Class:

→ Increasing order

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \cdots < 2^n < 3^n < \cdots < n^n$

Lower Bound — Avg Bound — Upper Bound

Asymptotic Notation.

* A simple method to represent Time Complexity
* To show class of a function.
* To compare the functions.
* 3 Notations.
  - O Big-oh Upper Bound
  - Ω Big omega Lower Bound
  - Θ theta Avg Bound → useful

  } If we cannot find exact place of a fun then we can go with UB&LB

* Time Complexity → $1 < \log N < \sqrt{n} < \cdots < n^n$ will be among

  OR

  It will be multiple of → $1 < \log N < \sqrt{n} < \cdots < n^n$

Big-oh :- The fun $f(n) = Og(n)$ iff ∃ +ve constant C & n_0 such that $f(n) \leq C \times g(n)$ $\forall n \geq n_0$

$f(n) = 2n+3$

$2n+3 \leq 10n$ → RHS → Single Term. $n \geq 1$ True

$f(n)$ C $g(n)$

Closest Bound ✓ $f(n) = O(n)$ True

$f(n) = O(n^2)$ True

$f(n) = O(2^n)$ True

$\therefore f(n) = O(n)$

or $2n+3 \leq 7n$ True

or $2n+3 \leq 1000n$ True

or $2n+3 \leq 2n+3n$

$5n$ True $n \geq 1$

or $2n+3 \leq 2n^2+3n^2$

$\leq 5n^2$ True $n \geq 1$

C $g(n)$

✗ $f(n) = O(\log n)$ false.

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \cdots < 2^n < 3^n < \cdots < n^n$.

Lowes Bound — Avg Bound — Upper Bound

Big Omega: The function $f(n) = \Omega(g(n))$ iff $\exists$ +ve constant c & n_0 such that $f(n) \geq c * g(n)$ $\forall$ $n \geq n_0$

Ex:

$f(n) = 2n + 3$.

$2n + 3 \geq 1 * n$ $\forall$ $n \geq 1$ ∴ $f(n) = \Omega(n)$ True

($f(n)$ ↑, c ↑, $g(n)$ ↑)

↓ Closest

$f(n) = \Omega(\log n)$ True

$f(n) = \Omega(n^2)$ false

Theta Notation: $f(n) = \Theta(g(n))$ iff $\exists$ +ve Constant c_1 c_2 & n_0.

Such that $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$.

Ex: $f(n) = 2n + 3$

$1 * n \leq 2n + 3 \leq 5 * n$. ∴ $f(n) = \Theta(n)$.

(c_1 ↑, $g(n)$ ↑, $f(n)$ ↑, c_2 ↑, $g(n)$ ↑)

Big-Oh

$f(n) = 3n+2 \quad g(n) = n.$

$f(n) = O(g(n))$

$f(n) \leq c\,g(n), \quad c > 0$

$f(n) \leq c\,g(n), \quad n_0 \geq 1$

Ex.

$3n+2 \leq cn$

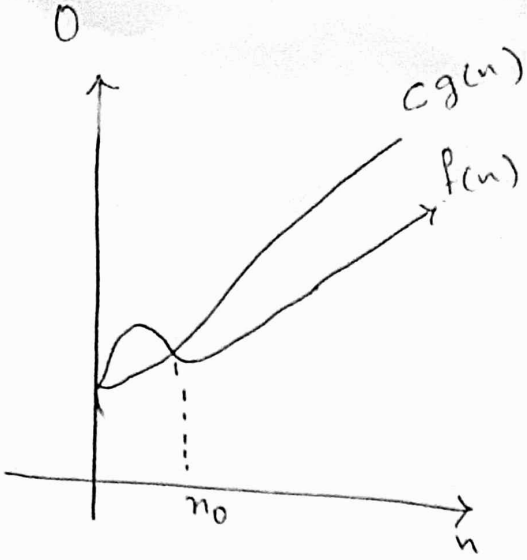
$c = 4$;

$c = 1$ ✗

$3n+2 \leq 4n$ $\quad c = 2$ ✓

$n \geq 2$ at $c = 4$ ⟹

$f(n) \leq cg(n)$. (c 4)



$f(n) \leq c\,g(n)$.

$n \geq n_0$

$c > 0, \quad n_0 \geq 1$

$f(n) = O(g(n))$.

$f(n) = 3n+2 \quad g(n) = n$

$f(n) = cg(n) \rightarrow$

$g(n) = n^2 \rightarrow$ True & Tightest Bound

$n^3 \rightarrow$ True

$n^n \rightarrow$ True

$2^n \rightarrow$ True

```
i=1, s=1
while( s <= n)
{
  i = i+1;
  s = i+1;
  ++i;
}
```

	$f(n)$.	$g(n)$.
	A()	B()
	{	{
	Some	Some
	}	}

No of st. : 5 > 2.

$[1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < n^4 \cdots < 2^n < 3^n < \cdots < n^n]$

Ex $f(n) = 2n^2 + 3n + 4$

$2n^2 + 3n + 4 \leq 2n^2 + 3n^2 + 4n^2$

$2n^2 + 3n + 4 \leq 9n^2 \quad n \geq 1$

$c \quad g(n)$

$\therefore f(n) = O(n^2)$.

Ex $f(n) = 2n^2 + 3n + 4$

$2n^2 + 3n + 4 \geq 1 \times n^2 \quad \Omega(n^2)$

Ex: $1 \times n^2 \leq 2n^2 + 3n + 4 \leq 9n^2 \quad \Theta(n^2)$.

$f(n) = n^2 \log n + n$

$1 \times n^2 \log n \leq n^2 \log n + n \leq 10\, n^2 \log n$

$O(n^2 \log n) \quad \Omega(n^2 \log n) \quad \Theta(n^2 \log n)$.

Algm

O(1)

Iterative

```
A()
{
 for(i=1 to N)
    more(a,b)
}
```

Recursive

```
A(n)
{
  if()
   A(n/2)
{
}
```

```
A()
{
for(i=1; i<=n; i++)
  S.O.P("Hello");
}
```

$A() = O(n)$.

Sum of i^{st} N Natural Nos $\frac{K(K+1)}{2}$.

```
A()
{
 i=1, s=1
 while(s<=n)
 {
   i++;
   s=s+i;
   S.O.P("Ravi")
 }
}
```

s: 1 3 6 10 15 21 --- n

i: 1 2 3 4 5 6 --- K

$= \frac{k(k+1)}{2} > n$

$\frac{k^2+k}{2} > n$

$K = O(\sqrt{n})$.

$k^2 > n$

$k = \sqrt{n}$.

```
A()
{
  i = 1
  for(i=1; i*i <= n; i++)
    S.o.p("Rahi");
}
```

$O(\sqrt{n})$.

$i^2 <= n$

$i = \sqrt{n}$.

```
A()
{
  int i, j, k, n;
  for(i=1; i<=n; i++)
  {
    for(j=1; j<=i; j++)
    {
      for(k=1; k<=100; k++)
      {
        sf("Rahi");
      }
    }
  }
```

i	1	2	3	4	5	$i = n$
j	1	2	3	4	5	$j = n$
k	1×100	2×100	3×100	4×100	5×100	$k = n\times100$

$= 100 + 2\times100 + 3\times100 + \cdots + n\times100$.

$= 100(1+2+3+\cdots+n)$.

$= 100\left(\frac{n(n+1)}{2}\right) = \frac{n^2+n}{2}$.

$O(n^2)$.

```
A()
{
  int i, j, k, n;
  for(i=1; i<=n; i++)
  {
    for(j=1; j<=i*i; j++)
    {
      for(k=1; k<=n/2; k++)
      {
        sf("Rahi");
      }
    }
  }
}
```

$i = 1 \quad i = 2 \quad i = 3$

$j = 1 \text{ time} \quad j = 4 \text{ time} \quad j = 9 \text{ times}$

$k = n/2 \times 1 \quad k = n/2 \times 4 \quad k = n/2 \times 9$.

$i = n$

$j = n^2$

$k = n/2 \times n^2$

$= n/2 \times 1 + n/2 \times 4 + n/2 \times 9 + \cdots + n/2 \times$

$= n/2(1+4+9+\cdots+n^2)$.

$= n/2\left(\frac{n(n+1)(2n+1)}{6}\right)$

$= O(n^4) \Rightarrow f(n) = n^k + n^{k-1} + \cdots = O(n^k)$

```
A()                    6
{                      3
for (i=1; i<n; i=i*2)
  ply ("ravi");
}
```

$i = 1 \quad 2 \quad 4 \quad \dots \quad n$

$2^0 \quad 2^1 \quad 2^2 \quad \dots \quad 2^k$

$2^k = n \Rightarrow k = \log n.$

$O(\log_2 n).$

$O(\log_3 n)$

$O(\log_6 n).$

$2^k = n.$

$k = \log n$

```
A()
{
int. i, j, k;
for(i= n/2 ; i<=n; i++)  --> n/2
 for (j=1; j<=n/2; j++)  --> n/2.
  for (k=1; k<=n; k=k*2) --> log n
    ply ("ravi");
```

n^2

$n/2 * n/2 * \log_2 n.$

$= O\left[\frac{n^2}{2} * \log_2 n\right]$

$=$

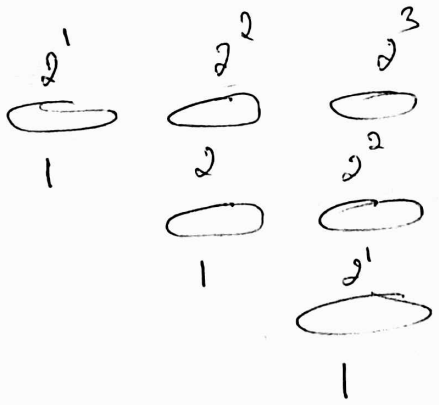
```
A()
{
int i,j,k;
for(i=n/2; i<=n; i++)  ---> n/2
  for(j=1; j<=n; j=2*j)  --> log2 n.
    for(k=1; k<=n; k=k*2)  -- log2 n
      printf("ravi");
}
```

$= n/2 \times \log_2 n \times \log_2 n$

$= O\left(n(\log_2 n)^2\right)$

Assume

```
A()
{
while(n>1)
  {
    n = n/2;
  }
}
```



$n = 2^k$

$k = \log_2 n$

$k = \log_5 n$

$\lfloor \log_2 20 \rfloor$

```
A()
{
for(i=1; i<=n; i++)
  for(j=1; j<=n; j=j+i)
    Pf("Rani");
}
```

$i=1$	$i=2$	$i=3$	$i=k$	$i=n$
$j=1$ to n	$j=1$ to n	$j=1$ to n	$j=1$ to n	$j=1$ to n
n time	$n/2$ time	$n/3$ time	n/k time	$n/1$ time

$$= n\left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$$

$$= n(\log n)$$

$$= O(n(\log n))$$

```
A()
{
int n = 2^(2^k);   // k times
for(i=1; i<=n; i++)
  {
    j=2
    while(j<=n)
    {
      j = j^2
      Pf("Rani");
    }
  }
}
```

$k=1$	$k=2$	$k=3$
$n=4$	$n=16$	$n = 2^{2^3} = 2^8$
$j=2,4$	$j=2,4,16$	$j=2, 2^2, 2^4, 2^8$
$n \times 2$ times	$n \times 3$ times	$n \times 4$ times

Total No of times S.O.P $n*(k+1)$.

$$n = 2^{2^k}$$

→ Apply log on Both sides

$$\log_2 n = 2^k$$

$$\log \log n = k$$

$$O(n \log\log n).$$

~~if(a>b)~~

{

temp = a

a = b;

b = temp;

}

$O(1)$.

Recursive - Back

① A(n)
{
if( )
ret (A(n/2) + A(n/2))
}

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

② A(n)
{
if(n>1)
return (A(n-1))
} $T(n) = 1 + T(n-1) \rightarrow (1)$

$T(n) = 1 + 1 + T(n-2)$
$= 2 + T(n-2)$
$= 2 + 1 + T(n-3)$
$= 3 + T(n-3)$
$= k + T(n-k)$
$=$

Back Substitution:

$T(n-1) = 1 + T(n-2)$ — (2)

$T(n-2) = 1 + T(n-3)$ — (3)

$n - k = 1$

$k = n - 1$

$= (n-1) + T(n-(n-1))$

$= (n-1) + T(1) = n - 1 + 1$

$= n.$

$T(n) = n$

$O(n)$

$T(n) = n + T(n-1)$; $n > 1$

$T(n-1) = (n-1) + T(n-2)$ — (1)

$T(n-2) = (n-2) + T(n-3)$ — (3)

(2) in (1)

$T(n) = n + T(n-1)$

$= n + (n-1) + T(n-2)$

$= n + (n-1) + (n-2) + T(n-3)$

$= n + (n-1) + (n-2) + \cdots + (n-k) + T(n-(k+1))$.

$n-(k+1) = 1$

$n-k-1 = 1$

$\Rightarrow \boxed{k = n-2}$

$= n + (n-1) + (n-2) + \cdots + (n-(n-2)) + T(n-(n-2+1))$.

$= n + (n-1) + (n-2) + \cdots + 2 + 1$

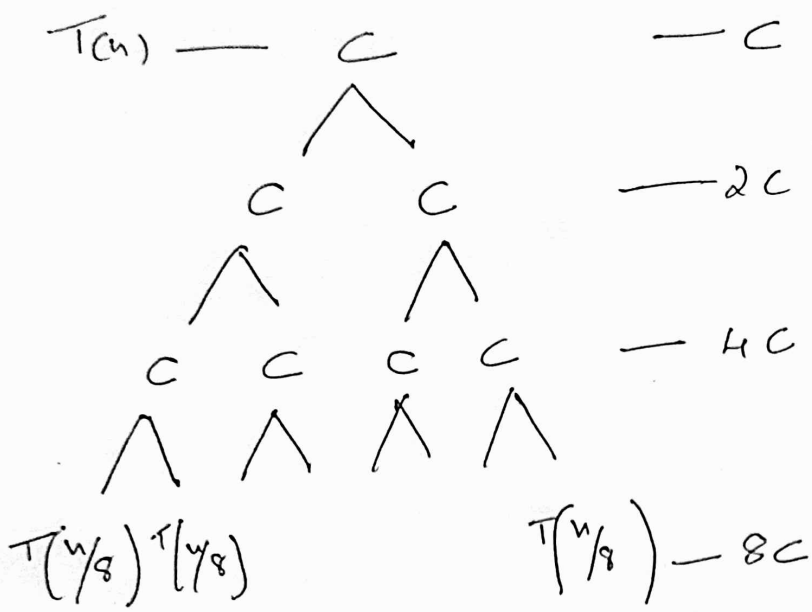
$= \frac{n(n+1)}{2} = O(n^2)$

Recursion Tree Method:

$T(n) = 2T(n/2) + c \quad ; \quad n > 1$

$= c \quad ; \quad n = 1$

Assume $n = 2^k$



$T(1) = T(n/n)$

$c + 2c + 4c + \cdots + nc$

$c(1 + 2 + 4 + \cdots + 2^k)$

$= c\left(\frac{1(2^{k+1} - 1)}{(2-1)}\right)$

$= c(2^{k+1} - 1)$

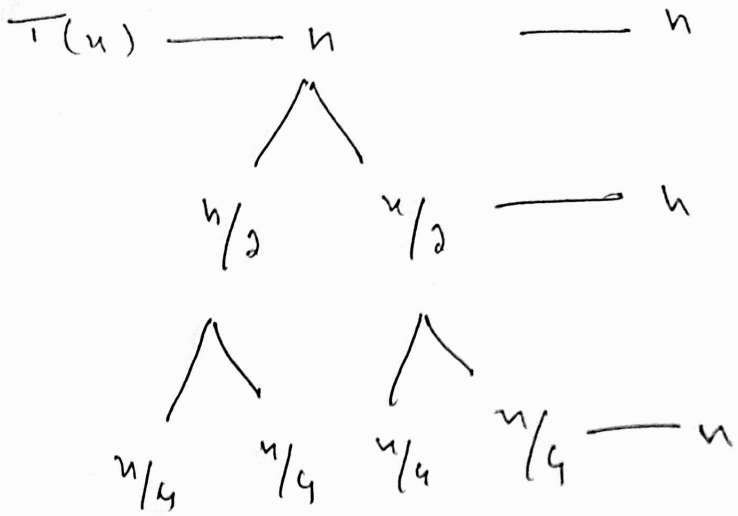
$= c(2n - 1)$

$= O(n).$

$T(n) = 2T(n/2) + n \quad ; \quad n > 1$

$= 1 \quad ; \quad n = 1$

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



$T(1) \cdot T(1) \; T(1) \; T(1) \cdot \text{—} \; n.$

$\frac{n}{2^0} - \frac{n}{2^1} - \frac{n}{2^2} \cdots \frac{n}{2^k}$

$(k+1) \quad n = 2^k$

$k = \log n$

$= n(\log n + 1)$

$O(n \log n)$.

$T(n) = aT(n/b) + \Theta(n^k \log^p n)$. $(\log n)^n$ $\log^2 n$.

$a \geq 1$, $b > 1$, $k \geq 0$ & p is real No $\log n \times \log n$ $\log \log n$.

1) if $a > b^k$, then $T(n) = \Theta(n^{\log_b a})$

2) if $a = b^k$

a) if $p > -1$, then $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$.

b) if $p = -1$, then $T(n) = \Theta(n^{\log_b a} \log \log n)$

c) if $p < -1$, then $T(n) = \Theta(n^{\log_b a})$.

3) if $a < b^k$

a) if $p \geq 0$ then $T(n) = \Theta(n^k \log^p n)$

b) if $p < 0$ then $T(n) = O(n^k)$.

1). ~~$T(n) = 3T(n/2) + n^2$~~.

$a = 3$ $b = 2$ $k = 2$ $p = 0$.

a	b^k
3	2^2

$3 < 4$.

$T(n) = \Theta(n^2 \log^0 n)$

$= \Theta(n^2)$.

3a).

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

$a=4 \quad b=2 \quad k=2 \quad p=0.$

a	b^k
4	2^2

$4 = 4$

2a) $T(n) = \theta(n^{\log_2 4} \log^1 n).$

$= \theta(n^2 \log n).$

3) $T(n) = T(n/2) + n^2.$

$a=1 \quad b=2 \quad k=2 \quad p=0.$

a	$<$	b^k
1		2^2

$1 < 4.$

3a. $T(n) = \theta(n^2 \log^0 n).$

4) $T(n) = 2^n T(n/2) + n^n.$ * Masters Theorem is not applied.

5) $T(n) = 16T(n/4) + n$

$a=16 \quad b^4 \quad k=1 \quad p=0$

$16 > 4^1 \quad T(n) = \theta(n^{\log_b a}).$

$= \theta(n^{\log_4 16})$

$= \theta(n^2).$

(5) $T(n) = 16\{T(n/4) + n$.

$a=16 \quad b=4 \quad k=1 \quad p=0$

$16 > 4^1 \quad T(n) = \theta(n^{\log_b a})$

$= \theta(n^{\log_4 16})$

$= \theta(n^2)$

(6) $T(n) = 2T(n/2) + n^1 \log^1 n$

$a=2 \quad b=2 \quad k=1 \quad p=1$

$2=2^1$

2a $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$.

$= \theta(n^1 \log^2 n)$

$= \theta(n \log^2 n)$.

(7) $T(n) = 2T(n/2) + n/\log n$

$= 2T(n/2) + n \log^{-1} n$.

$a=2 \quad b=2 \quad k=1 \quad p=-1$

$2=2^1$

2b. $T(n) = \theta(n^{\log_b a} \log\log n)$

$= \theta(n \log\log n)$

8) $T(n) = 2T(n/4) + n^{0.51}$

$a=2 \quad b=4 \quad k=0.51 \quad p=0$

$2 < 4^{0.51}$

3.a) $T(n) = \Theta(n^k \log^b n)$

$= \Theta(n^{0.51})$

9) $T(n) = 0.5\,T(n/2) + 1/n$ ✗

$a = 0.5$ (a should be greater than or Equal to 1 but $a=0.5$)!

10) $T(n) = 6T(n/3) + n^2 \log n.$

$a=6 \quad b=3 \quad k=2 \quad p=1$

$6 < 3^2$

3a) $T(n) = \Theta(n^k \log^b n)$

$= \Theta(n^2 \log^1 n)$

$= \Theta(n^2 \log n)$

11) $T(n) = 64\,T(n/8) - n^2 \log n.$

we do some problem & problem will get sub divided into

we Cannot apply master theorem when minus (-).

12) $T(n) = 7T(n/3) + n^2$

$a=7 \quad b=3 \quad k=2 \quad p=0$

$7 < 3^2$

3a) $T(n) = \theta(n^k \log^p n)$.

$= \theta(n^2 \log^0 n)$

$= \theta(n^2)$

13) $T(n) = 4T(n/2) + \log n$.

$a=4 \quad b=2 \quad k=0 \quad p=1$

$4 > 2^0 \quad T(n) = \theta(n^{\log_b a})$

$= \theta(n^2)$.

14) $T(n) = \sqrt{2}\, T(n/2) + \log n$

$a=\sqrt{2} \quad b=2 \quad k=0 \quad p=1$

$\sqrt{2} > 2^0$

$T(n) = \theta(n^{\log_b a}) = \theta(n^{\log_2 \sqrt{2}})$

$= \theta(\sqrt{n})$.

15) $T(n) = 2T(n/2) + \sqrt{n}$

$a=2 \quad b=2 \quad k=1/2 \quad p=0$

$2 > 2^{1/2} \quad T(n) = \theta(n^{\log_2 2})$

$= \theta(n)$

16) $T(n) = 3T(n/2) + n$.

$a = 3 \quad b = 2 \quad k = 1 \quad p = 0$.

$3 > 2^1 \quad T(n) = \theta(n^{\log_b a}) = \theta(n^{\log_2 3})$

17) $T(n) = 3T(n/3) + \sqrt{n}$

$a = 3 \quad b = 3 \quad k = 1/2 \quad p = 0$

$3 > 3^{1/2}$

$T(n) = \theta(n^{\log_3 3})$

$= \theta(n)$.

18) $T(n) = 4T(n/2) + cn^1$

$a = 4 \quad b = 2 \quad k = 1 \quad p = 0$

$a \quad b^k$

$4 > 2 \quad T(n) = \theta(n^2)$

19) $T(n) = 3T(n/4) + n \log n$

$a = 3 \quad b = 4 \quad k = 1 \quad p = 1$

$3 < 4^1$.

3a) $T(n) = \theta(n^1 \log^1 n)$

$= \theta(n \log n)$.

Insertion Sort (A, n)

{

for (j ← 2 to n) : 2, 3, 4, 5, 6 → $O(n-1)$

{ : $O(n)$

key = a[j]

i = j − 1;

while (i > 0 & A[i] > key)

{ ↳ $O(n)$

a[i+1] = A[i]

i = i − 1;

} : $O(n^2)$

a[i+1] = key

1	2	3	4	5	6
~~30~~	50	10	60	20	40
i	j				

key = 50

j = 2

i = j − 1 = 1

while (1 > 0 & 30 > 50)

T F

a[i+1] = key

a[2] = 50

1	2	3	4	5	6
~~30~~	~~50~~	10	60	20	40
	i	j			

key = 10

i = 2

while (2 > 0 & 50 > 10)

T T

1	2	3	4	5	6
30	50	~~10~~ 50	60	20	40
i					

while (1 > 0 & 30 > 10).

1	2	3	4	5	6
10	30	50	60	20	40
	i	j			

Key = 60

i = 3

while (3 > 0 & 50 > 60)

T F → false

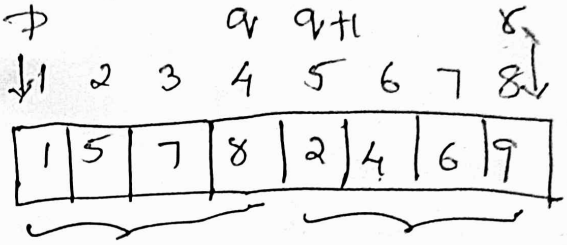
1	2	3	4	5	6
10	30	50	60	20	40
	20	30	50	60	

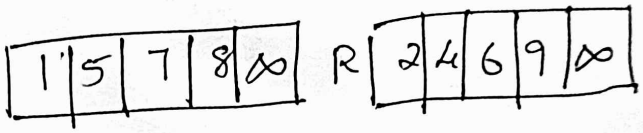
1	2	3	4	5	6
10	20	30	50	60	~~60~~ 40
			40	50	60

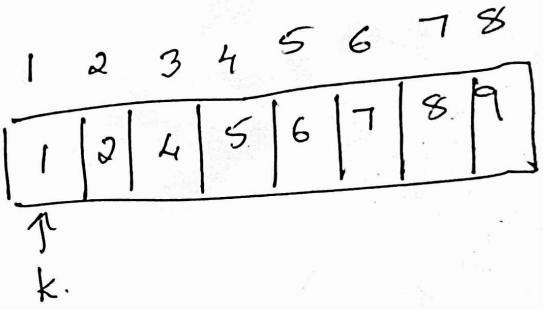
1	2	3	4	5	6
30	50	10 50	60	20	40
i=0	10	30			

Merge Sort.

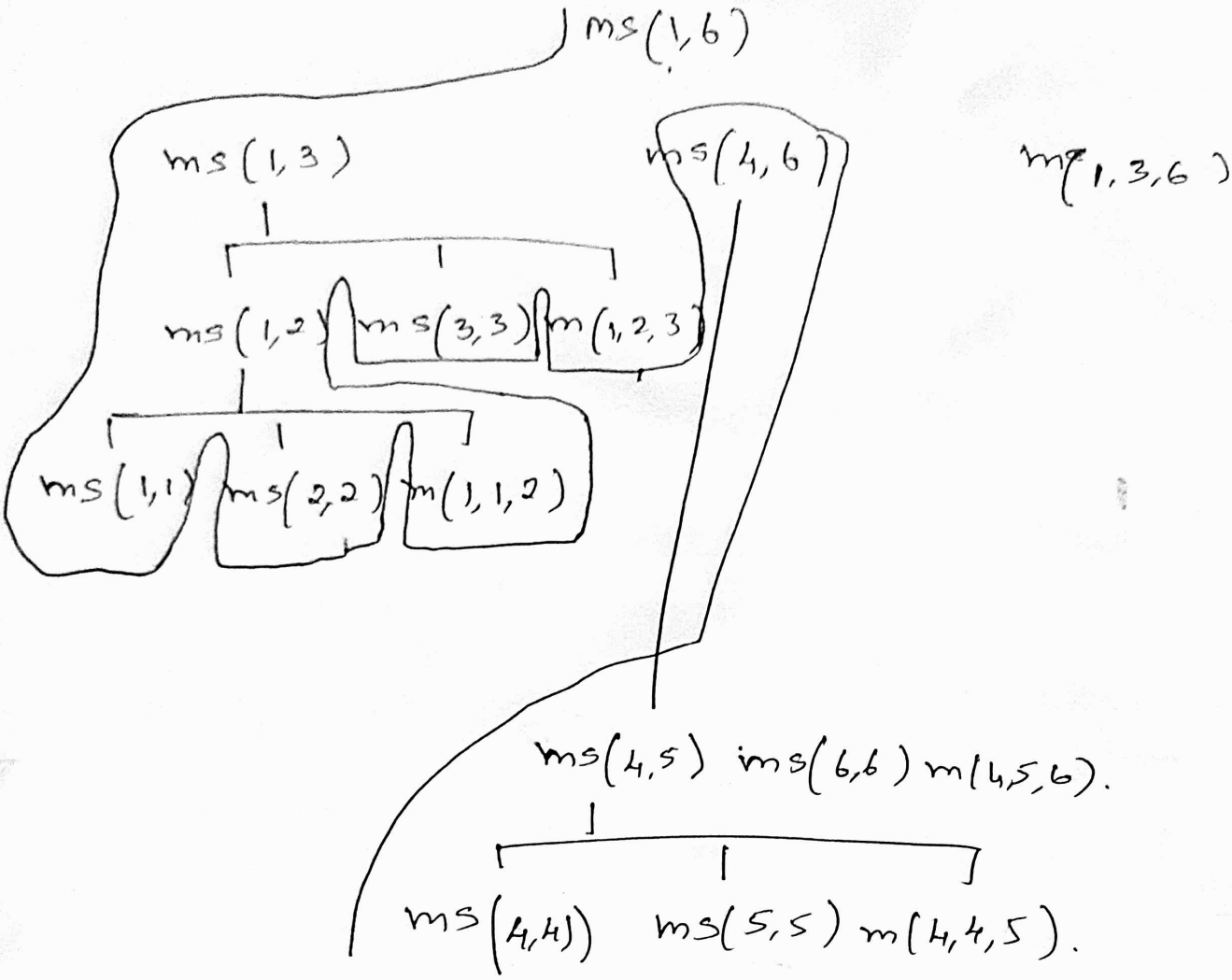
```
Merge (A, p, q, r)
{
  n1 = q - p + 1;
  n2 = r - q.
Let L[1 ---- n1+1] & R[1 to n2+1] be new arrays
for (i = 1 to n1)
    L[i] = A[p + i - 1]
for (j = 1 to n2)
    R[j] = A[q + j]
  L[n1+1] = ∞
  R[n2+1] = ∞
    i = 1, j = 1
for (k = p to r)
if (L[i] ≤ R[j])
      A[k] = L[i]
      i = i + 1;
else  A[k] = R[j]
      j = j + 1;
```








```
merge_sort (A, p, r)
{ if p < r
    q = ⌊(p + r)/2⌋
  merge_sort (A, p, q)
  merge_sort (A, q+1, r)
  merge (A, p, q, r)
}
```


ms(1,6)
ms(1,3)
ms(4,6)
ms(1,2)
ms(3,3)
m(1,2,3)
ms(1,1)
ms(2,2)
m(1,1,2)
ms(4,5)
ms(6,6)
m(4,5,6).
ms(4,4)
ms(5,5)
m(4,4,5).

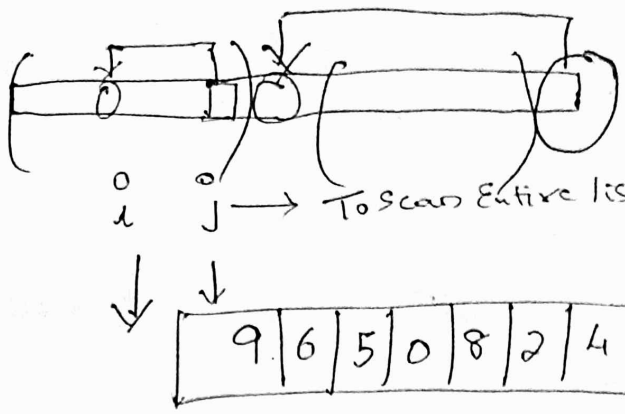
m(1,3,6)

9650 82

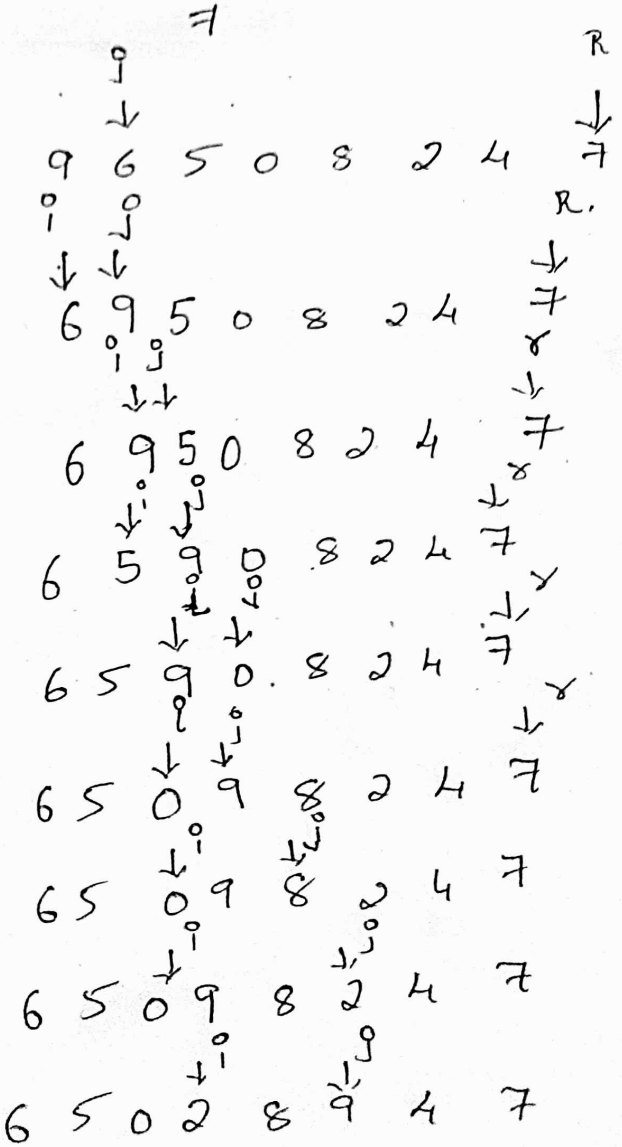
Quick Sort

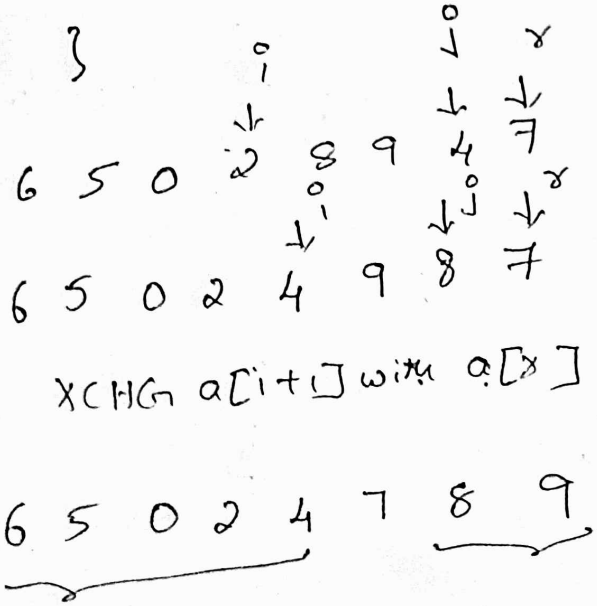
$O(n)$.

```
Partition(A, P, r)
{
  x = A[r]
  i = p-1
  for(j = p to r-1)
  {
    if(A[j] ≤ x).
    {
      i = i+1
      Exch a[p] with a[j]
    }
  }
  Xchg A[i+1] with A[r];
  return i+1
}
```



j → To scan Entire list & to find any Element less than 7

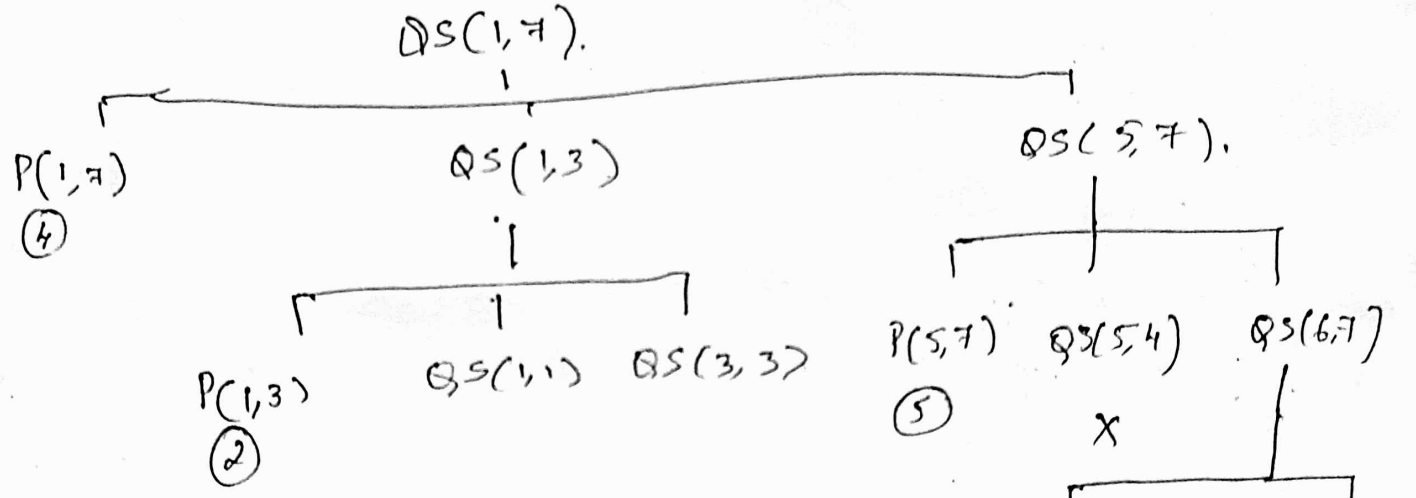


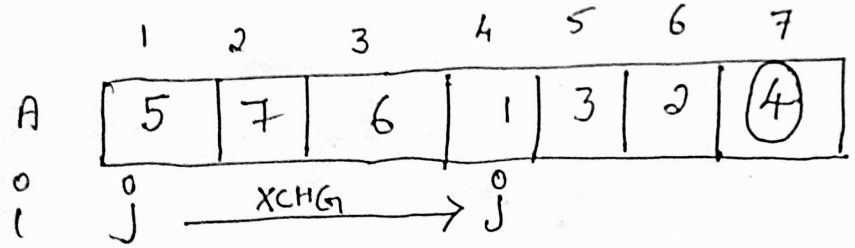


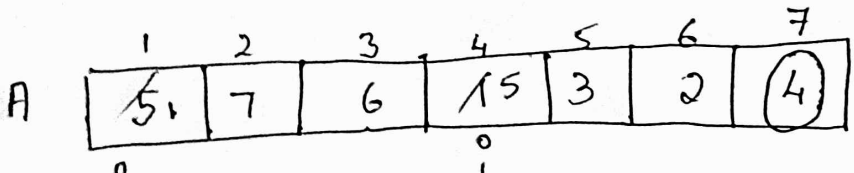
XCHG a[i+1] with a[r]

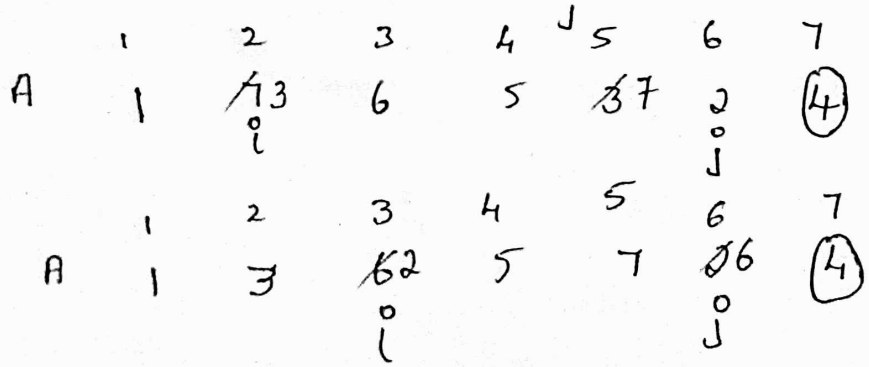
6 5 0 2 4 7 8 9

```
Quick Sort (A, P, r)
{
  if (p<r)
  {
  q = Partition (A, P, r).
  Quick Sort (A, P, q-1)
  Quick Sort (A, q+1, r).
```

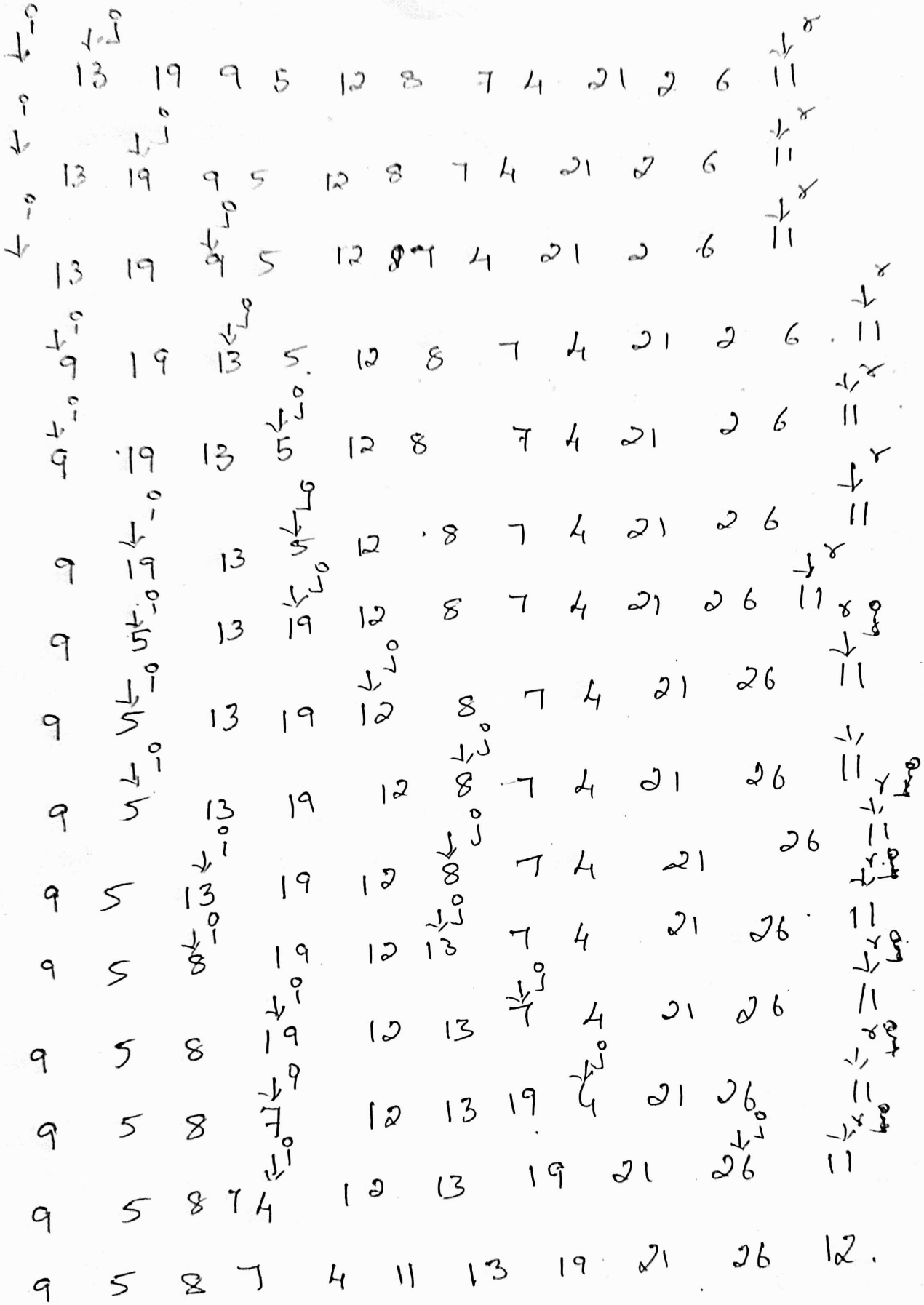


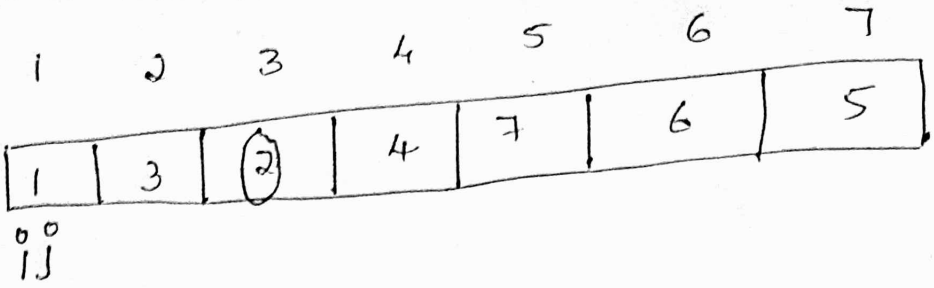




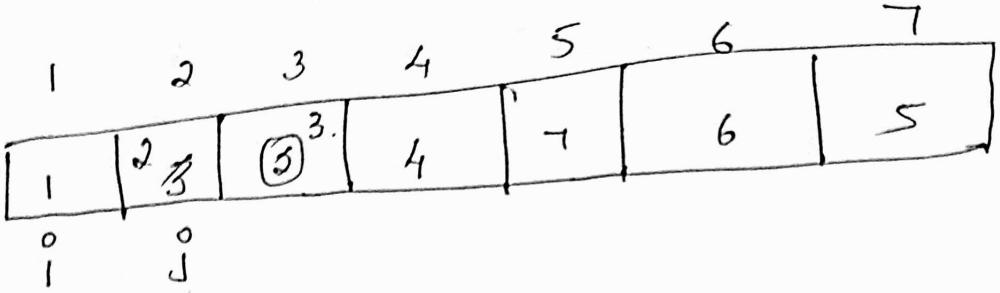


1	2	3	4	5	6	7
1	3	2	4	7	6	5

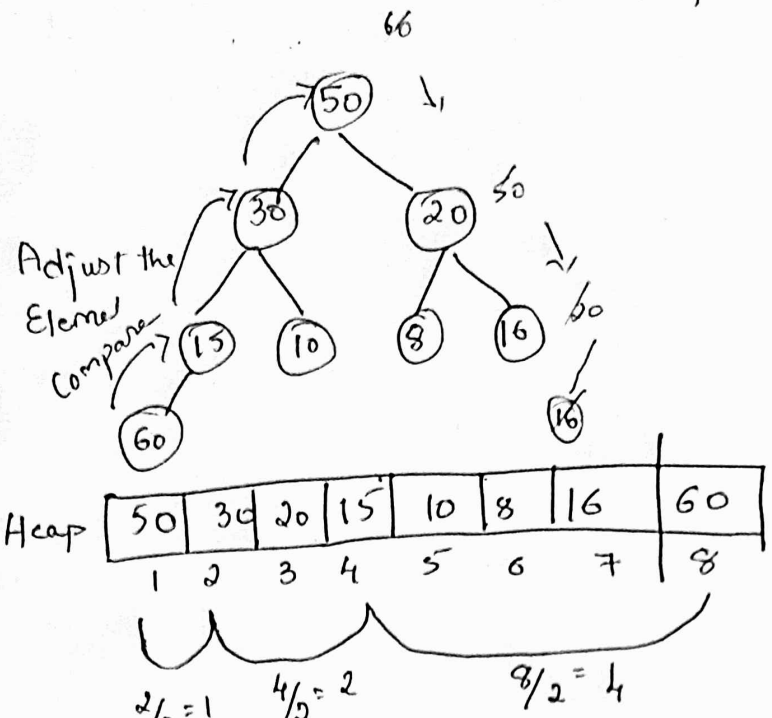


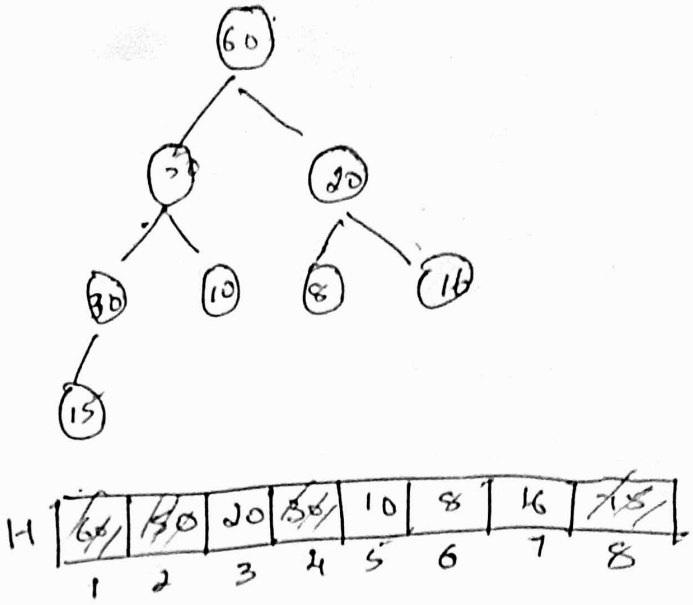


XCHG i & j both pointing same



Max Heap - Insert Operation





Time taken for swaps $\log n$ depends on Height.

$O(1)$ to $O(\log n)$

Ex: Insert value (6)

No swaps / Zero swaps

(1) Add new Element as leaf & Adjust the Element

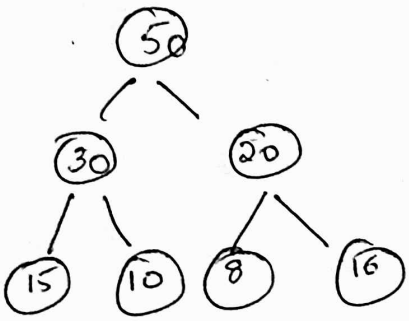
(2) Inserting -> Element is sent up wards

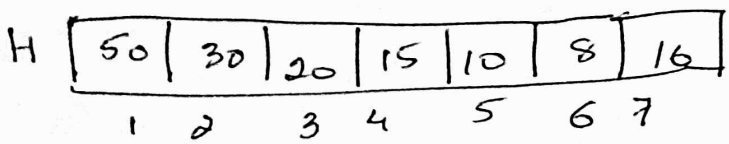
(3) Adjustment from leaf to root

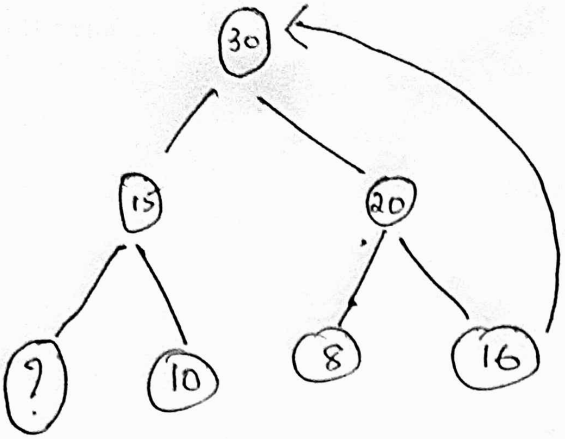
Max Heap - Delete Operation

* You cannot delete any Element apart from Root
* Use of Apples
* last Element will always take place of root
* Adjust the Element from Root towards leaf.

(1)

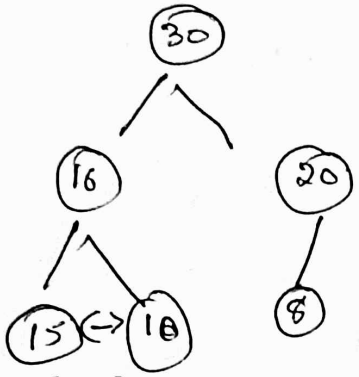






wrong delete

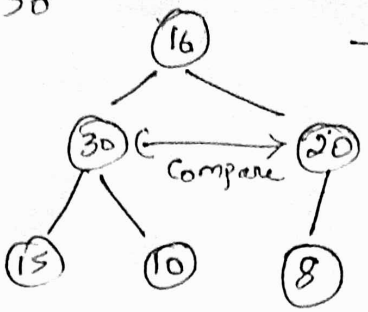
Find max Heap.



H

30	10	20	15	10	8	
1	2	3	4	5	6	7

50



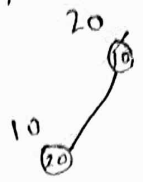
→ Complete Bin Tree but not max Heap.

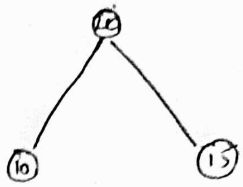
H

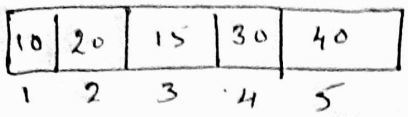
16	30	20	15	10	8	
1	2	3	4	5	6	7.

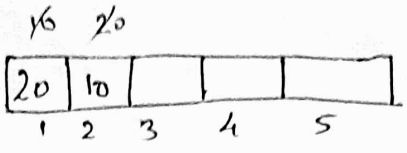
Creating & Deleting Heaps

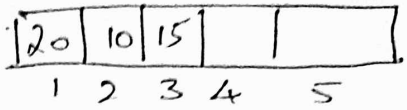
* Single node can be min/max Heap

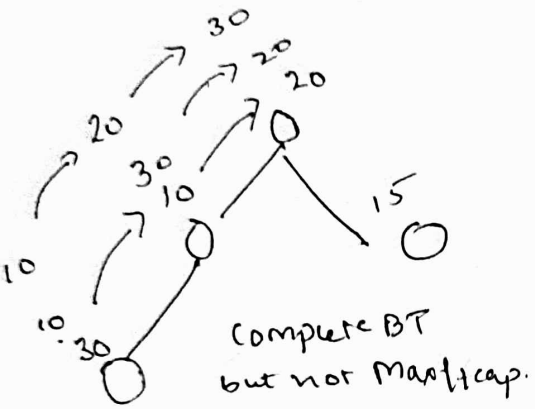


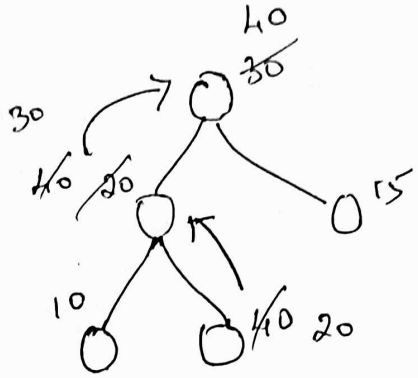


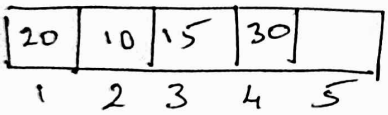


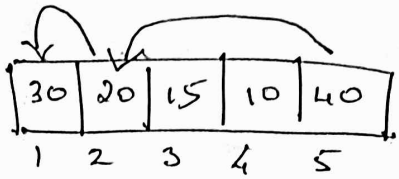










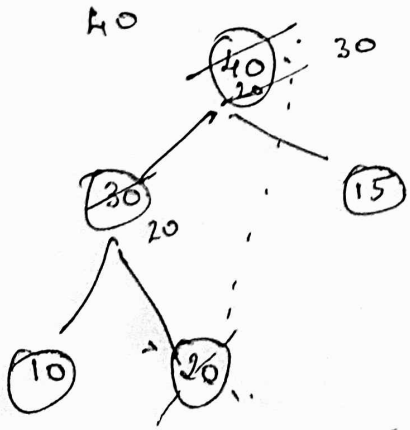


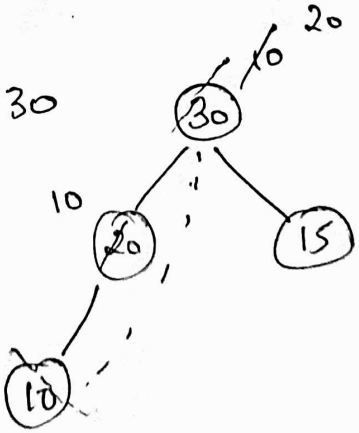
20 10 30 15

20 30 10 15

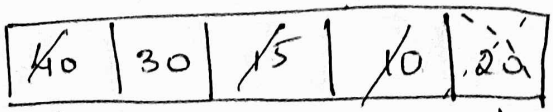
30 20 10 15

Deletion

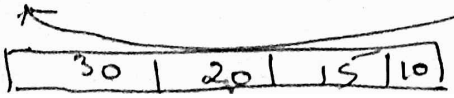


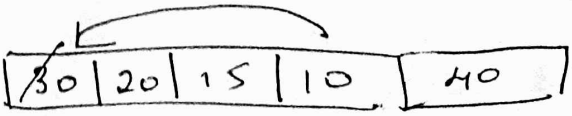


20 15



40



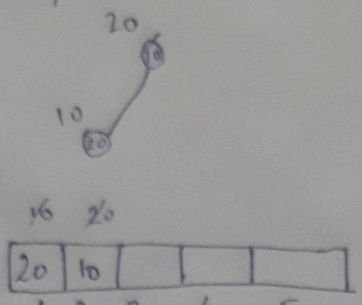


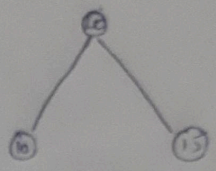
10

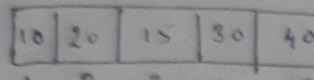
20 10 15

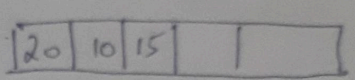
Creating & Deleting Heaps

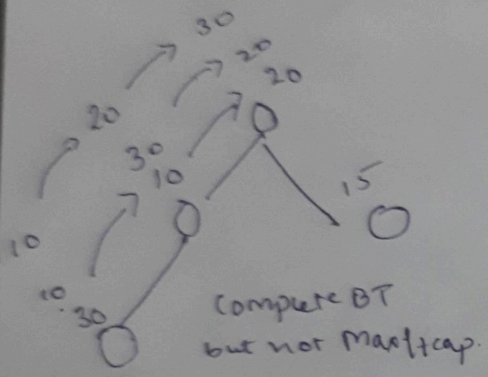
* Single node can be min/max Heap



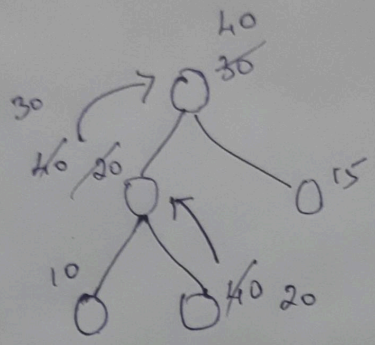


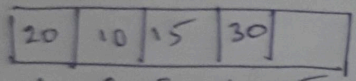






Complete BT but not Max Heap.

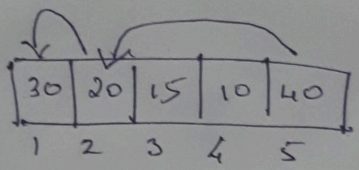




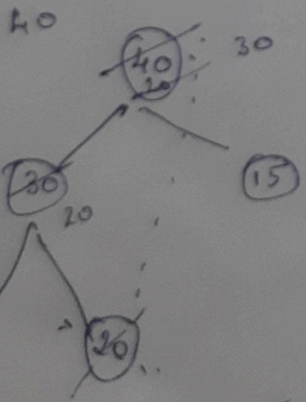
20 10 30 15

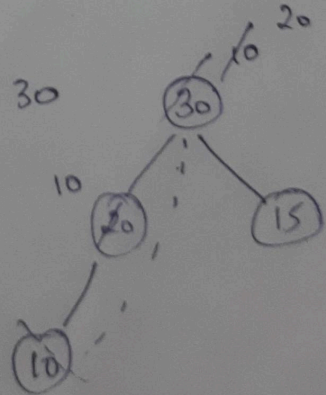
20 30 10 15

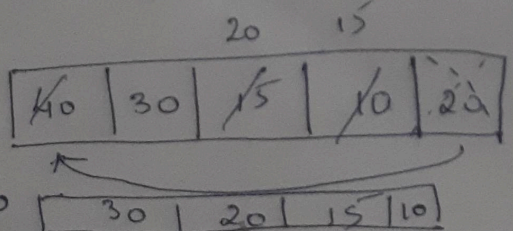
30 20 10 15

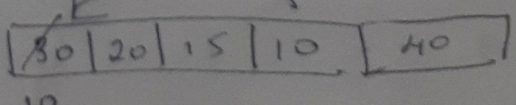


Deletion



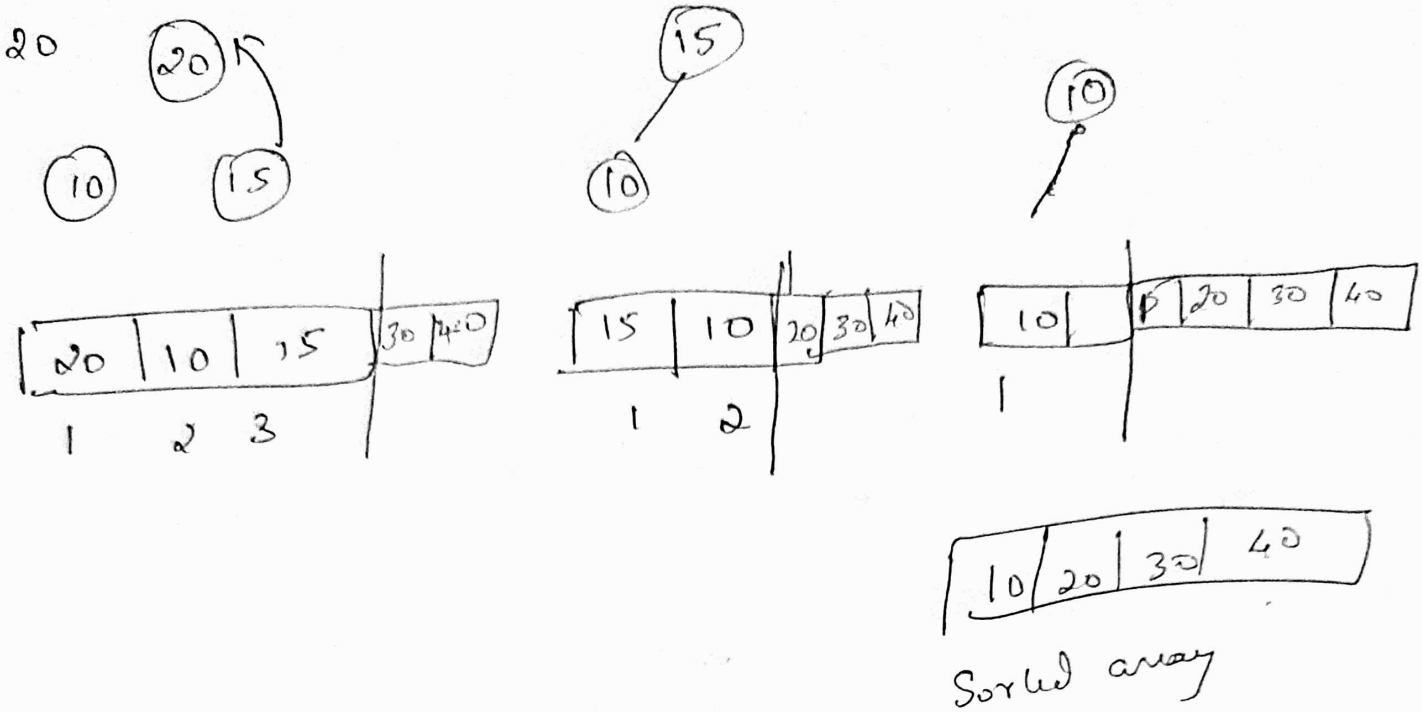






10

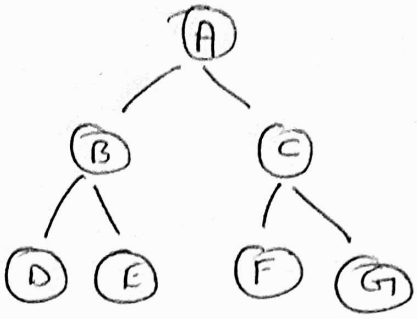
20 10 15


20
20
10
15
20 10 15 30 40
1 2 3
15
10
15 10 20 30 40
1 2
10
10 20 30 40
1
10 20 30 40
Sorted array

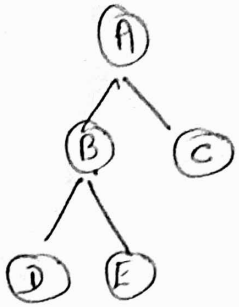
$2^{h+1} - 1$

Heap

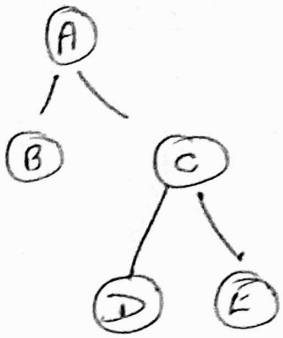
Complete Bin Tree

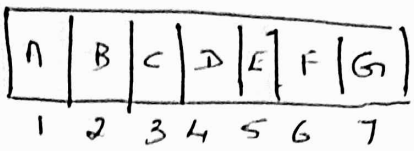


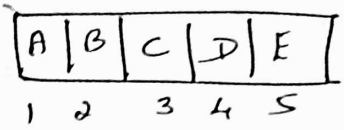
Complete Bin Tree

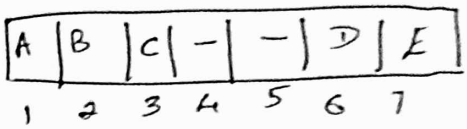


Not a Complete Bin Tree







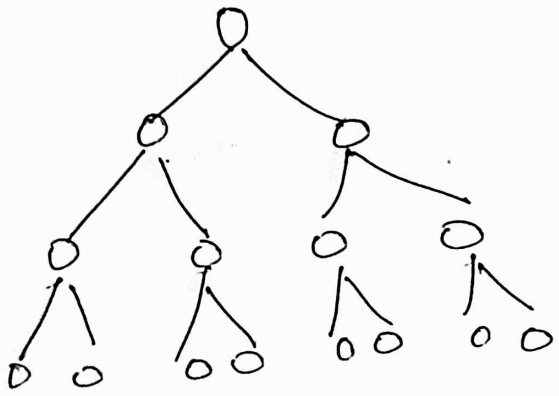


If a Node is at Index - i^0

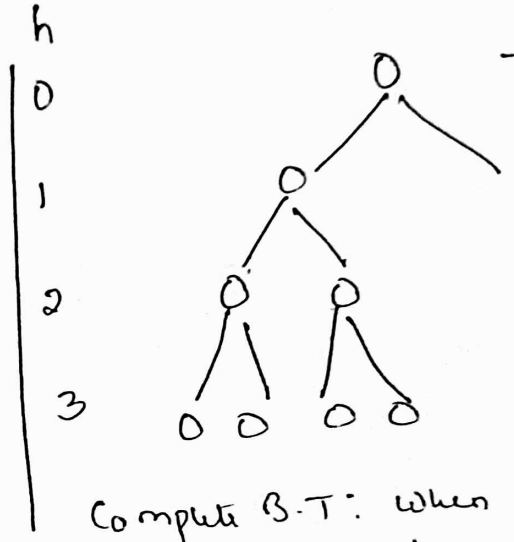
its left Child is at — $2 * i^0$

its right child is at - $2 \times i^0 + 1$

its present is at — $\left\lfloor \frac{i^0}{2} \right\rfloor$

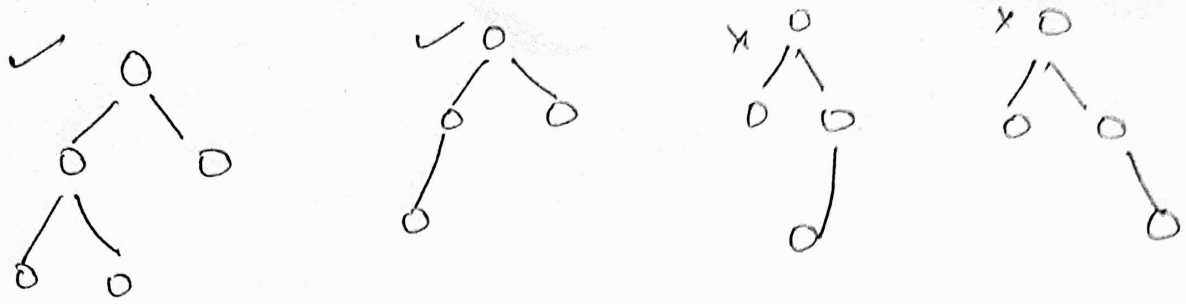


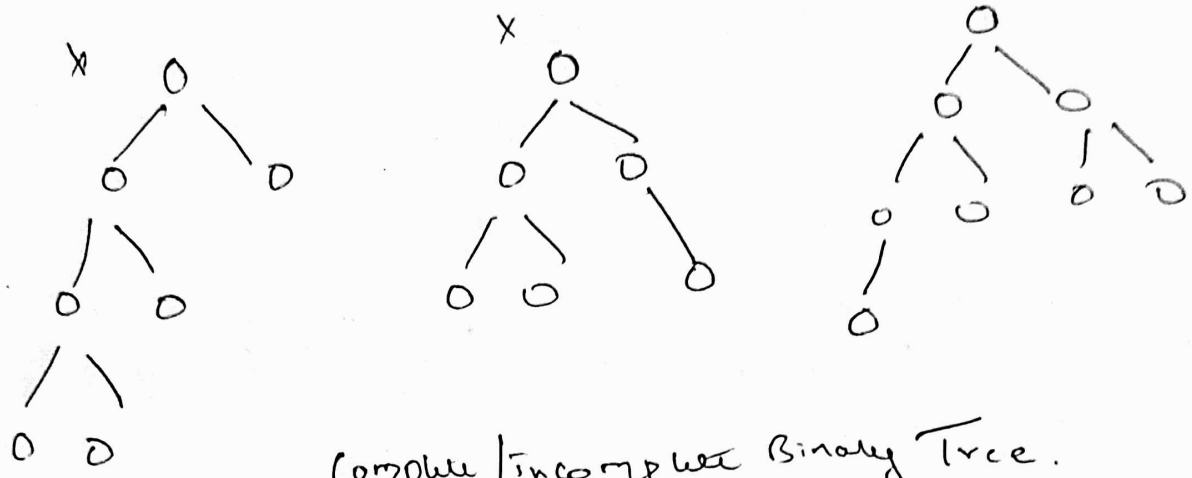
Full Binary Tree: A B.T with max No of Nodes is known as full B.T



Complete B.T: When we represent in an array there is no missing element.

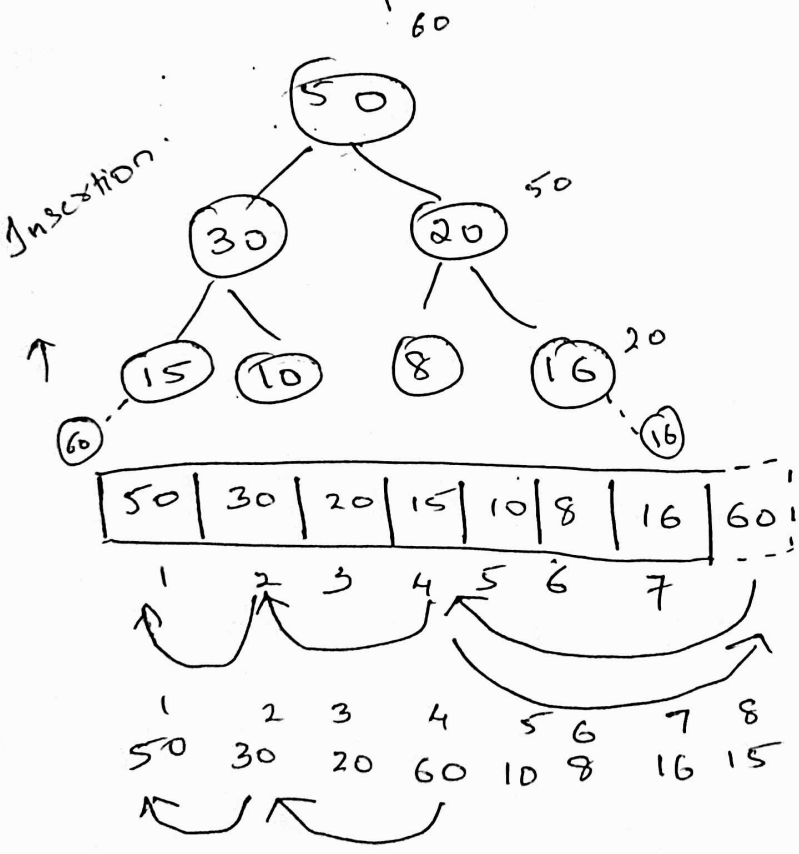
A full B.T upto height h-1 & in last level Elements are filled from left to right.



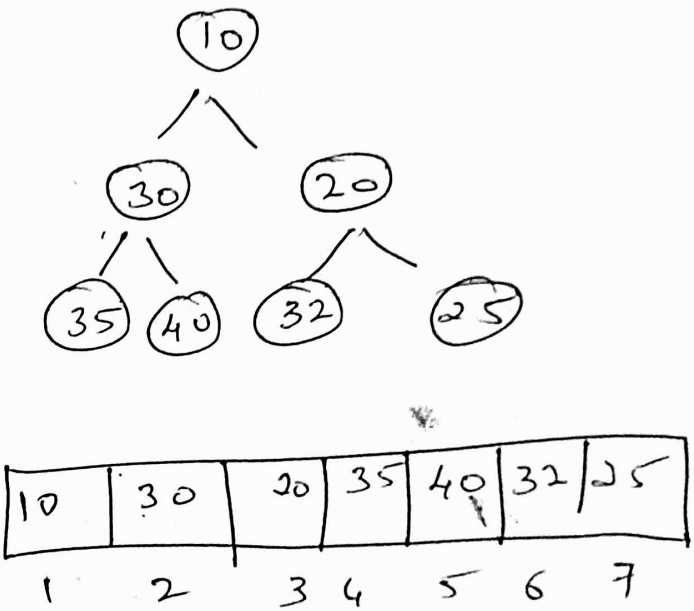


Complete / Incomplete Binary Tree.

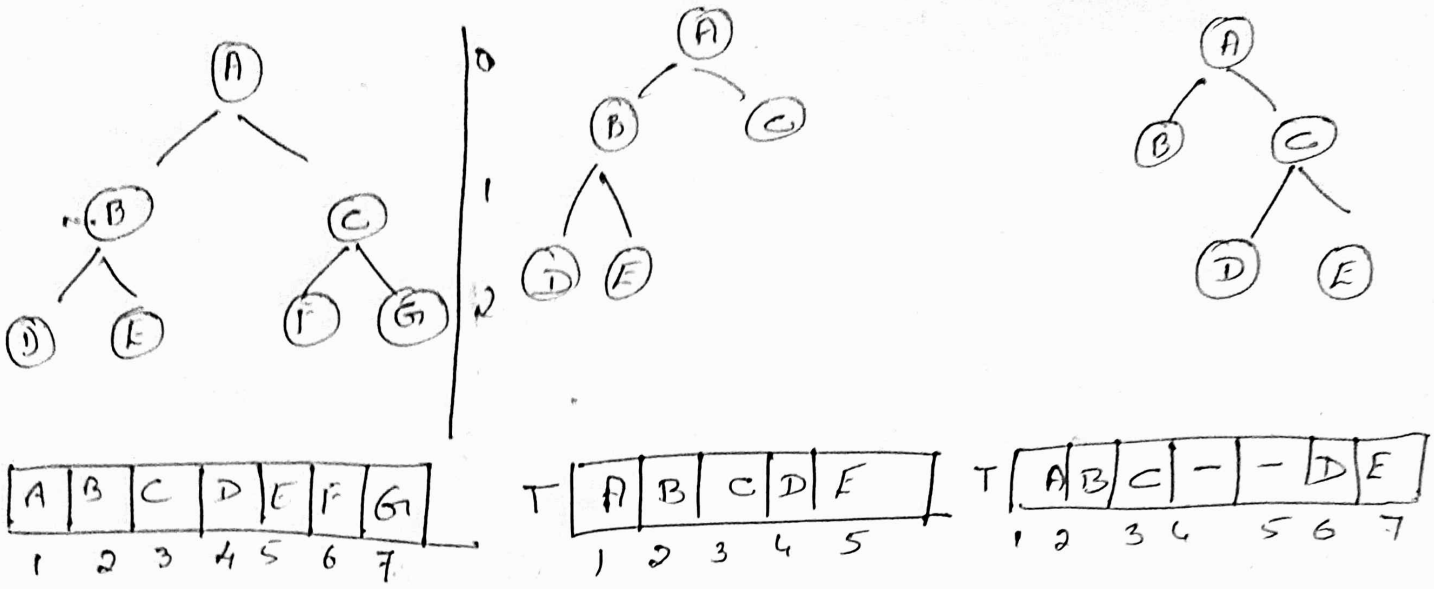
Max Heap



Min Heap



$2^{h+1} - 1$

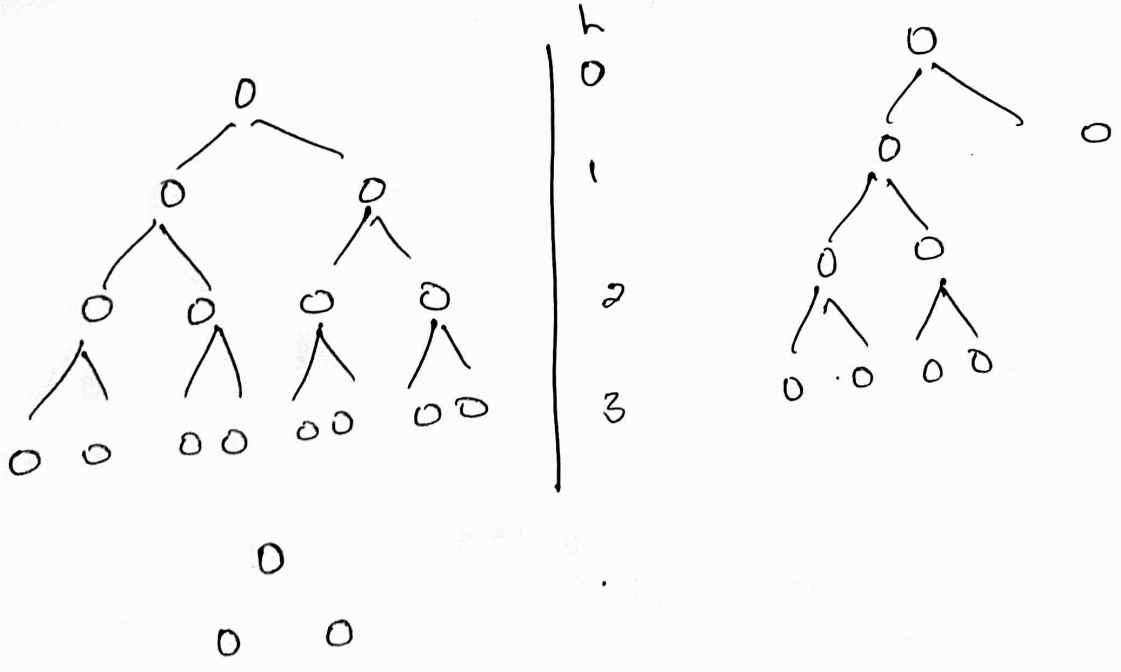


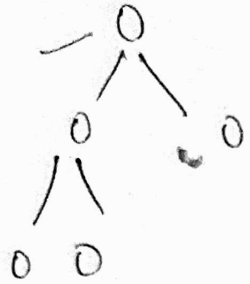
if a Node is at Index: i

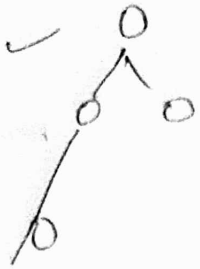
if its Left child is at: $2*i$

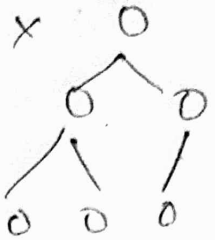
if its Right child is at: $2*i+1$

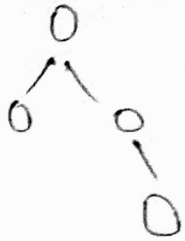
its Parent is at: $\left\lfloor \frac{i}{2} \right\rfloor$

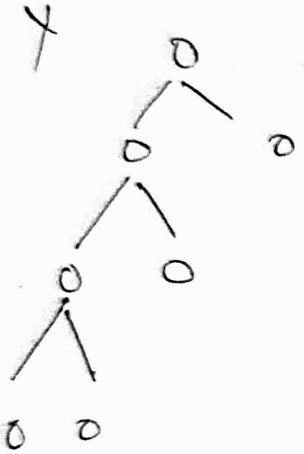


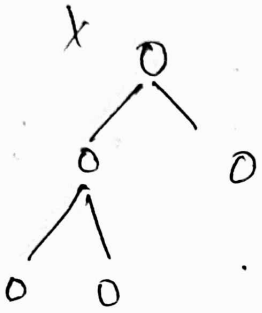




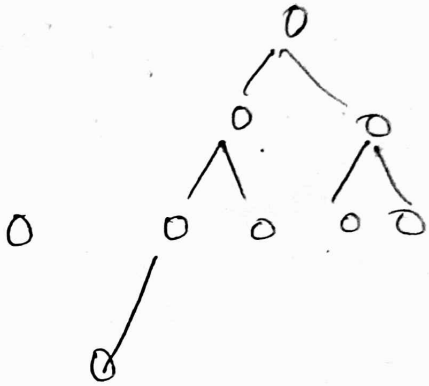








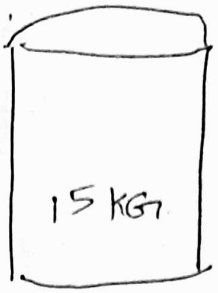
log n



Knapsack Problem.

n=7	Objects	1	2	3	4	5	6	7
m=15	Profits	10	5	15	7	6	18	3
	weights	2	3	5	7	1	4	1
	P/w	5	1.3	3	1	6	4.5	3.
	x	(1	2/3	1	0	1	1	1)
		x_1	x_2	x_3	x_4	x_5	x_6	x_7

$0 \leq x \leq 1$



15 - 1 = 14
14 - 2 = 12
12 - 4 = 8
8 - 5 = 3
3 - 1 = 2
2 - 2 = 0

$\sum x_i w_i = 1 \times 2 + \frac{2}{3} \times 3 + 1 \times 5 + 0 \times 7 + 1 \times 1 + 1 \times 4 + 1 \times 1$

$= 2 + 2 + 5 + 0 + 1 + 4 + 1 = 15$

$\sum x_i P_i = 1 \times 10 + \frac{2}{3} \times 5 + 1 \times 15 + 1 \times 6 + 1 \times 18 + 1 \times 3$

$= 10 + 2 \times 1.3 + 15 + 6 + 18 + 3 = 54.6.$

Constraint

$\sum x_i w_i \leq m$

Objective: $\max \sum x_i P_i$

```
Knapsack (w[1 ---- n], p[1 ---- n], W)
- for i = 1 to n
      do x[i] = 0
  weight = 0
  for i = 1 to n.
      if weight + w[i] ≤ W then.
          x[i] = 1
          weight = weight + w[i].
      else
          x[i] = (W - weight) / w[i]
          weight = W.
          break
      return x
```

Strassen's matrix Multiplication.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$m \times n$

$2 \times 2 \quad 2 \times 2$

$$C_{ij} = \sum_{k=1}^{n} A_{ik} * B_{kj}$$

```
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] += A[i,k] * B[k,j];
    }
  }
}
```

$O(n^3)$

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

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

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

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

$A = [a_{11}] \quad B = [b_{11}]$

$C = [a_{11} \times b_{11}]$.

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

A_{11} A_{12} A_{21} A_{22} B_{11} B_{12} B_{21} B_{22}

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

Algorithm: $MM(A, B, n)$

{

if ($n \leq 2$)

{ $c = 4$ formulas

}

else { $mid = n/2$

$mm(A_{11}, B_{11}, n/2) + mm(A_{12}, B_{21}, n/2)$

$mm(A_{11}, B_{12}, n/2) + mm(A_{12}, B_{22}, n/2)$

$mm(A_{21}, B_{11}, n/2) + mm(A_{22}, B_{21}, n/2)$

$mm(A_{21}, B_{12}, n/2) + mm(A_{22}, B_{22}, n/2)$

}

}

$$T(n) = \begin{cases} 1 & n \leq 2 \\ 8\,T(n/2) + n^2 & n > 2 \end{cases}$$

$a = 8$

$b = 2$

$f(n) = n^2$

$\log_b^a = \log_2^8 = 3$

$n^k = n^2$.

$\Theta(n^3)$.

$P = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) \cdot B_{11}$

$R = A_{11} \cdot (B_{12} - B_{22})$

$S = A_{22} \cdot (B_{21} - B_{11})$

$T = (A_{11} + A_{12}) \cdot B_{22}$

$U = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$

$V = (A_{12} - A_{22}) (B_{21} + B_{22})$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

$$T(n) = \begin{cases} 1 & n \leq 2 \\ 7T(n/2) + n^2 & n > 2 \end{cases}$$

$\log_2^7 = 2.81 \quad k = 2$

$O(n^{\log_2 7}) = O(n^{2.81})$.