**Heaps - Activity**

**PreLab:**

1. Suppose we are sorting an array of eight integers using heapsort, and we have just finished some heapify (either maxheapify or minheapify) operations. The array now looks like this:  
   16 14 15 10 12 27 28  
   How many heapify operations have been performed on root of heap?  
   **(A)** 1  
   **(B)** 2  
   **(C)** 3 or 4  
   **(D)** 5 or 6
2. A 3-ary max heap is like a binary max heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows: The root is stored in the first location, a[0], nodes in the next level, from left to right, is stored from a[1] to a[3]. The nodes from the second level of the tree from left to right are stored from a[4] location onward. An item x can be inserted into a 3-ary heap containing n items by placing x in the location a[n] and pushing it up the tree to satisfy the heap property.Which one of the following is a valid sequence of elements in an array representing 3-ary max heap?  
   **(A)** 1, 3, 5, 6, 8, 9  
   **(B)** 9, 6, 3, 1, 8, 5  
   **(C)** 9, 3, 6, 8, 5, 1  
   **(D)** 9, 5, 6, 8, 3, 1
3. What is the content of the array after two delete operations on the binary heap 25,14,16,13,10,8,12  
   **(A)** 14,13,12,10,8  
   **(B)** 14,12,13,8,10  
   **(C)** 14,13,8,12,10  
   **(D)** 14,13,12,8,10
4. In a binary max heap containing n numbers, the smallest element can be found in time

**(A)** 0(n)  
**(B)** O(logn)  
**(C)** 0(loglogn)  
**(D)** 0(1)

1. The minimum number of interchanges needed to convert the array 89, 19, 40, 17, 12, 10, 2, 5, 7, 11, 6, 9, 70 into a heap with the maximum element at the root is  
   **(A)** 0  
   **(B)** 1  
   **(C)** 2  
   **(D)** 3

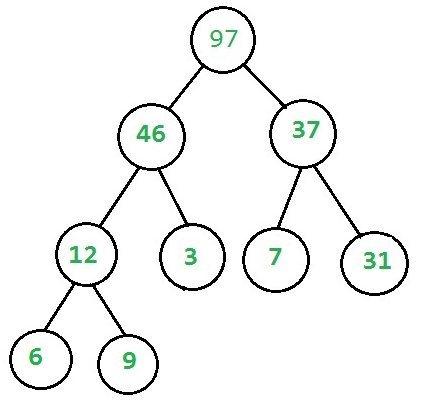
**Program:**

1. **Check if a given Binary Tree is Heap**

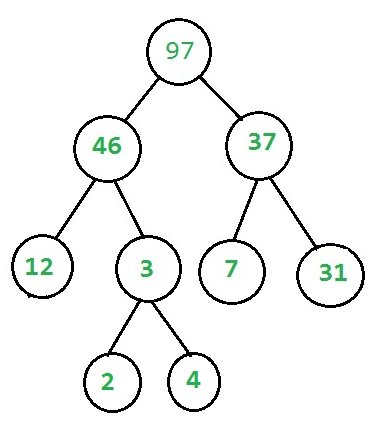
Given a binary tree we need to check it has heap property or not, Binary tree need to fulfill following two conditions for being a heap –

* It should be a complete tree (i.e. all levels except last should be full).
* Every node’s value should be greater than or equal to its child node (considering max-heap).

For example this tree contains heap property –

[](http://d1gjlxt8vb0knt.cloudfront.net/wp-content/uploads/yes.jpg)

While this doesn’t –

[](http://d1gjlxt8vb0knt.cloudfront.net/wp-content/uploads/no.jpg)

1. **Maximum distinct elements after removing k elements**

Given an array **arr[]** containing **n** elements. The problem is to find maximum number of distinct elements after removing **k** elements from the array.  
**Note:** 1 <= k <= n.

Examples:

Input : arr[] = {5, 7, 5, 5, 1, 2, 2}, k = 3

Output : 4

Remove 2 occurrences of element **5** and

1 occurrence of element **2**.

Input : arr[] = {1, 2, 3, 4, 5, 6, 7}, k = 5

Output : 2

**Approach:** Following are the steps:

1. Create a hash table to store the frequency of each element.
2. Insert frequency of each element in a max heap.
3. Now, perform the following operation **k** times. Remove an element from the max heap. Decrement its value by 1. After this if element is not equal to 0, then again push the element in the max heap.
4. After the completion of step 3, the number of elements in the max heap is the required answer.
5. **Convert min Heap to max Heap**

Given array representation of min Heap, convert it to max Heap in O(n) time.

Input: arr[] = [3 5 9 6 8 20 10 12 18 9]

3

/ \

5 9

/ \ / \

6 8 20 10

/ \ /

12 18 9

Output: arr[] = [20 18 10 12 9 9 3 5 6 8]

20

/ \

18 10

/ \ / \

12 9 9 3

/ \ /

5 6 8

The idea is very simple –Build Max Heap without caring about the input. Start from bottom-most and rightmost internal mode of min Heap and heapify all internal modes in bottom up way to build the Max heap.

1. **Implement insertion operation using Max k-ary heap**

K-ary heaps are a generalization of binary heap (K=2) in which each node have K children instead of 2. Just like binary heap, it follows two properties:

1) Nearly complete binary tree, with all levels having maximum number of nodes except the last, which is filled in left to right manner.

2) Like Binary Heap, it can be divided into two categories: (a) Max k-ary heap (key at root is greater than all descendants and same is recursively true for all nodes). (b) Min k-ary heap (key at root is greater than all descendants and same is recursively true for all nodes)

Examples:

3-ary max heap - root node is maximum of all nodes

10

/ | \

7 9 8

/ | \ /

4 6 5 7

**Implementation**

Assuming 0 based indexing of array, an array represents a K-ary heap such that for any node we consider:

* Parent of the node at index i (except root node) is located at index (i-1)/k
* Children of the node at index i are at indices (k\*i)+1 , (k\*i)+2 …. (k\*i)+k
* The last non-leaf node of a heap of size n is located at index (n-1)/k

1. **Write an efficient program for printing k largest elements in an array. Elements in array can be in any order.**

For example, if given array is [1, 23, 12, 9, 30, 2, 50] and you are asked for the largest 3 elements i.e., k = 3 then your program should print 50, 30 and 23.

**Method 4 (Use Max Heap)**  
1) Build a Max Heap tree in O(n)  
2) Use Extract Max k times to get k maximum elements from the Max Heap O(klogn)

**Time complexity:** O(n + klogn)

**Method 5(Use Oder Statistics)**  
1) Use order statistic algorithm to find the kth largest element. Please [see the topic selection in worst-case linear time](https://www.geeksforgeeks.org/kth-smallestlargest-element-unsorted-array-set-3-worst-case-linear-time/) O(n)  
2) Use [QuickSort](https://www.geeksforgeeks.org/quick-sort/) Partition algorithm to partition around the kth largest number O(n).  
3) Sort the k-1 elements (elements greater than the kth largest element) O(kLogk). This step is needed only if sorted output is required.

**Time complexity:** O(n) if we don’t need the sorted output, otherwise O(n+kLogk)

Thanks to Shilpi for suggesting the first two approaches.

**Method 6 (Use Min Heap)**  
This method is mainly an optimization of method 1. Instead of using temp[] array, use Min Heap.

1) Build a Min Heap MH of the first k elements (arr[0] to arr[k-1]) of the given array. O(k)

2) For each element, after the kth element (arr[k] to arr[n-1]), compare it with root of MH.  
……a) If the element is greater than the root then make it root and call [heapify](https://www.geeksforgeeks.org/binary-heap/) for MH  
……b) Else ignore it.  
// The step 2 is O((n-k)\*logk)

3) Finally, MH has k largest elements and root of the MH is the kth largest element.

Time Complexity: O(k + (n-k)Logk) without sorted output. If sorted output is needed then O(k + (n-k)Logk + kLogk)

All of the above methods can also be used to find the kth largest (or smallest) element.

1. **Find k numbers with most occurrences in the given array**

Given an array of n numbers and a positive integer k. The problem is to find k numbers with most occurrences, i.e., the top k numbers having the maximum frequency. If two numbers have same frequency then the larger number should be given preference. The numbers should be displayed in decreasing order of their frequencies. It is assumed that the array consists of k numbers with most occurrences.

Examples:

Input : arr[] = {3, 1, 4, 4, 5, 2, 6, 1},

k = 2

Output : 4 1

Frequency of 4 = 2

Frequency of 1 = 2

These two have the maximum frequency and

4 is larger than 1.

Input : arr[] = {7, 10, 11, 5, 2, 5, 5, 7, 11, 8, 9},

k = 4

Output : 5 11 7 10

1. **Given level order traversal of a Binary Tree, check if the Tree is a Min-Heap**

Given the level order traversal of a Complete Binary Tree, determine whether the Binary Tree is a valid Min-Heap

Examples:

Input : level = [10, 15, 14, 25, 30]

Output : True

The tree of the given level order traversal is

10

/ \

15 14

/ \

25 30

We see that each parent has a value less than

its child, and hence satisfies the min-heap

property

Input : level = [30, 56, 22, 49, 30, 51, 2, 67]

Output : False

The tree of the given level order traversal is

30

/ \

56 22

/ \ / \

49 30 51 2

/

67

We observe that at level 0, 30 > 22, and hence min-heap property is not satisfied