1. Use mathematical induction to prove:

For any natural number n, $n^3 - n$ is divisible by 3

Solution:-

Mathematical induction:-

For any natural number n, n^3-n is divisible by 3.

(1) Let n=1, 2,

1^3-1=0=3(0)
2^3-2=6=3(2)

According to Mathematical induction, from above steps, assume this will be true for n=k,

(2) Let n=k,

k^3-k=3m _____(i)
(which is divisible by 3)

(3) Let n=k+1, if n=k is true then n=k+1 will be true,

$(k+1)^3-(k+1) = k^3+3k^2+3k+1-k-1$
$= (k^3-k)+3(k^2+k)$
$= 3m+3(k^2+k)$ (:. Substituting value of k^3-k from equation(i))
$= 3(m+ k^2+k)$ (which is divisible by 3)

So, according to above proof of mathematical induction process, function n^3-n is divisible by 3.

2. Write algorithm to calculate Factorial by following the requirements:

a. Write algorithm using loop. then prove the algorithm is correct

Solution:-

Pseduo code of Algorithm using loop to calculate Factorial of number.

```
read n
fact=1, i=1
while(i<=n)
        fact=fact*i
        i=i+1
display fact
```

We have to use Mathematical induction to prove correctness of above algorithm,

For any natural number n, n!= n*(n-1)*(n-2)*….*3*2*1 = n*(n-1)!

(1) Let n=1, 2, 3

1! = 1
2! = 2*1! = 2*1 = 2
3! = 3*2! = 3*2*1 = 6

According to Mathematical induction, from above steps, assume this will be true for n=k,

(2) Let n=k,

k! = k*(k-1)*…*2*1 = k*(k-1)! _____(i)

(3) Let n=k+1, if n=k is true then n=k+1 will be true,

(k+1)! = (k+1)*k*(k-1)*….*2*1
= (k+1)*k! (:. Substituting value of k! from equation(i))

So, according to above proof of mathematical induction process, function n!=n*(n-1)! is true.

b. Write algorithm using recursion, then calculate the running time

Solution:-

Pseduo code of Algorithm using recursion to calculate Factorial of number.

```
read n
if n=1
      return 1
return n*fact(n-1)
```

Total running time,

$T(n) = C_1+C_2(n-1)$

$= C_2(n-1)+C_1$

$= C_2n+C_1-C_2$ (compared to an+b)

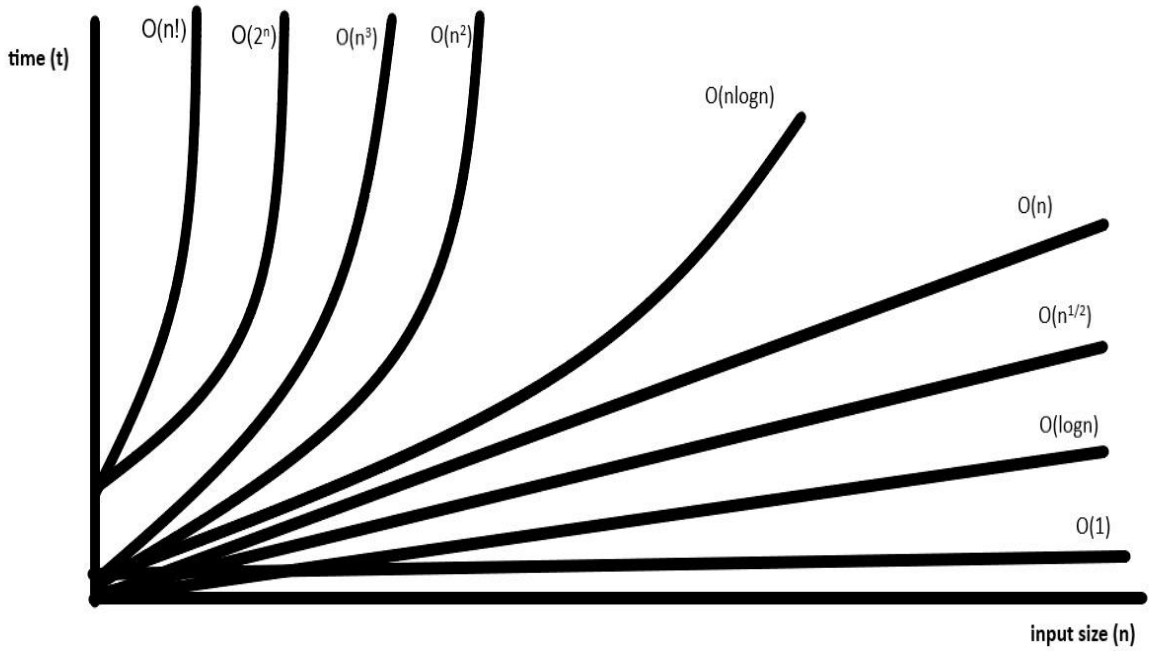
$O(n)$

So, total time to running above algorithm is O(n).

3. draw the function curve to compare the following different running times (n is the input size)

$n!$, 2^n, n^3, n^2, $n \log n$, n, $(n)^{1/2}$, $\log n$

Solution:-



4. Given two sorted arrays, use any language you are familiar to implement algorithm to merge the two arrays so that the resulting array is sorted. what is time complexity of this merge?

Solution:-

Algorithm to merge the two arrays so that the resulting array is sorted in java.

```
public class Merge
{
        public static int[] merge(int [] a, int [] b)
        {
          int l1=a.length;
          int l2=b.length;

          int [] sorted=new int[l1+l2];

          int i=0,j=0,k=0;

          while(i<l1 && j<l2)
          {
                if (a[i]<b[j])
                {
                        sorted[k++]=a[i++];
                }
                else
                {
                        sorted[k++]=b[j++];
                }
          }
```

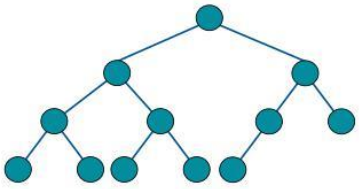
```
    while(i<l1)
    {
            sorted[k++]=a[i++];
    }
    while(j<l2)
    {
            sorted[k++]=b[j++];
    }
    return sorted;
  }

  public static void main (String [] args)
  {
    int [] a={1,3,6,7};
    int [] b={2,5,8,9,11};

    int [] sorted=merge(a,b);

    for(int i=0;i<sorted.length;i++)
    {
            System.out.println(sorted[i]+" ");
    }
  }
}
```

5. A complete binary tree is a binary tree that every level, except possibly the last, is completely filled, and all nodes are as far left as possible.



if total number of nodes is N, what is the level of the tree?

Solution:-

Suppose height of tree is h,

then number of levels (L) = h+1. _____(i)

maximum possible nodes in each level of complete binary tree are,

at level 1, number of nodes = 1 = 2^0

at level 2, number of nodes = 2=2^1

at level 3, number of nodes = 4=2^2

at level 4, number of nodes = 8=2^3

at level 5, number of nodes =16= 2^4

.
.
.

at level h, number of nodes = $2^{(h-1)}$

at level h+1, number of nodes = 2^h

total number of nodes = $2^0+2^1+2^2+2^3+\ldots..+2^{(h-1)}+ 2^h$

$= 1+2+4+8+\ldots..+2^{(h-1)}+2^h$

n= total number of nodes (as given in question)

n= $1+2+4+8+\ldots..+2^h$

n= $2^{(h+1)}$ (According to $S_n=[a(r^n-1)]/[n-1]$)

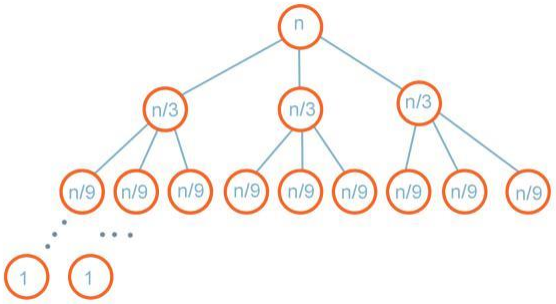
$n = 2^L$ (because, L= h+1 as per equation(i))

$\log_2(n) = L$

$L = \log_2(n)$

Therefore, total number of levels in above complete binary tree are $\log_2(n)$.

6. Assuming $n = 3^k$,
if we build a tree that each node has a value. with root having value n, and each node has 3 children, each having value of 1/3 of the parent node value…and so forth, like this:



if the minimum node value is 1, then what is the level of the tree.

Solution:-

Total amount of work done in each level is n,

So, we can conclude that,

$T(n) = 3T(n/3) + n \quad ; n>1$

$= 1 \quad ; n=1$

Therefore,

$T(n/3) = 3T(n/9)+ (n/3)$

total amount of work done = (n/number of nodes at that level)

$= (n/3^0) + (n/3^1) + (n/3^2) + \ldots\ldots + (1)$ (where $1=(n/3^k)$)

So, $1=(n/3^k)$

$n=3^k$ (as per given in question)

$k=\log_3 n$

$k+1=(\log_3 n+1)$ (as we have 0 to k = (k+1))

Therefore, number of levels in above tree are $(\log_3 n+1)$.