## Solution 1 -

We will find the tight bound by calculating both upper and lower bounds.

**(i) Upper Bound :**

For upper bound, take max. term i.e. $n^3 \log n$ . Since every term is less than or equal to this and there are total 'n' number of terms, upper bound will be

= no. of terms * largest valued term

= n * $n^3 \log n$

= $n^4 \log n$

**(ii) Lower Bound :**

For lower Bound, We consider second half of series i.e.

$$(\frac{n}{2})^3 \log \frac{n}{2} + \ldots\ldots + n^3 \log n$$

These are $\frac{n}{2}$ terms and the lowest term is $(\frac{n}{2})^3 \log \frac{n}{2}$ . So, lower bound will be

= no. of terms * lowest valued term

= $(\frac{n}{2}) * (\frac{n}{2})^3 \log \frac{n}{2}$

= $(\frac{n}{2})^4 \log \frac{n}{2}$

thus lower bound is also $n^4 \log n$ .

Since, Both lower and upper bounds are $n^4 \log n$ . So, tight bound will also be $n^4 \log n$ .

Thus, sum of $\sum_{i=1}^{n} i^3 \log i$ is theta $(n^4 \log n)$

## Solution 2-

S(n) = $a + ar + ar^2 + \ldots + ar^{n-1}$

case 1. if r < 1

we can see that

$a \le S(n) \le \frac{a}{1-r}$ (

and since a and r both are constants, we can say that S(n) is theta(constant).
or theta(F). where F=a.

Case 2. if r > 1

we know

$$ar^{n-1} \le S(n) \le a(\frac{r^n - 1}{r-1}) = a(\frac{r^{n-1} - 1/r}{1 - 1/r})$$

and since a and r both are constants, we can say that S(n) is theta $(r^{n-1})$
or theta(L) where L= $ar^{n-1}$

## Solution 3-

Given : (1)a series of n points ordered clockwise which forms convex polygon P
(2)input point X

To find : Whether given point X present inside convex polygon P.

Solution :

We can use approach similar to binary search here. A convex polygon is the one which has all internal angles < 180. If we draw one line between opposite endpoints , the two polygons which form afterwards remain convex. We use this special property of convex polygon to design recursive algorithm.

Recursive Algorithm -

(1) Base case : if n=3, suppose p1,p2,p3 are 3 points and input point X.
Make 3 triangles - p1-X-p2 , p2-X-p3 , p3-X-p1.
calculate areas of above 3 triangles, add them and check whether calculated area equals to area formed by points p1,p2 and p3. If both areas are equal then point X is inside polygon formed by p1,p2 and p3, else not.

(2) Recursive step : if n > 3, join 1st point p1 to p(n/2), making 2 convex polygons and line L separating them. Now recurse on the polygon depending on which side the point lies with respect to line L.

Complexity : Our recurrence relation of above algorithm will be $T(n) = T(n/2) + c$ (c is constant). It can be easily verified as base case will take O(1) or constant time and recursive step will take T(n/2) as we are calling same algorithm to half number of points. Solving(Same as binary search), we get complexity O(logn).