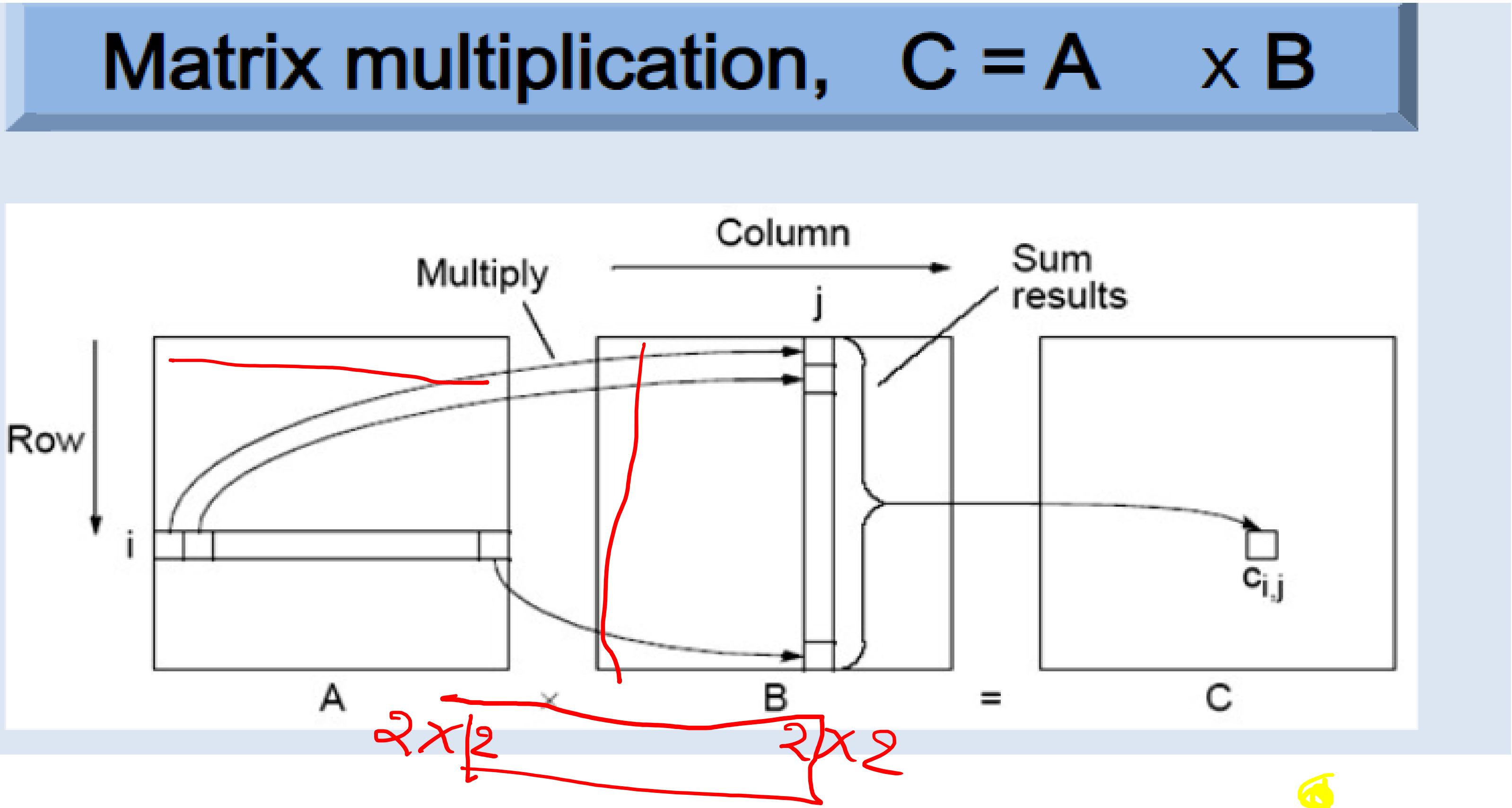
Strassen's Matrix Multiplication



$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

A 2x2 same 2x2

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

2x2

$c_{11} = a_{11}b_{11} + a_{12}b_{21}$

$c_{12} = a_{11}b_{12} + a_{12}b_{22}$

$c_{21} = a_{21}b_{11} + a_{22}b_{21}$

$c_{22} = a_{21}b_{12} + a_{22}b_{22}$

Sequential Code

Assume throughout that the matrices are square (*n* x *n* matrices).
The sequential code to compute **A** x **B** :

n
n
n

```
for (i = 0; i < n; i++)
        for (j = 0; j < n; j++) {
                c[i][j] = 0;
                for (k = 0; k < n; k++)
                        c[i][j] = c[i][j] + a[i][k] * b[k][j];
}
```

$O(n^3)$

Divide & conquer :-

two $n \times n$ matrix where $\boxed{n = 2^k}$ k is positive

$[\ ]_{2\times 2}$, $[\ ]_{4\times 4}$ $[\ ]_{8\times 8}$

$$C = A_{2\times 2} \times B_{2\times 2}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$c_{11} = a_{11} \cdot b_{11} + a_{12} \cdot b_{21}$

$c_{12} = a_{11} \cdot b_{12} + a_{12} b_{22}$

$c_{21} = a_{21} \cdot b_{11} + a_{22} b_{21}$

$c_{22} = a_{21} \cdot b_{12} + a_{22} b_{22}$

8 multiplication

4 addition

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ & A_{11} & & A_{12} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ & A_{21} & & A_{22} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right]_{4\times4}$$

$\frac{4}{2} \times \frac{4}{2}$ $\quad$ $\frac{n}{2} \times \frac{n}{2}$

$$B = \left[\begin{array}{cc|cc} b_{11} & b_{12} & b_{13} & b_{14} \\ & B_{11} & & B_{12} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ \hline b_{31} & b_{32} & b_{33} & b_{34} \\ & B_{21} & & B_{22} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array}\right]_{4\times4}$$

$\frac{4}{2} \times \frac{4}{2}$

$C_{11} = \underset{1}{A_{11} B_{11}} + \underset{2}{A_{12} B_{21}}$

$C_{12} = \underset{3}{A_{11} B_{12}} + \underset{4}{A_{12} B_{22}}$

$C_{21} = \underset{5}{A_{21} B_{11}} + \underset{6}{A_{22} B_{21}}$

$C_{22} = \underset{7}{A_{21} B_{12}} + \underset{8}{A_{22} B_{22}}$

$$T(n) = \begin{cases} 1 & n \le 2 \\ 8T\left(\frac{n}{2}\right) + 4n^2, & n > 2 \end{cases}$$

$\hookrightarrow O(n^3)$ ✓

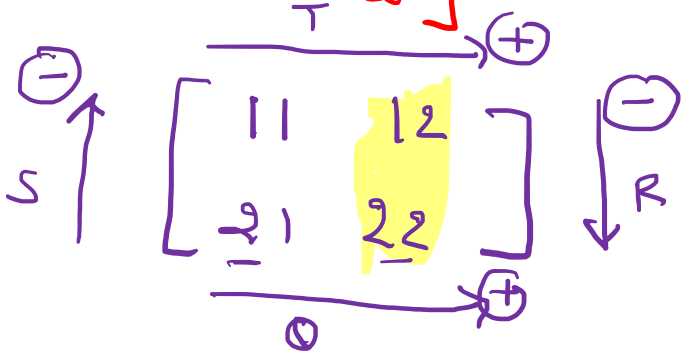
Naive method

Strassen's matrix multiplication (★★★★)

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C = A \times B$$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$



$B_{11}\ A_{11}\ A_{22}\ B_{22}$

$$P = (A_{11}+A_{22})\cdot(B_{11}+B_{22})$$
$$Q = B_{11}\cdot(A_{21}+A_{22})$$
$$R = A_{11}\cdot(B_{12}-B_{22})$$
$$S = A_{22}\cdot(B_{21}-B_{11})$$
$$T = B_{22}\cdot(A_{11}+A_{12})$$
$$U = (A_{21}-A_{11})\cdot(B_{11}+B_{12})$$
$$V = (B_{21}+B_{22})\cdot(A_{12}-A_{22})$$

$\Rightarrow$ 7 multi

10 Add/Sub
+8
18

$\Rightarrow \cancel{8}\ 7$

$$C_{11} = P+S-T+V$$
$$C_{12} = R+T$$
$$C_{21} = Q+S$$
$$C_{22} = P+R-Q+U$$

RAT

$$\Rightarrow T(n) = 7T\left(\frac{n}{2}\right)+18n^2 \Rightarrow O(n^{\log_2 7})$$

$$\Rightarrow O(n^{2.81})$$

$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

T +

$+ S \begin{bmatrix} 11 & 12 \\ 21 & 22 \end{bmatrix} R$

Q

$B_{11} A_{11} A_{22} B_{22}$

Solⁿ:

$P = (1+5)(5+8) =$

$Q = 5(\quad)$

$R = 1(\quad)$

$S = 4(\quad)$

$T = 8(\quad)$

$U =$

$V =$

$C_{11} =$

$C_{12} =$

$C_{21} =$

$C_{22} =$

P: 183, 184, 185, 186, 188, 192, 193, 197, 198, 200, 202, 203, 207, 218, 222, 224, 225, 226, 232, 233, 235, 238, 613, 621, 637, 650

T005

I 482, 483, 484, 485, 492, 496, 500, 501, 504, 508, 511, ~~[illegible]~~, 515, 516, 517, 521, 523, 624, 628, 629,

Heap Sort Algorithm

Top
↓
Bottom

left to right

8

19 21

What is a heap?

A heap is a complete binary tree, and the binary tree is a tree in which the node can have the at most two children. A complete binary tree is a binary tree in which all the levels except the last level, i.e., leaf node, should be completely filled, and all the nodes should be left-justified.

→ $O(n \log n)$

What is heap sort?

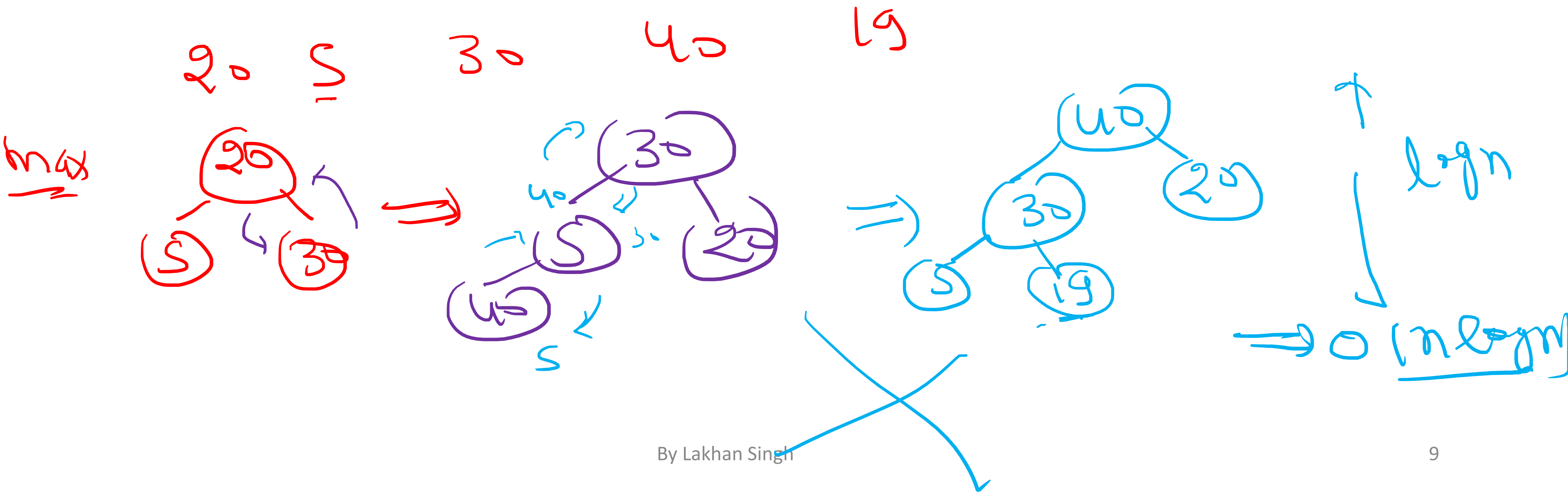
Heapsort is a popular and efficient sorting algorithm. The concept of heap sort is to eliminate the elements one by one from the heap part of the list, and then insert them into the sorted part of the list.

Heapsort is the in-place sorting algorithm.

Step by Step Process

The Heap sort algorithm to arrange a list of elements in ascending order is performed using following steps...

- **Step 1 -** Construct a **Binary Tree** with given list of Elements.
- **Step 2 -** Transform the Binary Tree into **Min Heap.**
- **Step 3 -** Delete the root element from Min Heap using **Heapify** method.
- **Step 4 -** Put the deleted element into the Sorted list.
- **Step 5 -** Repeat the same until Min Heap becomes empty.
- **Step 6 -** Display the sorted list.

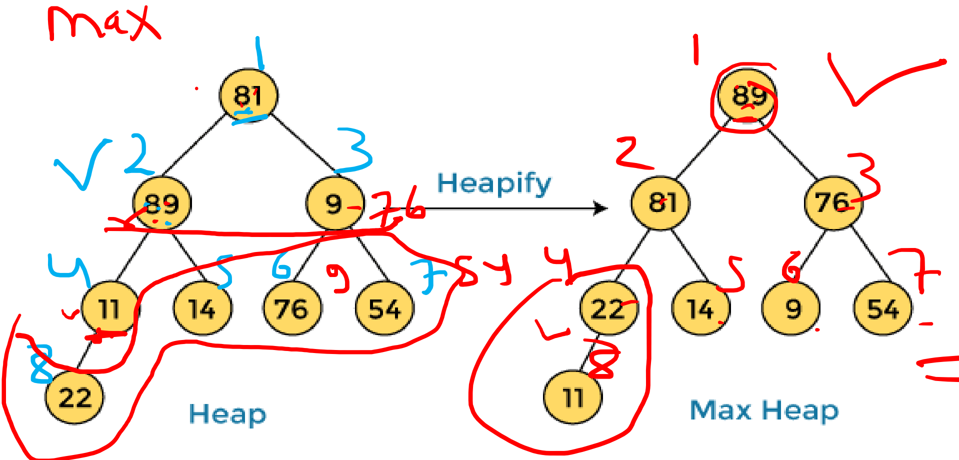


81	89	9	11	14	76	54	22

max heap :- parent ≥ child

min heap : parent ≤ child

max



$\left\lfloor \frac{n}{2} \right\rfloor + 1 \Rightarrow$ Leaf node

$\Rightarrow O(n)$

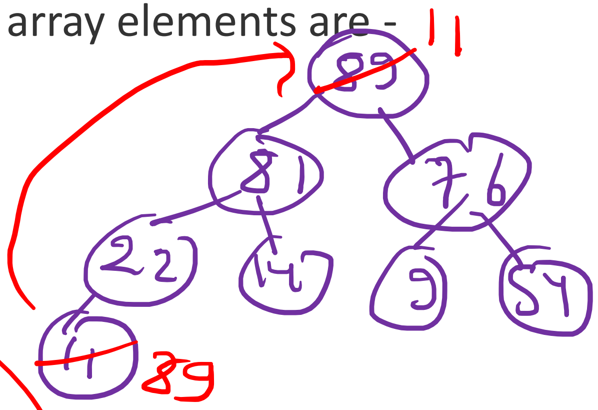
$i =$

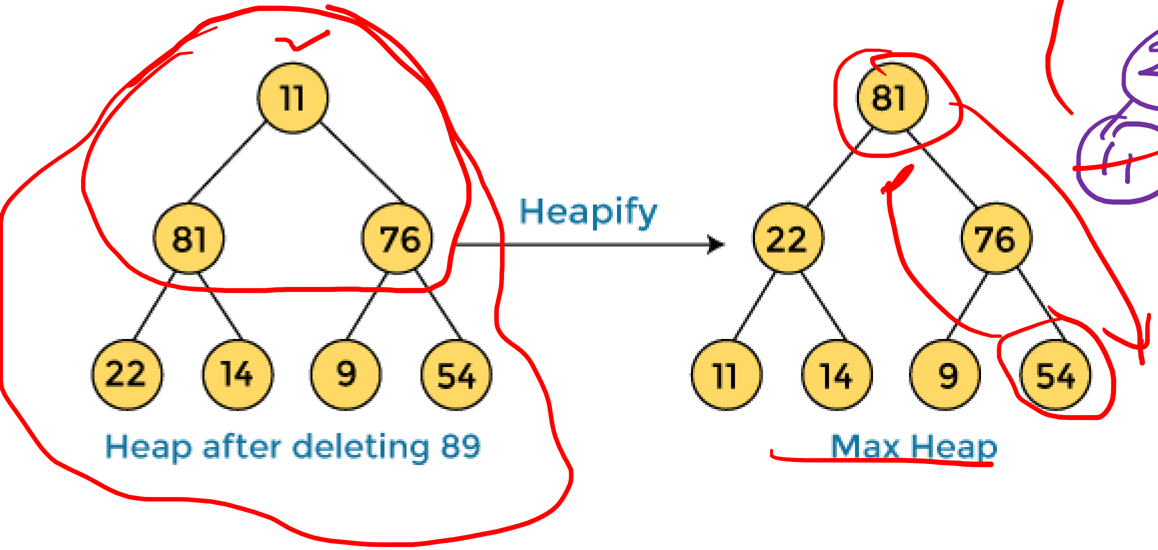
Left $= 2 \times i$

right $= 2i + 1$

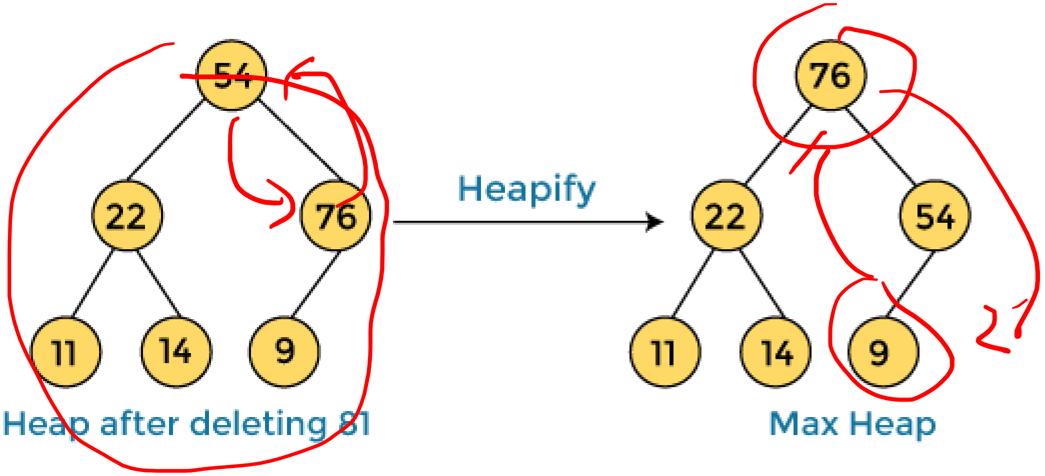
After converting the given heap into max heap, the array elements are -

89	81	76	22	14	9	54	11

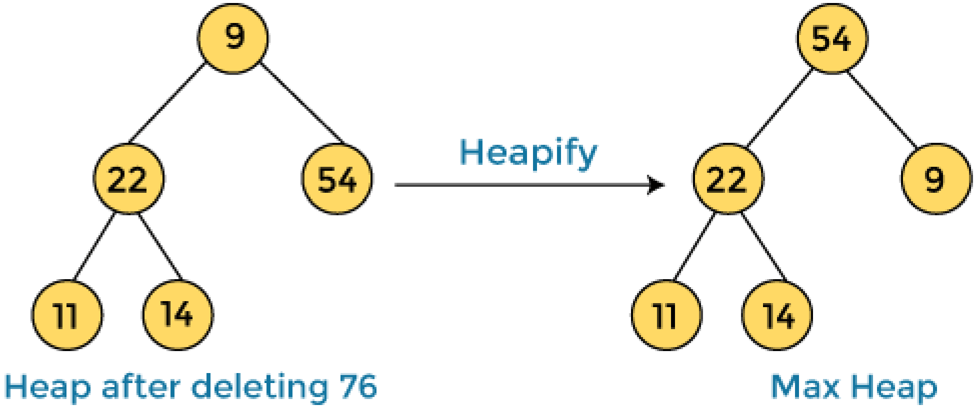




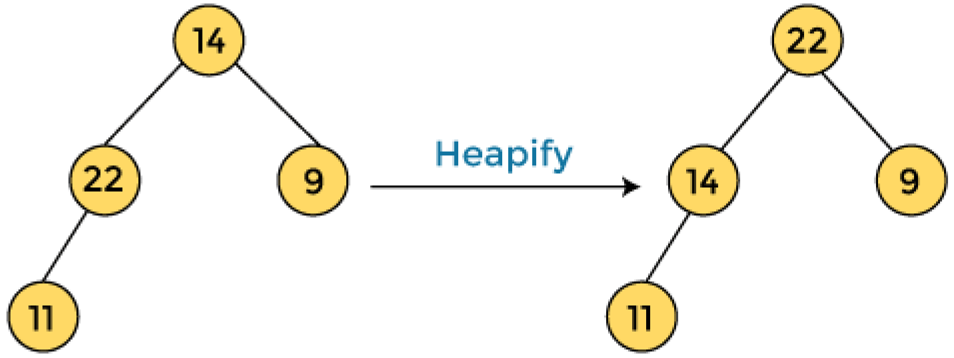
81	22	76	11	14	9	54	89



76	22	54	11	14	9	81	89



54	22	9	11	14	76	81	89

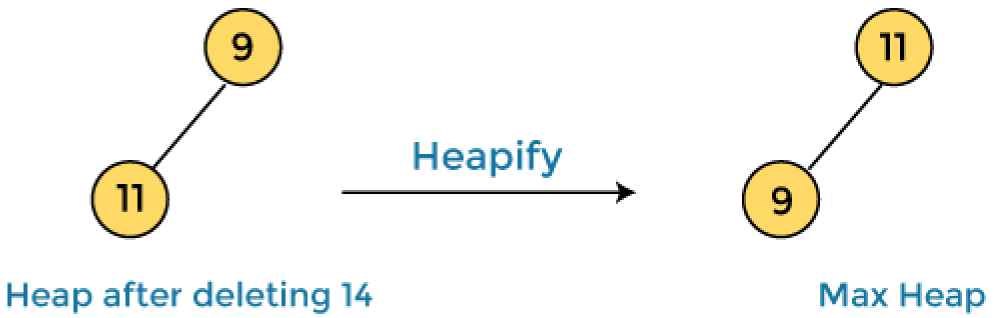


Heap after deleting 54 Max Heap

22	14	9	11	54	76	81	89

Heap after deleting 22 Max Heap

14	11	9	22	54	76	81	89



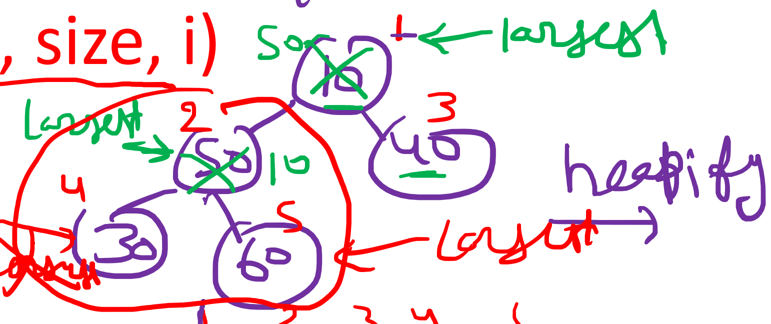
11	9	14	22	54	76	81	89

9	11	14	22	54	76	81	89

9	11	14	22	54	76	81	89

A 5 2 10 50 40 30 60

```
Heapify(A, size, i)
{
 largest=i
 L = 2i
 R= 2i + 1
While (L≤ size && A[L] > A[largest])
 {
   largest =L;
 }
While (R≤ size && A[R] > A[largest])
{
 largest =R;
}
```



A: 10 50 40 30 60

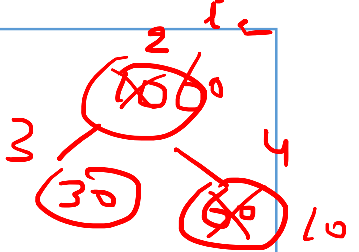
50 10 40 30 60

size=5
i=1
L=2
R=3

i=2
L=4
R=5

3 ≤ 5

5 ≤ 5



```
If(largest !=i)
{
swap(A[largest],A[i]);
Heapify (A, size,largest);
}
}
Heapsort(A,size)
{
For(i=size/2; i>0; i--)
{   heapify( A,size,i)
}
For(i=size ; i>0 ; i--)
{  swap(A[1],A[i]);
   Heapify(A,size,i);  }
}
```

insert

deletion

By Lakhan Singh

Radix Sort Algorithm

Radix sort is the linear sorting algorithm that is used for integers. In Radix sort, there is digit by digit sorting is performed that is started from the least significant digit to the most significant digit.

The process of radix sort works similar to the sorting of students names, according to the alphabetical order.

It is not an in-place sorting algorithm because it requires extra space.

Radix Sort is a stable sort because it maintains the relative order of elements with equal values.

Radix sort algorithm is a non-comparative sorting algorithm.

Algorithm

```
Radix Sort(arr)
max = largest element in the given array
d = number of digits in the largest element (or, max)
Now, create d buckets of size 0 - 9
for i -> 0 to d
sort the array elements using counting sort (or any stable sort) according to the digits at the ith place
```

181	289	390	121	145	736	514	212