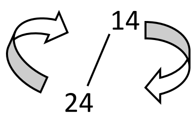
Heap Tree Construction:

Inserting key one by one | **Heapify method**

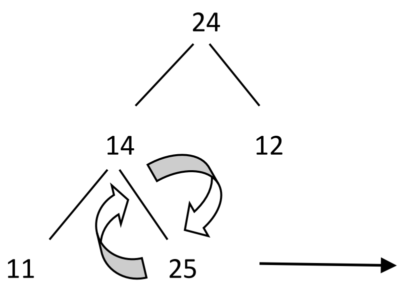
- Insert key one by one in the given order TC = O(nlogn) no of elements

  14 24 12 11 25 8 35

  Max Heap / Min Heap

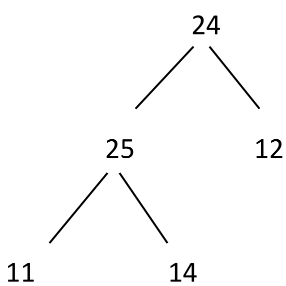


Violating MaxHeap property, compare 24 > 14 then swap



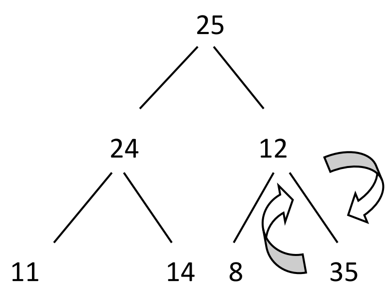
Insert 12 take O (1) time , then compare 12 with its parent

25 > 14 , yes , perform swap

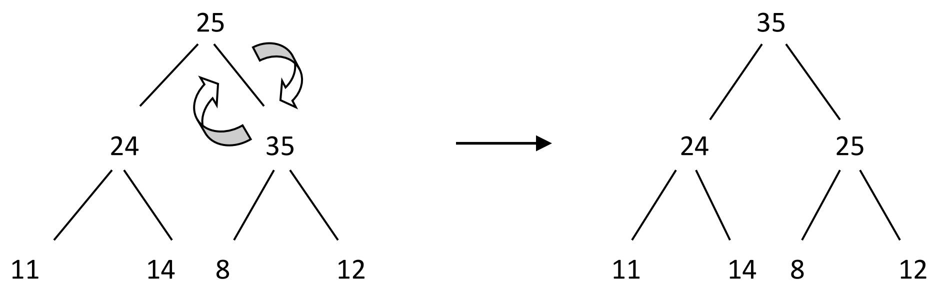


Still, It is not a MaxHeap , now compare 25 with its parent (24), 25 > 24, so perform swap

Now insert elements 8 and 35



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



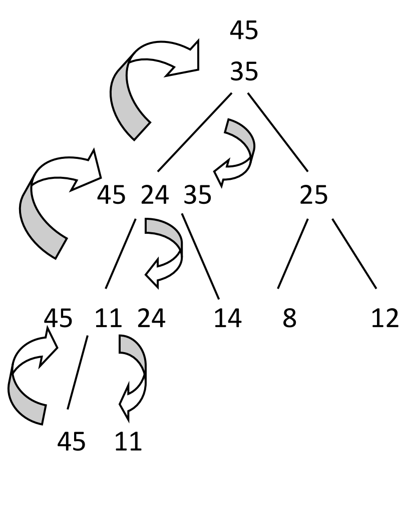
Best Case: In the best case, each time a node is inserted, then max heap property will always be satisfied

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

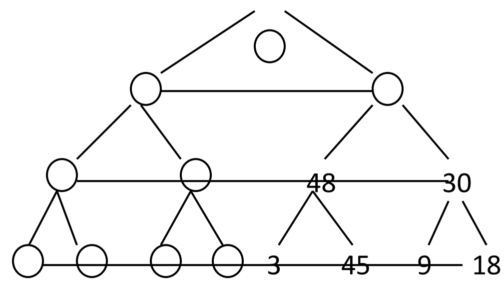
O(1) + log n + log n

=> logn + logn = 2logn

=> O (logn)

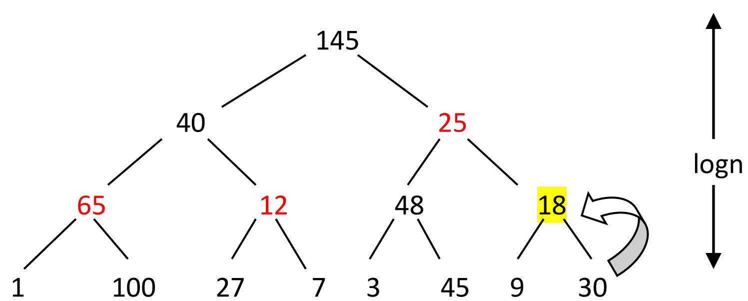
Time for inserting ‘n’ nodes in the heap in the worst case: O (nlogn)

Heapify Method :

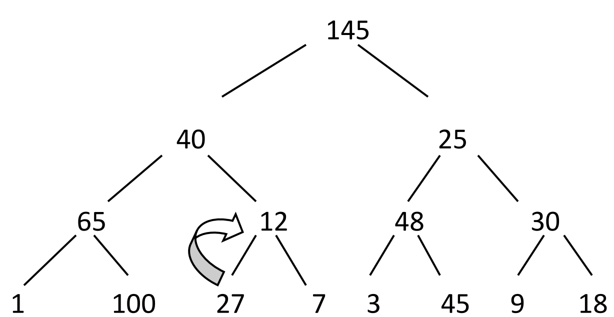


- Create complete binary tree first
- Heapify method to create Max Heap / Min Heap
- No Swapping required in last level leaves
- last level leaf nodes =Zero swapping
- if there are 'n' total elements in tree then n/2 will be leaf nodes
- 15 elements _____ leaves n/2 = 15/2 = ceil (7.5) = 8
- Ignore n/2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

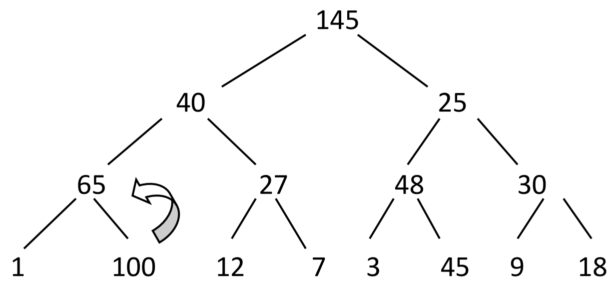
145 , 40 , 25 , 65 , 12 , 48 , 18 , 1 , 100 , 27 , 7 , 3 , 45 , 9 , 30



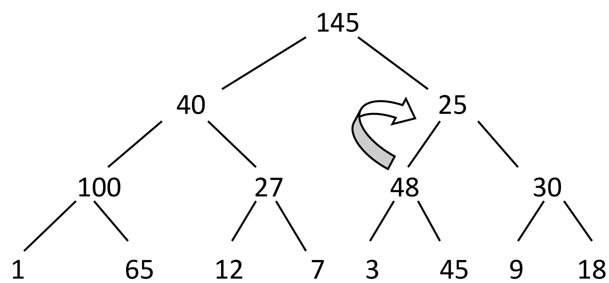
Swap 18 and 30, since 30 > 18 (maxheap property violated)



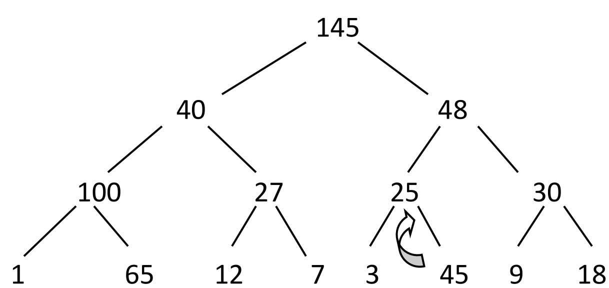
Swap 12 and 27, Maxheap property violated



Swap 100 and 65, Maxheap property violated

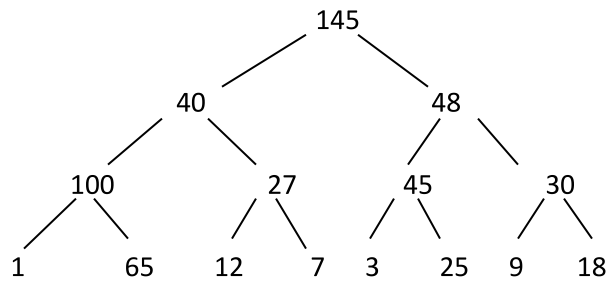


Swap 48 and 25, Maxheap property violated

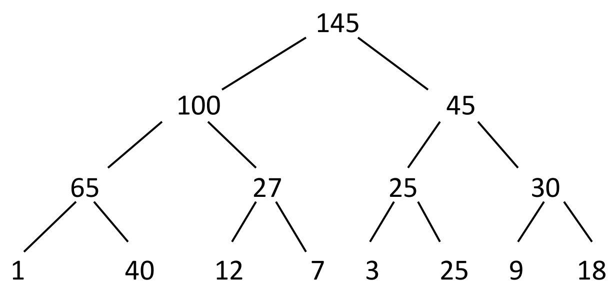


When 25 moved downward, it disturbed the heap and hence maxheap property violated, since 45> 25

Swap 45 and 25



Swap 100 and 40, Maxheap property violated



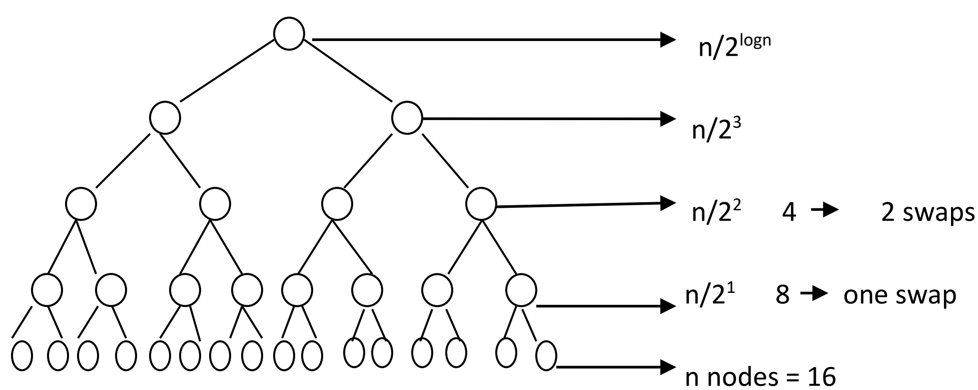
Geometric progression :
GP is a type of sequence where each succeeding term is produced by multiplying each preceding term by a fixed number

AP series is a series which has 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



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

Total swaps = S

S= $[\frac{n}{2^0} * 0 + \frac{n}{2^1} * 1 + \frac{n}{2^2} * 2 + \frac{n}{2^3} * 3 \dots\dots\dots\dots.. \frac{n}{2^{logn}} * logn]$

Taking "n" as common

S= $n\left[0 + \frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} \dots\dots\dots\dots.. \frac{logn}{2^{logn}}\right] - - - - - - - - - - - - - - - (1)$

Now Multiply equation (1) by $\frac{1.}{2}$

$\frac{s}{2} = n\left[\frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} \dots\dots\dots\dots.. \frac{logn-1}{2^{logn}} + \frac{logn}{2^{logn+1}}\right] - - - - - - - - - - - - - - - (2)$

Now subtract eq (2) from eq (1)

$$\frac{s}{2} = n\left[(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} \dots\dots\dots\dots.. \frac{logn - (logn - 1)}{2^{logn}}) - \frac{logn}{2^{logn+1}}\right]$$

$$\frac{s}{2} = n\left[(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} \dots\dots\dots\dots.. \frac{1)}{2^{logn}}) - \frac{logn}{2^{logn+1}}\right]$$

Using Geometric Progression formula:

$s = \frac{a(1- r^n)}{1-r}$

// common ratio = r= $\frac{1}{2}$

a= first term= = $\frac{1}{2}$

n is the last power which is logn

putting values in G.P Formula :

$$\frac{s}{2} = n\left(\frac{\frac{1}{2}\ (1 - (\frac{1}{2})^{logn})}{(1 - \frac{1}{2})} - \frac{logn}{2^{\ (logn+1)}}\right)$$

$$\frac{s}{2} = n\left(\frac{\frac{1}{2}\ (1-\frac{1}{2^{logn}})}{(1-\frac{1}{2})} - \frac{logn}{2^{\ (logn+1)}}\right)$$

$$\frac{s}{2} = n\left(\frac{\cancel{\frac{1}{2}}\ (1-\frac{1}{2^{logn}})}{\cancel{\frac{1}{2}}} - \frac{logn}{2^{\ (logn+1)}}\right)$$

$$\frac{s}{2} = n\left(\frac{2^{logn}-1)}{2^{logn}} - \frac{logn}{2^{\ logn}.2^{\ 1}}\right)$$

$since\ 2^{\ logn} = n$

$Let\ y = \ 2^{\ logn}$

Taking log on both sides

$log\ y = log2^{\ logn}$

$log\ y = \ logn\ log_2{}^2$

$log\ y = \ logn$

Divide both sides by log

$y = \ n$

$and\ \ y = \ 2^{\ logn}$

So
$2^{\ logn} = n$

$$\frac{s}{2} = n\left(\frac{n-1)}{n} - \frac{logn}{2n}\right)$$

$$\frac{s}{2} = n\left(\frac{2(n-1)-logn)}{2n}\right)$$

$$\frac{s}{2} = n\left(\frac{2n - 2 - logn)}{2n}\right)$$

$$\frac{s}{2} = \cancel{n}\left(\frac{2n - 2 - logn)}{2\cancel{n}}\right)$$

$$\frac{s}{2} = \left(\frac{2n - 2 - logn)}{2}\right)$$

Multiply both sides by 2

$$s = 2n - 2 - logn$$

$$s = O(n)$$