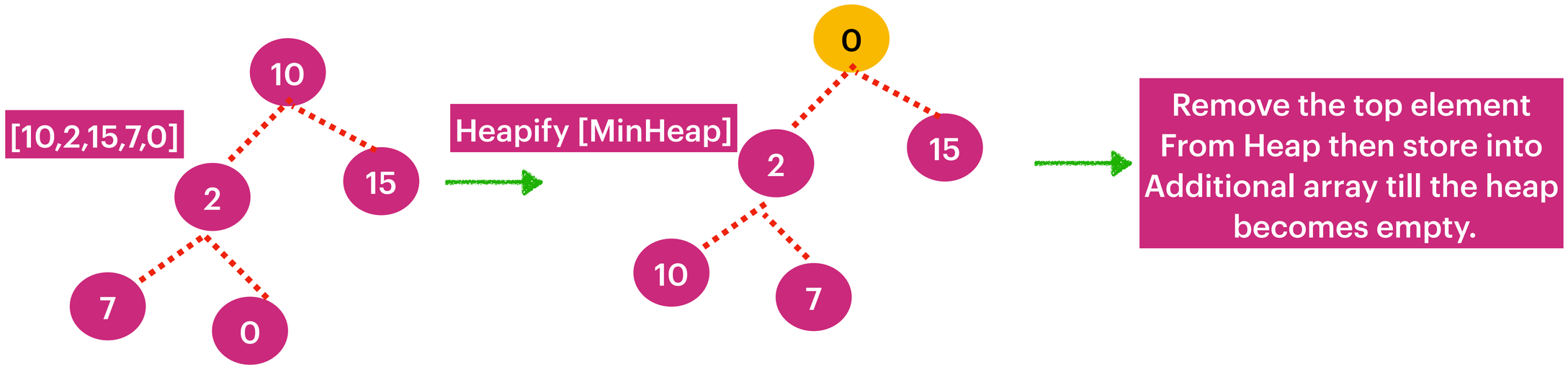
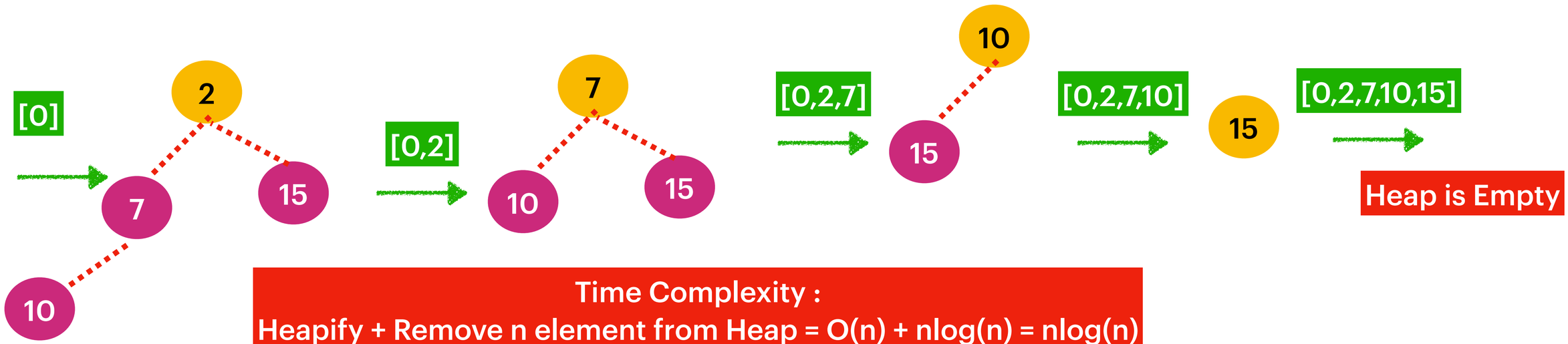
HeapSort





Time Complexity :
Heapify + Remove n element from Heap = O(n) + nlog(n) = nlog(n)
Space Complexity : O(n)

Kth Largest Element in an Array

Given an integer array nums and an integer k, return the kth largest element in the array.
Note that it is the kth largest element in the sorted order, not the kth distinct element.

Ex1 :

Input: nums = [3,2,1,5,6,4], k = 2
Output: 5

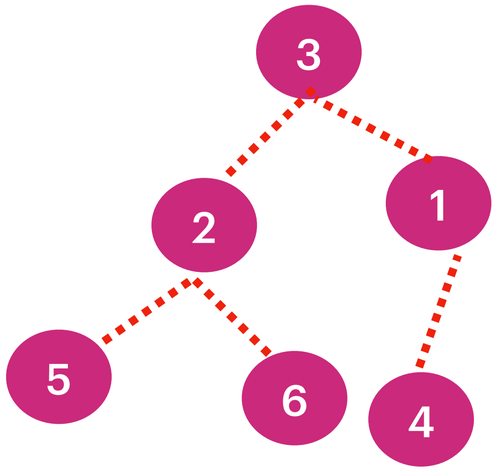
Ex2 :

Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
Output: 4

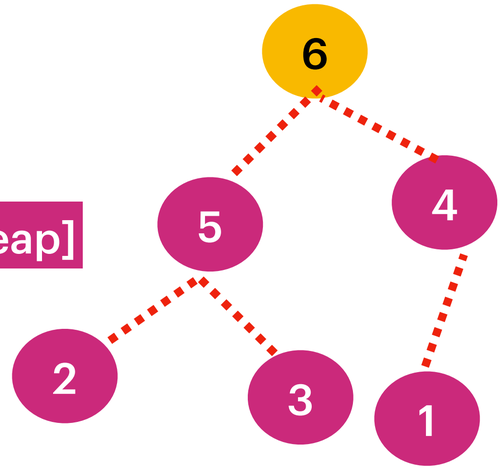
Constraints:

1 <= k <= nums.length <= 104
-104 <= nums[i] <= 104

Input: nums = [3,2,1,5,6,4], k = 2

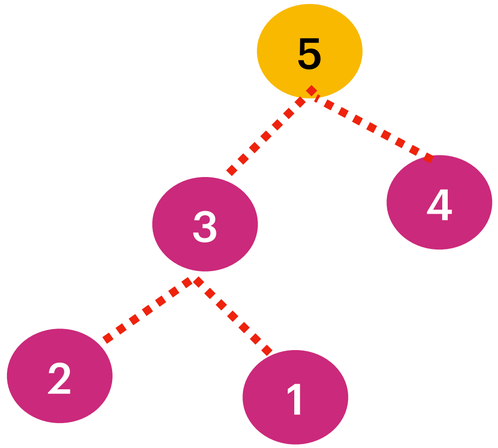


Heapify [MaxHeap]

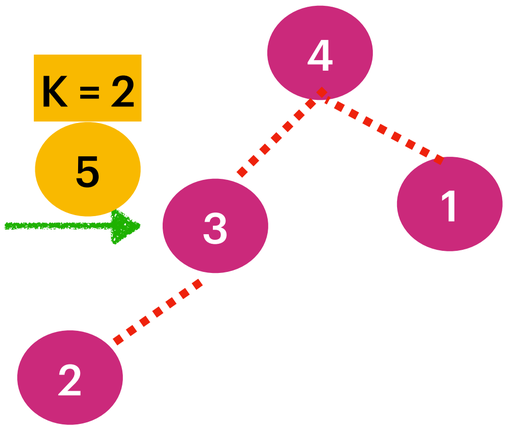


Remove Elements till k, The kth removal one is the kth Largest Element

K = 1



K = 2

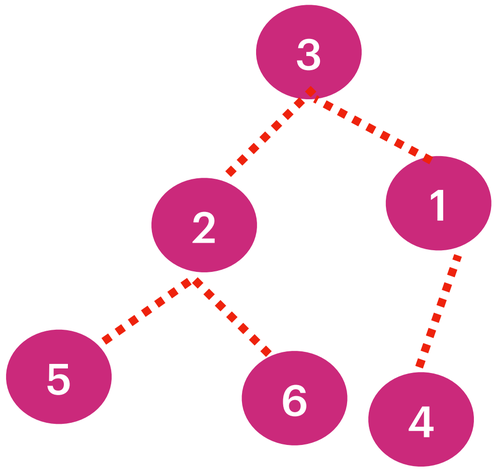


K = 2 , we reached Kth Largest i.e 5

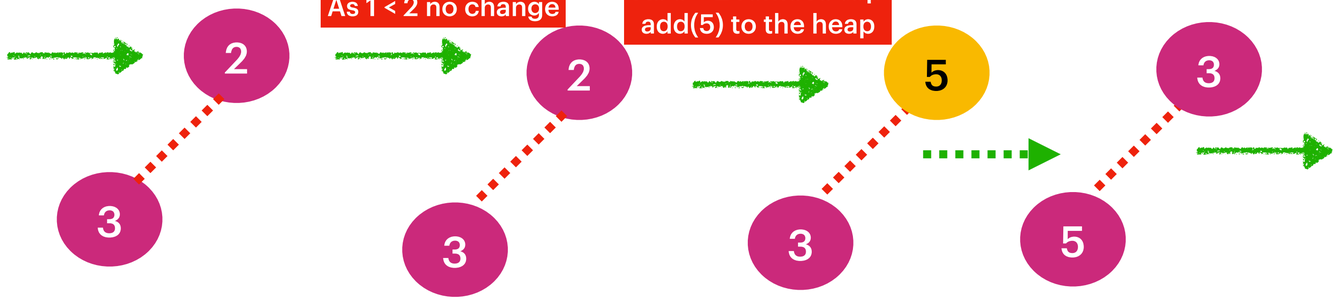
Time Complexity : Heapify + Removed K Elements
O(N) + O(klogn) = O(klogn)
Space Complexity : O(n)

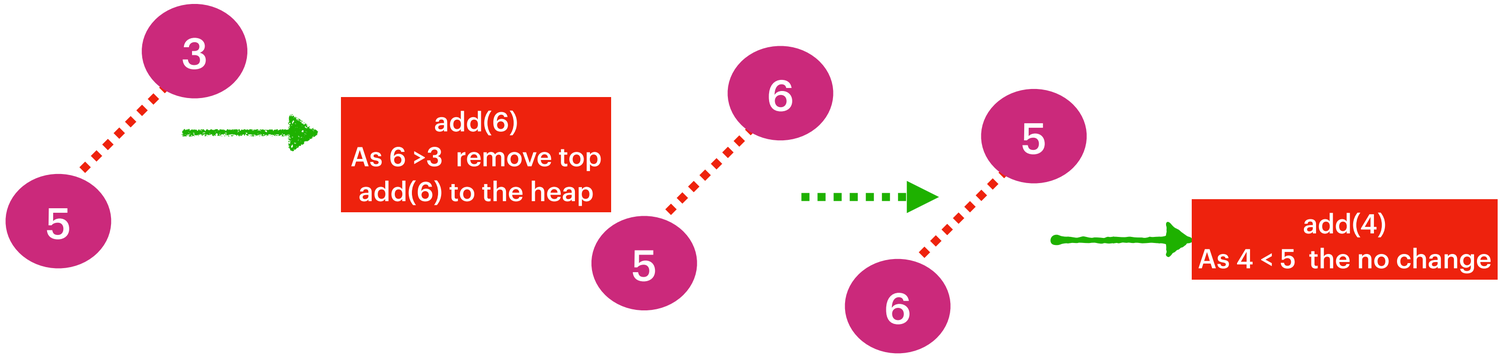
Input: nums = [3,2,1,5,6,4], k = 2

Kth Largest : 5



Construct MinHeap Of size k
HeapSize : 2





Time Complexity : O(nlogk)
Space Complexity : O(k)

We are done with processing then return top element : 5
Which is the Kth is the largest element .

Top K Frequent Elements

Given an integer array nums and an integer k, return the k most frequent elements.
You may return the answer in any order.

Ex1 :

Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]

Ex2 :

Input: nums = [1], k = 1
Output: [1]

Constraints:

1 <= nums.length <= 105
k is in the range [1, the number of unique elements in the array].
It is guaranteed that the answer is unique.

Follow up: Your algorithm's time complexity must be better than O(n log n), where n is the array's size.