size reduces to 1.

**Python Code:**

def heapify(arr, n, i):

largest = i

left = 2 \* i + 1

right = 2 \* i + 2

if left < n and arr[left] > arr[largest]:

largest = left

if right < n and arr[right] > arr[largest]:

largest = right

if largest != i:

arr[i], arr[largest] = arr[largest], arr[i]

heapify(arr, n, largest)

def heap\_sort(arr):

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[0], arr[i] = arr[i], arr[0]

heapify(arr, i, 0)

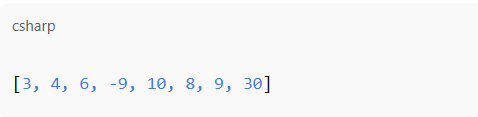
# Example usage:

nums = [3, 4, 6, -9, 10, 8, 9, 30]

heap\_sort(nums)

print("Sorted array:", nums)

**input:**



**Output:**

****

**Result: the given Num sort is successfully executed and output is verified**

**Performance analysis:**

**Time Complexity: O(nlogn)O(n \log n)  
Space Complexity: O(1)O(1) (in-place sorting)**