# Heap Tree Construction:

**Inserting key one by one Heapify method**

* Insert key one by one in the given order TC = O(nlogn) no of elements

14 24 12 11 25 8 35

Max Heap / Min Heap

14

Violating MaxHeap property, compare 24 > 14 then swap

24

24

14 12 Insert 12 take O (1) time , then compare 12 with its parent

11 25 25 > 14 , yes , perform swap

24

Still, It is not a MaxHeap , now compare 25 with its parent (24), 25 > 24, so perform swap

25 12

11 14

Now insert elements 8 and 35

25

24 12

11 14 8 35

Still it is not a maxheap, since 35 > 25, so swap 35 with 12

25 35

24 35 24 25

11 14 8 12 11 14 8 12

**Best Case:** In the best case, each time a node is inserted, then max heap property will always be satisfied

Time of inserting a single node is O(1)

Time for inserting ‘n’ nodes in the heap is O (n)

**Showing Worst Case ;**

e.g. : Inserting **45** in already built Heap

**Worst Case :**

No of Comparisons = height of Binary Tree = logn

No of Swaps = height of Binary Tree = logn

Time for inserting a node in worst case=

Insertion time + No. of Comparisons + No. of Swaps

O(1) + log n + log n

=> logn + logn = 2logn

=> O (logn)

Time for inserting ‘n’ nodes in the heap in the worst case: O (nlogn)

45

35

45 24 35 25

45 11 24 14 8 12

45 11

**Heapify Method :**

48 30

3 45 9 18

* Create complete binary tree first
* Heapify method to create Max Heap / Min Heap
* No Swapping required in last level leaves
* last level leaf nodes =Zero swapping
* if there are ‘n’ total elements in tree then n/2 will be leaf nodes
* 15 elements \_\_\_\_ leaves n/2 = 15/2 = ceil (7.5) = 8
* Ignore n/2 leaf nodes ( no swapping required ) ,so time complexity will be effected , we start with rightmost non-leaf node in 2nd last level

145 , 40 , 25 , 65 , 12 , 48 , 18 , 1 , 100 , 27 , 7 , 3 , 45 , 9 , 30

145

40 25

logn

65 12 48 18

1 100 27 7 3 45 9 30

Swap 18 and 30, since 30 > 18 (maxheap property violated)

145

40 25

65 12 48 30

1 100 27 7 3 45 9 18

Swap 12 and 27, Maxheap property violated

145

40 25

65 27 48 30

1 100 12 7 3 45 9 18

Swap 100 and 65, Maxheap property violated

145

40 25

100 12 48 30

1 65 27 7 3 45 9 18

Swap 48 and 25, Maxheap property violated

145

40 48

100 12 25 30

1 65 27 7 3 45 9 18

When 25 moved downward, it disturbed the heap and hence maxheap property violated, since 45> 25

Swap 45 and 25

145

40 48

100 12 45 30

1 65 27 7 3 25 9 18

Swap 100 and 40, Maxheap property violated

**Geometric progression :**

145 GP is a type of sequence where

each succeeding term is produced

100 45 by multiplying each preceding term

by a fixed number

40 12 25 30

**AP series** is a series which has

1 65 27 7 3 25 9 18 consecutive times having a

common diff b/w the turns as

a constant value .

Corresponding to Last node (root node ), the max no of swaps depends on height of tree = logn - so max swaps = logn

n/2logn

n/23

n/22 4 2 swaps

n/21 8 one swap

n nodes = 16

Last Level = n nodes, then second last level will have n/2 nodes

Total swaps = S

S=

Taking "n" as common

S=

Now Multiply equation (1) by

Now subtract eq (2) from eq (1)

Using Geometric Progression formula:

// common ratio = r=

a= first term= =

n is the last power which is logn

putting values in G.P Formula :

Taking log on both sides

Divide both sides by log

So

Multiply both sides by 2