Algorithms
Chapter 5: Transform and conquer

GATE CS PYQ
Solved by Monalisa

- Section 5: Algorithms
- Searching, sorting, hashing. Asymptotic worst case time and space complexity. Algorithm design techniques : greedy, dynamic programming and divide-and-conquer . Graph traversals, minimum spanning trees, shortest paths
- Chapter 1:Algorithim Analysis:-Algorithm intro , Order of growth ,Asymptotic notation, Time complexity, space complexity, Analysis of Recursive & non recursive program, Master theorem]
- Chapter 2:Brute Force:-Sequential search, Selection Sort and Bubble Sort , Radix sort, Depth first Search and Breadth First Search.
- Chapter 3: Decrease and Conquer :- Insertion Sort, Topological sort,Binary Search .
- Chapter 4: Divide and conquer:-Min max problem , matrix multiplication ,Merge sort ,Quick Sort , Binary Tree Traversals and Related Properties .
- Chapter 5: Transform and conquer:- Heaps and Heap sort, Balanced Search Trees.
- Chapter 6: Greedy Method:-knapsack problem , Job Assignment problem, Optimal merge, Hoffman Coding, minimum spanning trees, Dijkstra's Algorithm.
- Chapter 7: Dynamic Programming:-The Bellman-Ford algorithm ,Warshall's and Floyd's Algorithm ,Rod cutting, Matrix-chain multiplication ,Longest common subsequence ,Optimal binary search trees
- Chapter 8: Hashing.
- Reference : Introduction to Algorithms by Thomas H. Cormen
- Introduction to the Design and Analysis of Algorithms, by Anany Levitin
- My Note

- **GATE CS 2009 | Question: 59**
- Consider a binary max-heap implemented using an array.
  Which one of the following array represents a binary max-heap?
- (A) {25,12,16,13,10,8,14}
- (B) {25,14,13,16,10,8,12}
- (C) {25,14,16,13,10,8,12}
- (D) {25,14,12,13,10,8,16}
- Ans : (C) {25,14,16,13,10,8,12}

GATE CS 2009 | Question: 60

Consider a binary max-heap implemented using an array.
What is the content of the array after two delete operations on{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}

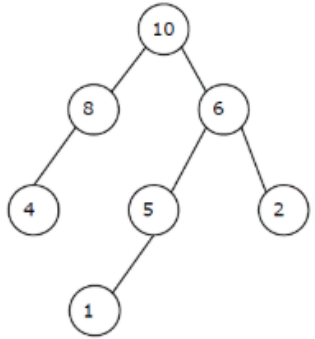
- **for** *i* = 7
- Exchange A[1] with A[7]
- A:*heap-size* = A:*heap-size*-1
- MAX-HEAPIFY(A,1)
- Exchange A[1] with A[6]
- A:*heap-size* = A:*heap-size*-1
- MAX-HEAPIFY(A,1)
- Ans : (D) {14,13,12,8,10}

1	2	3	4	5	6	7
25	14	16	13	10	8	12
12	14	16	13	10	8	**25**
12	14	16	13	10	8	
16	14	**12**	13	10	8	
8	14	12	13	10	**16**	
8	14	12	13	10		
14	13	12	8	10		

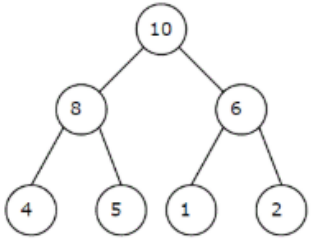
GATE CS 2011 | Question: 23

A max-heap is a heap where the value of each parent is greater than or equal to the value of its children. Which of the following is a max-heap?

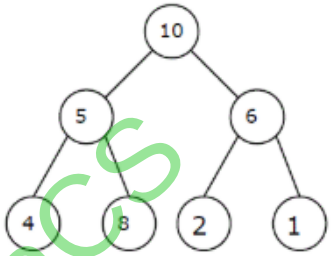
(A)



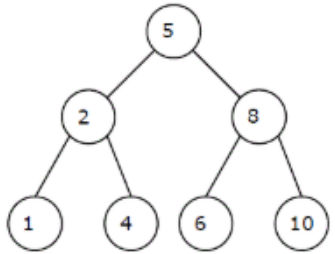
(B)



(C)



(D)



- A binary tree is max-heap if it is a complete binary tree and it follows the max-heap property
- A) is not a max-heap because it is not a complete binary tree
- B) is a max-heap because it is complete binary tree and follows max-heap property.
- C &D are complete binary tree but is not following the max heap property i.e. the value of parent node is not always greater than the child nodes.
- C) 5 is parent of 8.violates the max-heap property.
- D) 5 is parent of 8, 2 is parent of 4,8 is parent of 10.
- Violates the max-heap property.
- Ans : (B)

- **GATE CS 2013 | Question: 7**
- Which one of the following is the tightest upper bound that represents the time complexity of inserting an object into a binary search tree of *n* nodes?
- (A)O(1) (B)O(log n) (C)O(n) (D)O(nlogn)
- To insert an element, first we need to find location for it.
- The search operation can take O(n) for a left skewed BST or a right skewed BST.
- The time complexity of inserting an object into a BST of n nodes is O(n).
- Ans : (C)O(n)

- GATE CS 2013 | Question: 30
- The number of elements that can be sorted in $\Theta(\log n)$ time using heap sort is
- (A) $\Theta(1)$ (B) $\Theta(\sqrt{\log n})$ (C) $\Theta(\frac{\log n}{\log \log n})$ (D) $\Theta(\log n)$
- The HEAPSORT procedure takes time $\Theta(x \log x)$, if x is number of elements.
- If running time = $\Theta(\log n)$, x=?.
- (A) $\Theta(1 \log 1) \neq \Theta(\log n)$
- (B) $\Theta(\sqrt{\log n} \log \sqrt{\log n}) \neq \Theta(\log n)$
- (C) $\Theta(\frac{\log n}{\log \log n} \log \frac{\log n}{\log \log n})$
- $= \Theta(\frac{\log n}{\log \log n} \log(\log n - \log \log n)) = \Theta(\frac{\log n}{\log \log n} (\log \log n - \log \log \log n))$
- $= \Theta(\frac{\log n}{\log \log n} \times \log \log n - \frac{\log n}{\log \log n} \times \log \log \log n)$
- $= \Theta(\log n - \frac{\log n \, log \, log \, log n}{\log \log n}) \cong \Theta(\log n)$
- (D) $\Theta(\log n \log \log n) \neq \Theta(\log n)$
- Ans : (C) $\Theta(\frac{\log n}{\log \log n})$

GATE CS 2014 Set 2 | Question: 12

A priority queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is: 10, 8, 5, 3, 2. Two new elements 1 and 7 are inserted into the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

(A) 10, 8, 7, 3, 2, 1, 5

(B) 10, 8, 7, 2, 3, 1, 5

(C) 10, 8, 7, 1, 2, 3, 5

(D) 10, 8, 7, 5, 3, 2, 1

After insertion of 1

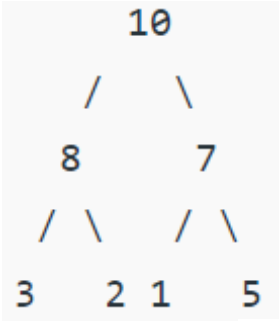
No need to heapify as 5 is greater than 1.

After insertion of 7

Max-Heapify(5)

The level order traversal in this max heap is:
10, 8, 7, 3, 2, 1, 5.

Ans : **(A)** 10, 8, 7, 3, 2, 1, 5



GATE CS 2014 Set 3 | Question: 39

Suppose we have a balanced binary search tree ***T*** holding *n* numbers. We are given two numbers **L** and **H** and wish to sum up all the numbers in ***T*** that lie between **L** and **H**. Suppose there are m such numbers in **T**. If the tightest upper bound on the time to compute the sum is $O(n^a \log^b n + m^c \log^d n)$, the value of a+10b+100c+1000d is _____.

First, you'll have to check if the elements L and H are present in the BST or not.

If L,H are present ,It will take 2log(n) time $\equiv \Theta(\log n)$

Now there are m elements between L and H. So to sum up m element it will take $\Theta(m)$ time.

So total running time $=\Theta(\log n+m)$

$\Longrightarrow a=0,b=1,c=1,d=0$

$\therefore 0+(10\times1)+(100\times1)+(1000\times0)=110.$

Ans :110

- **GATE CS 2015 Set 1 | Question: 23**
- What are the worst-case complexities of insertion and deletion of a key in a binary search tree?
- (A) $\Theta(\log n)$ for both insertion and deletion
- (B) $\Theta(n)$ for both insertion and deletion
- (C) $\Theta(n)$ for insertion and $\Theta(\log n)$ for deletion
- (D) $\Theta(\log n)$ for insertion and $\Theta(n)$ for deletion
- The time complexities for search, insert and delete on a BST is always proportional to height of BST. Height may become O(n) in worst case.
- Worst case BST are **left skewed BST** and **right skewed BST.**
- worst-case complexities of insertion and deletion of a key in a binary search tree $\Theta(n)$.
- Ans: (B) $\Theta(n)$ for both insertion and deletion

GATE CS 2015 Set 1 | Question: 32

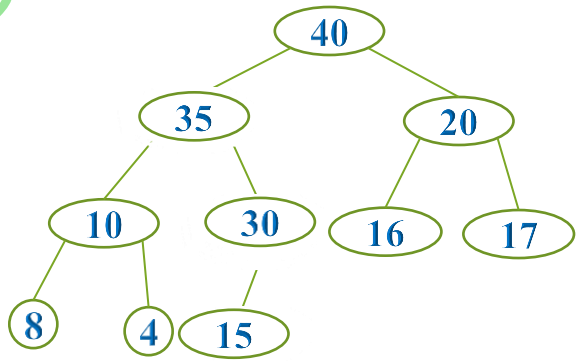
Consider a max heap, represented by the array: 40, 30, 20, 10, 15, 16, 17, 8, 4.

Array index	1	2	3	4	5	6	7	8	9
Value	40	30	20	10	15	16	17	8	4

Now consider that a value 35 is inserted into this heap. After insertion, the new heap is

- **(A)** 40, 30, 20, 10, 15, 16, 17, 8, 4, 35
- **(B)** 40, 35, 20, 10, 30, 16, 17, 8, 4, 15
- **(C)** 40, 30, 20, 10, 35, 16, 17, 8, 4, 15
- **(D)** 40, 35, 20, 10, 15, 16, 17, 8, 4, 30

Ans: **(B)** 40, 35, 20, 10, 30, 16, 17, 8, 4, 15



- **GATE CS 2015 Set 2 | Question: 17**
- Consider a complete binary tree where the left and the right subtrees of the root are max-heaps. The lower bound for the number of operations to convert the tree to a heap is
- (A) $\Omega(\log n)$ (B) $\Omega(n)$ (C) $\Omega(n \log n)$ (D) $\Omega(n^2)$
- Since left and right subtree are already a heap. So we can apply Heapify (node) which take *log n* time .
- Max-Heapify(root).
- Ans :(A) $\Omega(\log n)$

GATE CS 2015 Set 3 | Question: 19

Consider the following array of elements. ⟨89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100⟩. The minimum number of interchanges needed to convert it into a max-heap is

(A) 4 **(B)** 5 **(C)** 2 **(D)** 3

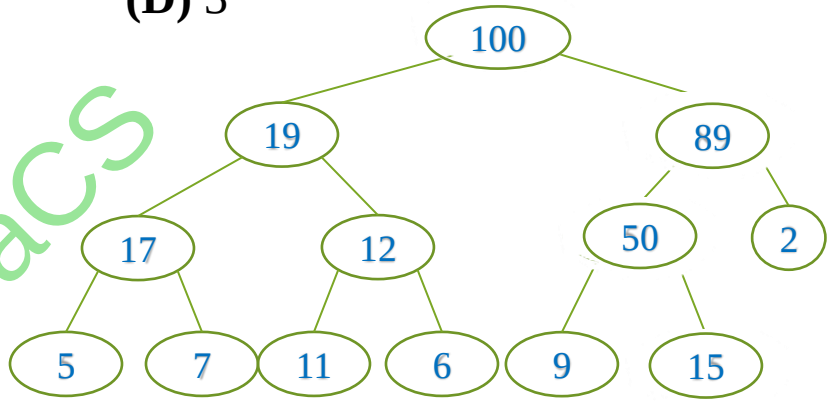
1.Exchange 15 and 100

2.Exchange 50 and 100

3.Exchange 89 and 100

BUILD-MAX-HEAP(*A*)

1. A.*heap-size* = A.*length*
2. **for** *i* = ⌊A.*length*/2⌋ **downto** 1
3. MAX-HEAPIFY(A,*i*)



1	2	3	4	5	6	7	8	9	10	11	12	13
89	19	50	17	12	15	2	5	7	11	6	9	100

89	19	50	17	12	**100**	2	5	7	11	6	9	**15**
89	19	**100**	17	12	**50**	2	5	7	11	6	9	**15**
100	19	**89**	17	12	**50**	2	5	7	11	6	9	**15**

Ans : **(D)** 3

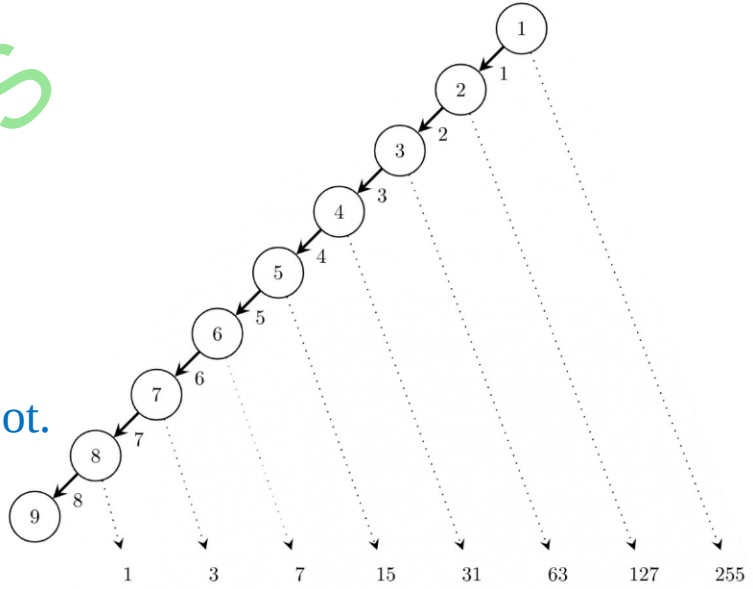
- **GATE CS 2016 Set 1 | Question: 37**
- An operator *delete(i)* for a binary heap data structure is to be designed to delete the item in the *i*-th node. Assume that the heap is implemented in an array and *i* refers to the *i*-th index of the array. If the heap tree has depth *d* (number of edges on the path from the root to the farthest leaf), then what is the time complexity to re-fix the heap efficiently after the removal of the element?
- **(A)** O(1) **(B)** O(d) but not O(1) **(C)** O(2^d) but not O(d) **(D)** O($d2^d$) but not O(2^d)
- *delete(i)* will delete i^{th} index element in O(1) time.
- The efficiency of deletion is determined by the number of key comparisons needed to "heapify" the tree after deletion .
- Heapify =O(depth of heap tree)=O(d)
- *delete(i)* =O(1)+O(d)=O(d)
- Ans : **(B)** O(d) but not O(1)

GATE CS 2016 Set 2 | Question: 34

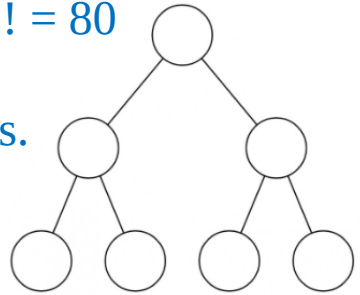
A complete binary min-heap is made by including each integer in [1, 1023] exactly once. The depth of a node in the heap is the length of the path from the root of the heap to that node. Thus, the root is at depth 0. The maximum depth at which integer 9 can appear is

_____

- Node with integer 1 is root only. level 1.
- Node 2 at level 2 as child of node 1.
- Node 3 at level 3 as child of node 2.
- and so for nodes 4,5,6,7,8
- Node 9 at level 9 as child of node 8.
- So total levels are 9.
- Node 9 is at level 9 and depth of node 9 is 8 from the root.
- Ans : 8



- **GATE CS 2018 | Question: 46**
- The number of possible min-heaps containing each value from {1, 2, 3, 4, 5, 6, 7} exactly once is _____.
- We have 7 distinct integers {1,2,3,4,5,6,7} ,pick the minimum element and make it the root of the min heap.
- So, there is only 1 way to make the root of the min heap.
- Now we are left with 6 elements.
- C(6,3)∗2! : Pick up any 3 elements for the left subtree and each left subtree combination can be permuted in 2! ways by interchanging the children and similarly, for right subtree .
- Total ways to design a min heap from 6 elements = C(6,3) ∗ 2! ∗ C(3,3) ∗ 2! = 80
- Or,
- Binary heap is a complete binary tree, so with 7 nodes, we can have 3 levels.
- 1 will be the root.1 possibility.
- 2 will be at 2^{nd} level. 2 possibility .let left side.
- Now at 3^{rd} level 1^{st} position 5 possibility .2^{nd} position 4 possibility .5*4
- From rest 3 minimum will be parent .other 2 will be leaf .2 possibility .
- =5*4*2*2=80
- Ans :80



- **GATE CS 2019 | Question: 40**
- Consider the following statements:

I. The smallest element in a max-heap is always at a leaf node

II. The second largest element in a max-heap is always a child of a root node

III. A max-heap can be constructed from a binary search tree in Θ(n) time

IV. A binary search tree can be constructed from a max-heap in Θ(n) time

- Which of the above statements are TRUE?
- (A) I, II and III (B)I, II and IV (C)I, III and IV (D)II, III and IV
- I. In a max heap, the smallest element is always present at a leaf node. So we need to check for all leaf nodes for the minimum value. Worst case complexity will be O(n).True
- II. The second largest element in a max-heap is always a child of a root node .True
- III. A max-heap can be constructed from a binary search tree in Θ(n) time. Build heap for any array of size n is Θ(n) time . True
- IV. A binary search tree can be constructed from a max-heap in $\Omega(n \log n)$ time . False
- Ans : (A) I, II and III

- **GATE CS 2020 | Question: 6**
- What is the worst case time complexity of inserting n^2 elements into an AVL-tree with n elements initially?
- (A) $\Theta(n^4)$ (B) $\Theta(n^2)$ (C) $\Theta(n^2 \log n)$ (D) $\Theta(n^3)$
- AVL Tree all operations(insert, delete and search) will take $\Theta(\log n)$ time.
- For one element $\Theta(\log n)$
- Inserting n^2 elements. In worst case it will take $\Theta(n^2 \log n)$ time.
- Ans: (C) $\Theta(n^2 \log n)$

GATE CS 2020 | Question: 41

In a balanced binary search tree with n elements, what is the worst case time complexity of reporting all elements in range [*a*,*b*]? Assume that the number of reported elements is *k*.

A. Θ(log *n*) B. Θ(log *n*+*k*) C. Θ(*k*log*n*) D. Θ(*n*log*k*)

Let n=7,a=3,b=8,k=5

First, you'll have to check if the elements a and b are present in the BST or not.

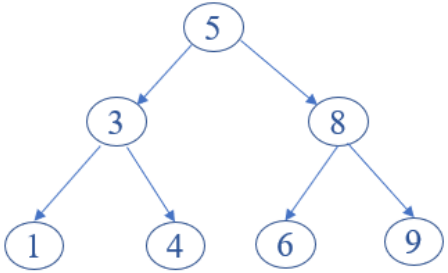
a(3),b(8) are present ,It will take 2log(n) time =Θ(log n)

Traversing the k elements would take additional Θ(k) time.

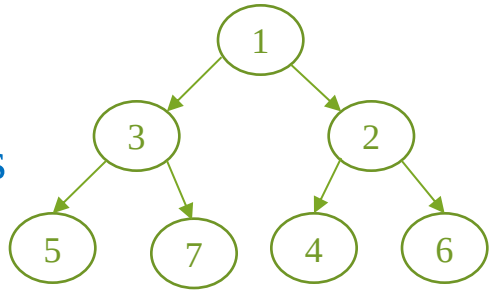
It will just traverse inorder and will get k element in range[a,b]

So total running time =Θ(log n+k)

Ans : (B) Θ(log n+k)



- **GATE CS 2020 | Question: 47**
- Consider the array representation of a binary min-heap containing 1023 elements. The minimum number of comparisons required to find the maximum in the heap is _____.
- In a Min-heap having n elements, there are $\lceil n/2 \rceil$ leaf nodes.
- So, there will be $\lceil 1023/2 \rceil = \lceil 511.5 \rceil = 512$ elements in leaf nodes.
- To find maximum of $\lceil n/2 \rceil$ elements you need ($\lceil n/2 \rceil$ -1) comparisons.
- 512 – 1=511 minimum number of comparisons required to find the maximum
- Ans : 511
- In example from 7 nodes 4 nodes are leaf node
- To find maximum of 4 we need 3 comparisons

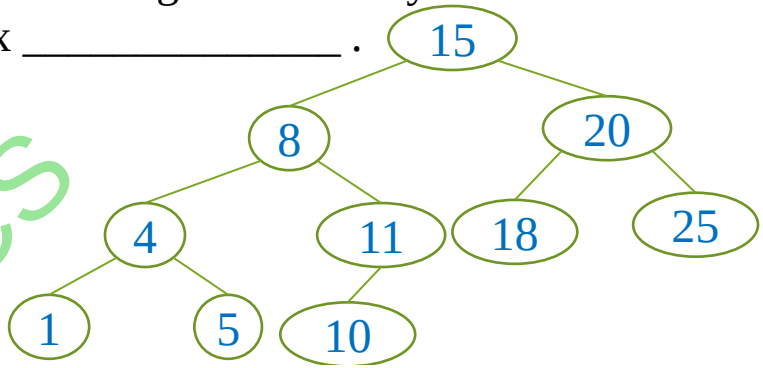


- **GATE CS 2021 Set 2 | Question: 2**
- Let H be a binary min-heap consisting of n elements implemented as an array. What is the worst case time complexity of an optimal algorithm to find the maximum element in H?
  (A) Θ(1) **(B)** Θ(logn) **(C)** Θ(n) **(D)** Θ(nlogn)
- In max heap maximum element present at root while in a min heap, maximum element is present in one of the leaf nodes.
- In a Min-heap having n elements, there are $\lceil n/2 \rceil$ leaf nodes.
- We can't perform binary search here because its not BST and heaps need not be in sorted order.
- So we need to perform linear search on leaves($\lceil n/2 \rceil$ elements) to find maximum element.
- To find maximum of $\lceil n/2 \rceil$ elements you need ($\lceil n/2 \rceil$ -1) comparisons.
- In linear search $\lceil n/2 \rceil$-1 comparisons so time complexity Θ(n)
- Ans: **(C)** Θ(n)

- **GATE CS 2021 Set 1 | Question: 10**
- A binary search tree T contains n distinct elements. What is the time complexity of picking an element in T that is smaller than the maximum element in T?
- (A)$\Theta(n \log n)$ (B)$\Theta(n)$ (C)$\Theta(\log n)$ (D)$\Theta(1)$
- For picking an element which is Not maximum (smaller than maximum) time complexity will be $\Theta(1)$
- As we only need to compare any two elements.
- Pick any two elements at constant time, compare them and return smaller of those two elements.
- Ans : (D)$\Theta(1)$

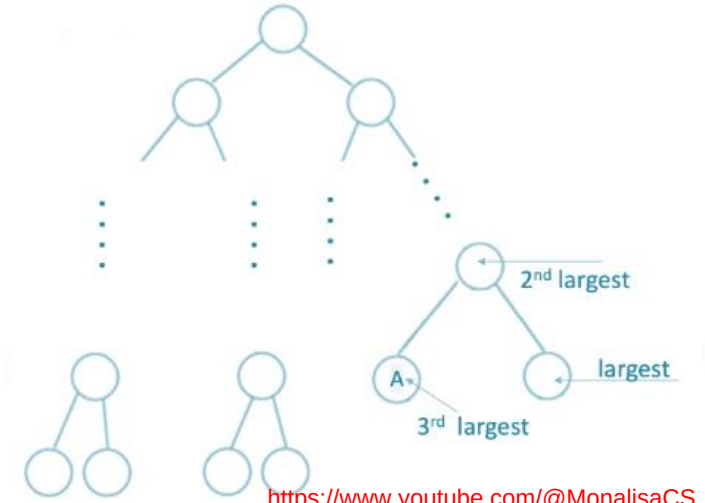
GATE CS 2022 | Question: 18

- Suppose a binary search tree with 1000 distinct elements is also a complete binary tree. The tree is stored using the array representation of binary heap trees. Assuming that the array indices start with 0, the 3rd largest element of the tree is stored at index _____ .
- Lets take 10 elements {1,4,5,8,10,11,15,18,20,25}
- Height=$\lfloor \log_2 n \rfloor = \lfloor \log_2 10 \rfloor = 3$
- Array representation will be in level order like heap tree



15	8	20	4	11	18	25	1	5	10
0	1	2	3	4	5	6	7	8	9

- 3rd largest element of the tree is stored at index 5
- Height =$\lfloor \log_2 1000 \rfloor = 9$
- For 1000 elements, there will be 10 levels (root is at level 1). But last level will not be completely filled.
- 2^{10}-1=1023 elements required to completely filled.
- 2^9 -1=511 elements can be present till height 9.
- As index start from 0
- So largest element will stored at index 510.
- 3rd largest element will be stored at index 509.
- Ans: 509



- **GATE CS 2023 | Question: 2**
- Which one of the following sequences when stored in an array at locations A[1],…,A[10] forms a max-heap?
- (A)23,17,10,6,13,14,1,5,7,12 (B)23,17,14,7,13,10,1,5,6,12
- (C)23,17,14,6,13,10,1,5,7,15 (D)23,14,17,1,10,13,16,12,7,5
- LEFT (*i*) **return** *2i*
- RIGHT (*i*) **return** *2i* +1

	1	2	3	4	5	6	7	8	9	10	
(A)	23	17	10	6	13	14	1	5	7	12	Not max heap 14>10,7>6
(B)	23	17	14	7	13	10	1	5	6	12	Max heap
(C)	23	17	14	6	13	10	1	5	7	15	Not max heap 7>6,15>13
(D)	23	14	17	1	10	13	16	12	7	5	Not max heap 12,7>1

- **Ans : (B)**

- **GATE CS 2023 | Question: 36**
- Let A be a priority queue for maintaining a set of elements. Suppose A is implemented using a max-heap data structure. The operation **Extract-Max(A)** extracts and deletes the maximum element from A. The operation **Insert(A,key)** inserts a new element *key* in A. The properties of a max-heap are preserved at the end of each of these operations.
  When *A* contains *n* elements, which one of the following statements about the worst case running time of these two operations is TRUE?
- (A) Both **Extract-Max(A)** and **Insert(A, *key*)** run in O(1).
- (B) Both **Extract-Max(A)** and **Insert(A, *key*)** run in O(log(*n*)).
- (C) **Extract-Max(A)** runs in *O*(1) whereas **Insert(A,*key*)** runs in O(*n*).
- (D) **Extract-Max(A)** runs in *O*(1) whereas **Insert(A,*key*)** runs in O(log(*n*)).
- **Extract-Max(A)** extracts and deletes the maximum element from A.
- Extract and deletes maximum in O(1).
- MAX-HEAPIFY to maintain Max heap is O(log(n)).
- So **Extract-Max(A)** runs in O(log (n))
- **Insert(A,key)** Insert an element to the last level of max heap O(1)
- MAX-HEAPIFY to maintain Max heap is O(log(n)).
- Ans : (B) Both **Extract-Max(A)** and **Insert(A, *key*)** run in O(log(*n*)).

- **GATE CS 2024 | Set 1 | Question: 31**
- An array [82,101,90,11,111,75,33,131,44,93] is heapified. Which one of the following options represents the first three elements in the heapified array?
- (A)82,90,101 (B)82,11,93 (C)131,11,93 (D)131,111,90
- 1st element is the root =highest element from array =131
- Option A & B are wrong as not starting with 131
- (C) 11 can't be 2nd position
- (D) 131,111,90 are 1st three elements in the heapified array .
- Ans : (D)131,111,90
-

- **GATE CS 2024 | Set 1 | Question: 33**
- Consider a binary min-heap containing 105 distinct elements. Let k be the index (in the underlying array) of the maximum element stored in the heap. The number of possible values of k is (A)53 (B)52 (C)27 (D)1
- In the min heap leaf nodes or last level contains the maximum value of the array.
- Number of leaf node= $\left\lceil \frac{n}{2} \right\rceil = \left\lceil \frac{105}{2} \right\rceil = 53$
- Ans : (A)53

GATE CS 2024 | Set 2 | Question: 29

You are given a set V of distinct integers. A binary search tree T is created by inserting all elements of V one by one, starting with an empty tree. The tree T follows the convention that, at each node, all values stored in the left subtree of the node are smaller than the value stored at the node. You are not aware of the sequence in which these values were inserted into T, and you do not have access to T.Which one of the following statements is TRUE?

(A)Inorder traversal of T can be determined from V

(B)Root node of T can be determined from V

(C)Preorder traversal of T can be determined from V

(D)Postorder traversal of T can be determined from V

In order traversal of BST gives a sorted sequence .

So, if we sort the array that will be our inorder traversal.

As you do not have access to T ,from sorted sequence you can know in order traversal of T.

Ans : (A)Inorder traversal of T can be determined from V