C.S.E 221 – lab 01

1. Naive approach:

```
bool prime[n] = {0};   O(1)
void isprime(int n)
{
  for(int i=2; i<=n; i++){
       int cnt = 0;   O(1)
       for(int j=2; j<i; j++){
       if (i%j == 0) cnt++ (O(1)
   }        O(1)
   if(cnt==0) prime[i] = 1;  O(1)
  }        O(1)
O(n) for(int i=2; i<=n; i++){
   if (prime[i]) O(1)
      System.out.print(i + " ");   O(1)
  }
}
```

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

$\frac{n-2+1}{1} = n-3 \Rightarrow O(n-3) = O(n)$

$\therefore$ Time complexity $= O(n^2) + O(n) + O(1)$

$= O(n^2)$

2. Optimal sieve:

```
void sieveofEnathosthenes (int n){
  boolean prime[] = new boolean [n+1]; O(1)
  for (int i=0; i<n ; i++)        O(n)
        prime[i] = true;   O(1)
  for (int P=2 ; P<= sqrt(n) ; P++){      √n-2+1
                                          = O(n)
     if (prime[P] == true) {  O(1)
        for(i = P*P ; i<=n ; i+= P)      O(logn)
           prime[i]=false;
     }
  }
   for (int i=2 ; i<=n ; i++) {  n-2+1
       if (prime[i] == true) O(1)     => O(n)
         system.out.print (i+" "); O(1)
   }
```

O(n) * O(logn) => O(nlogn)

∴ worst case time complexity = O(n log n)

CSE 221: Lab01 - Assignment 1

Recursion Tree Time Complexity -

~~3) $T(n) = T(n/3) + 2T(n/3) + n$~~

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

$T(1) = 1$

Now, $T(n) = T(n/2) + n-1$

$\Rightarrow T(2^m) = T(2^{m-1}) + 2^m - 1$

$\Rightarrow T(2^{m-1}) = T(2^{m-2}) + 2^{m-1} - 1$

$\Rightarrow T(2^{m-2}) = T(2^{m-3}) + 2^{m-2} - 1$

- - - - - - - - - - -

$\Rightarrow T(2^2) = T(2^1) + 2^2 - 1$

$\Rightarrow T(2) = T(2^0) + 2 - 1$

$\Rightarrow T(2^0) = T(1) + 2^0 - 1$

$= 0$

Adding all of these,

$$T(2^m) = 2^{m+1} - m$$
$$= 2^1 . 2^m - m$$
$$= 2n - \log_2 n \quad \left[\because \begin{array}{l} 2^m = n \\ m = \log_2 n \end{array}\right]$$
$$= 2n$$
$$\approx O(2n)$$
$$\approx O(n)$$

$\therefore T(n) \approx O(n)$ (Ans)

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

$T(1) = 0$

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

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

— — — — — —

$\Rightarrow T(3) = T(2) + 3 - 1$

$\Rightarrow T(2) = T(1) + 2 - 1$

$\Rightarrow T(1) = 0 + 1 - 1$

$= 0$

Adding all of these,

$$T(n) = n + (n-1) + (n-2) - - - - + 2 + 1 + 0 - (1 \times n)$$

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

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

(Ans).

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

$$= 3\left[3T(n/3^2) + n/3\right] + n$$

$$\left[\because T(n/3) = 3T\, n/3^2 + n/3\right]$$

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

$$= 3^2\left[3T(n/3^3) + n/3^2\right] + 2n.$$

$$\left[\because T(n/3^2) = 3T(n/3^3) + (n/3^2)\right]$$

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

$$- - - - -$$

$$= 3^k T(n/3^k) + kn.$$

Now, $T(n/3^k) = T(1)$

$$\Rightarrow n/3^k = 1 \Rightarrow 3^k = n$$

$$\therefore k = \log_3 n.$$

$T(n) = 3^k T(1) + kn$

$= n \times 1 + \log_3 n \cdot n$

$= n + n \log_3 n$

$= O(n \log_3 n)$ (Ans.)

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

$= 2[2T(n/2) + n^2] + n^2$

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

$= 2^2 [2T(n/2^3) + n^2] + 3n^2$

$= 2^3 T(n/2^3) + 5n^2$

$= 2^k T(n/2^3) + 5n^2$

$= 2^k T(n/2^k) + (2k - 1)n^2$

$= 2 \log_2 n + (2 \log_2 n - 1) n^2$

$= n + 2n^2 \log_2 n - n^2$

$= n^2$

$\approx O(n^2)$ (Ans)

Pseudocode to coding:

```
import Java.Util.Scanner;
public class lab-01{
  public static void main(String []args){
        Scanner sc= new Scanner(System.in);
        int n= sc.nextInt();
        int a=n,sum=0;
    while(n>0){
           int r=n%10;
           sum= sum+r*r*r;
           n= n/10;
        }
        if(a== sum){
           System.out.printlen("Armstrong Number");
        }
      else{
        System.out.printlen("It is not an Armstrong
                          Number");
         }
        }
      }
```