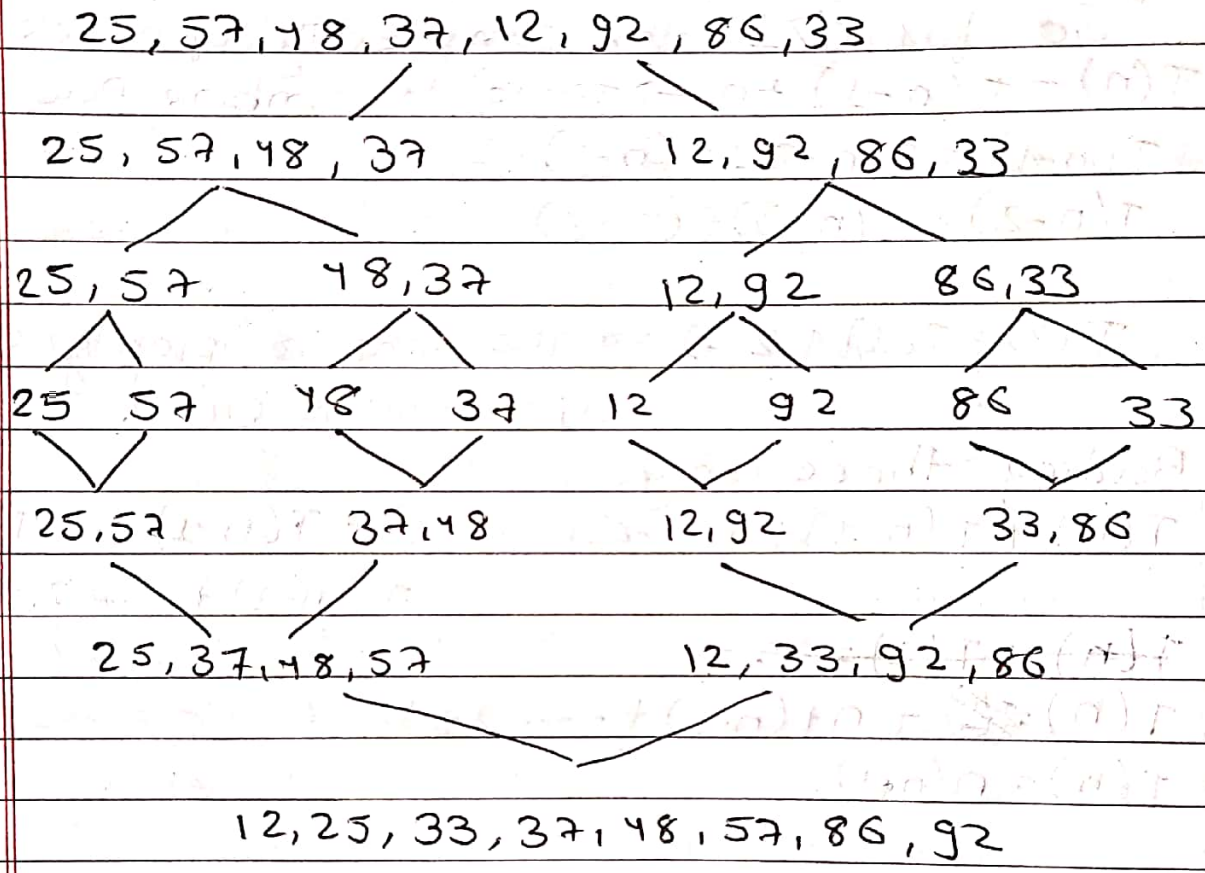
Merge Sort

Steps:-

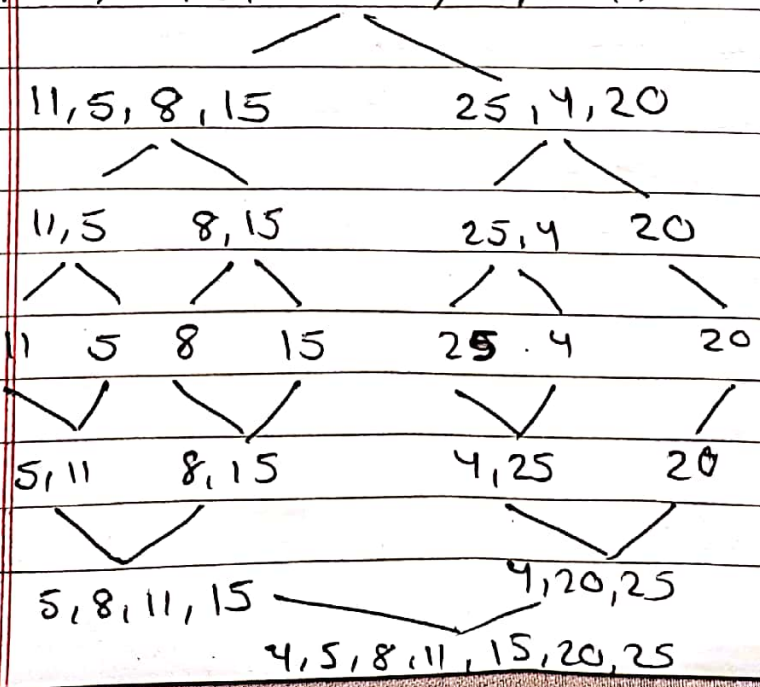
→ Divide the given array in equal halves.

→ Merge the partition subarray recursively

For eg:



Q 11, 5, 8, 15, 25, 4, 20, ~~12~~



Date: / /

Analysis of Merge sort.

The total time complexity for merge sort is $n/2$ for partitioning & linear time for merge ie n.

So, $T(n) = 2T(n/2) + n$

using telescoping method.

$T(n/2) = 2T(n/4) + \frac{n}{2}$

Now substituting the value of $T(n/2)$ in eqn. of $T(n)$ we get,

$$T(n) = 2(2T(n/4) + n/2) + n$$
$$= 4T(n/4) + n + n$$
$$= 4T(n/4) + 2n$$

Similarly, for $T(n/4)$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{n}{4}.$$

now substituting the value of $T(n/4)$ in eqn of $T\left(\frac{n}{2}\right)$

$$4\left[2T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + 2n$$

$$= 8T\left(\frac{n}{8}\right) + n + 2n = 8T\left(\frac{n}{8}\right) + 3n$$

This eqn for some number

$$T(n) = 2^k T(n/2^k) + kn$$

This is valid for number k which is less than the value of n ie we can write $k = \log n$ & $2^k = n$ now

$$= n T(n/n) + \log n \cdot n$$
$$= nT(1) + n\log n = n + n\log n$$

ie O(nlogn).

Heap and Heap Sorting.

A heap is defined as an almost complete binary tree of nodes n such that the value of each node is less than or equal to the value of parent or greater than or equals to its parent.

There are two types of heap, they are

1. descending heap. (min heap)

It is almost complete binary tree in which the value of each node is smaller or equal to the value of its parent

2. ascending heap. (max heap)

It is almost complete binary tree in which the value of each node is greater or equal to the value of its parent.

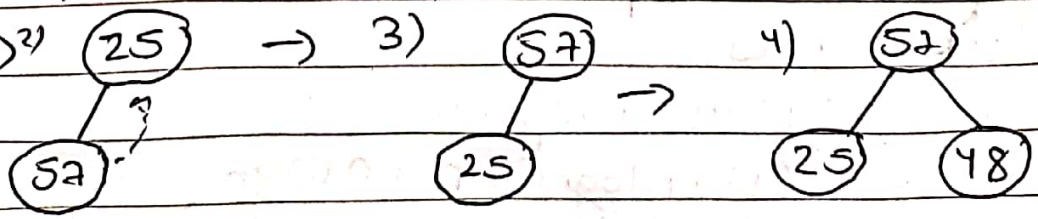
Sorting by using Heap

The heap sorts algorithm sorts by representing its input as a heap in its array. Basically there are two steps / phases in heap sorting ~~there~~ they are ① creating a heap (create a heap using descending ~~order~~)

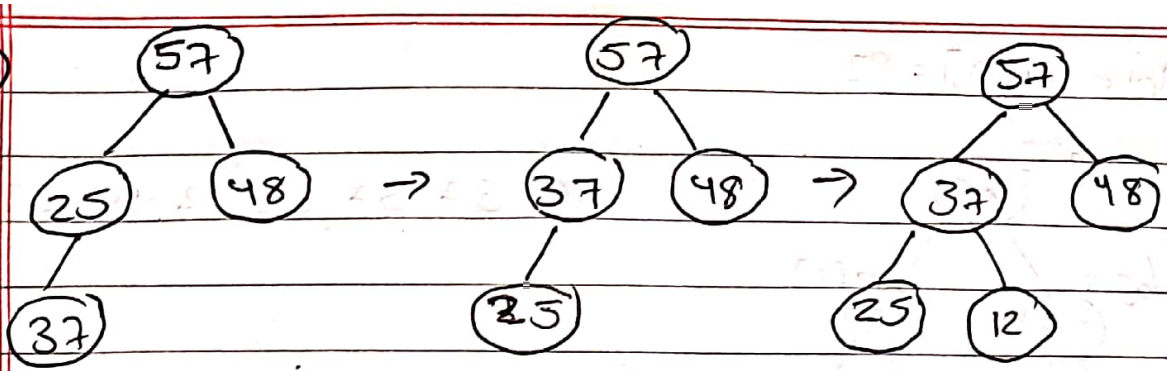
② Asjusting the heap by using heap array (process of deleting elements.

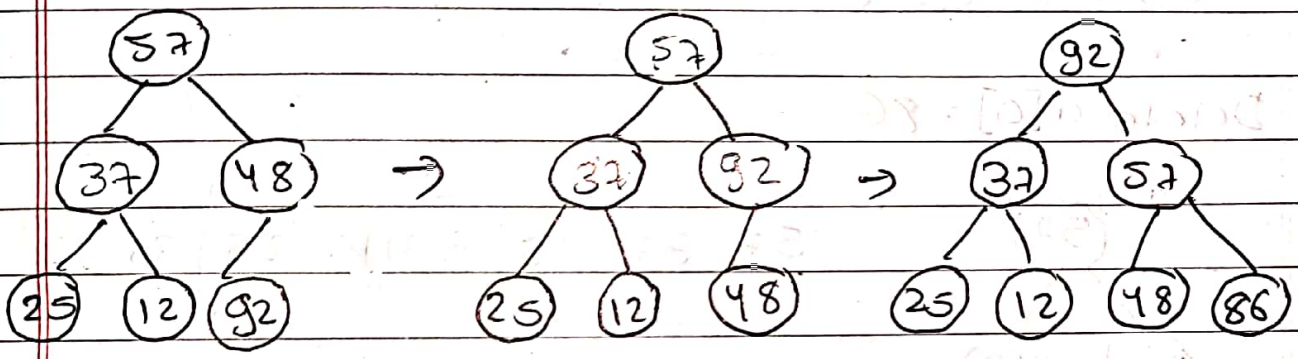
25, 57, 48, 37, 12, 92, 86, 33

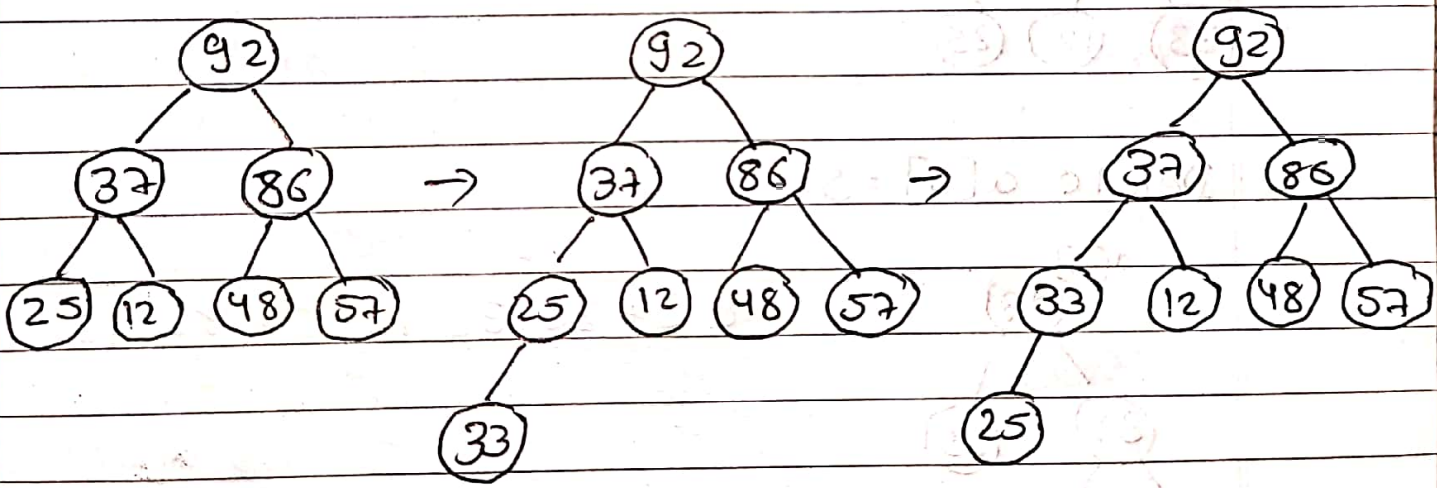
→ 1) 25 → 2) 25 → 3) 57 → 4) 57



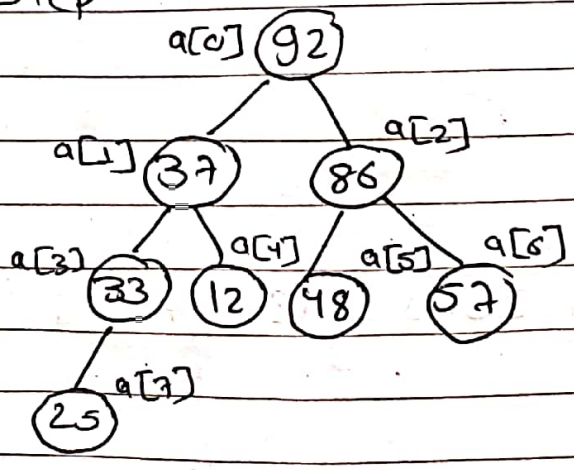
5)





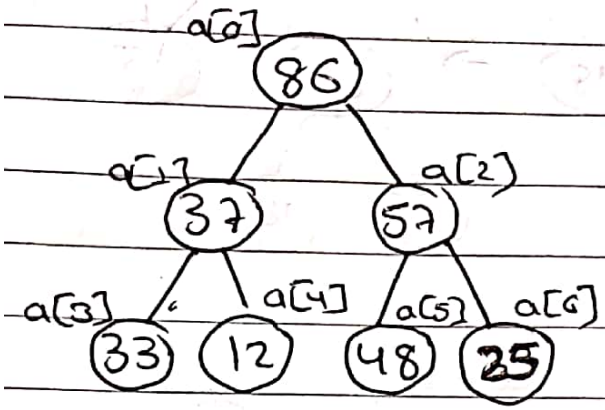


Step 2:



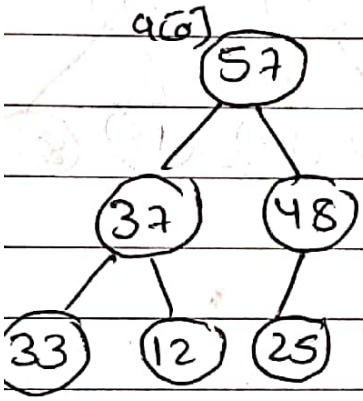
92	37	86	33	12	48	57	25

Delete a[0] = 92.



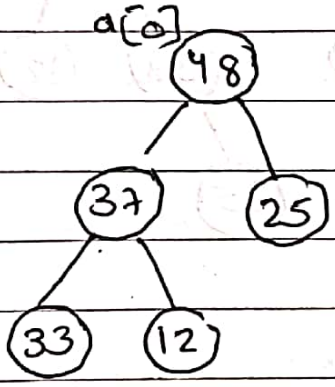
86	37	57	33	12	48	25	92

Delete a[0] = 86



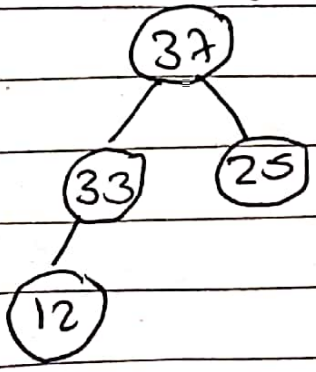
57	37	48	33	12	25	86	92

Delete a[0] = 57



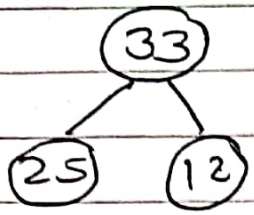
48	37	25	33	12	57	86	92

Delete a[0] = 48



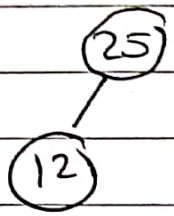
37	33	25	12	48	57	86	92

Delete 37



33	25	12	37	48	57	86	92

Delete a[0] = 33



25	12	33	37	48	57	86	92

Delete a[0] = 25

12	25	33	37	48	57	86	92

Analysis

The time complexity of heap sort in worst case is $O(n \log n)$

Algorithms	Best case	Average case	Worst case
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Heap sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Shell sort	$O(n \log n)$	$O(n \log n^2)$ or $O(n^3 n)$	$O(n^{3/2})$