Exactly3Divisors

Time Complexity: O(N^1/2 * N^1/4)
Space complexity : O(1)

```
class Solution
{
   public boolean isPrime(int n)
    {
        if(n<=1)
            return false;

        for(int i=2;i<=Math.sqrt(n);i++)
            if(n%i==0)
                return false;

        return true ;
    }
    public int exactly3Divisors(int N)
    {

        int counter=0;
        N = (int)Math.sqrt(N);
        for(int i=1;i<=N;i++)
        {

            if(isPrime(i))
                counter++;
        }
        return counter;
    }
}
```

if number is perfect square or has sqrt as prime number then it has exactly 3 divisors

Factorial

Time Complexity : O(n)
Space Complexity : O(1)

```
int fact =1;
for (int i=2;i<=n;i++){
        fact = fact * i;
}
return fact;
```

Time Complexity : O(2^n)
Space Complexity : O(1)

```
if(n==0){
        return 1
}
return n * fact(n-1);
```

Gcd or Hcf

1st Approach

Time Complexity : O(log(min(n1,n2)))
Space Complexity : O(1)

```
while(n1>0 & n2>0){
        if(n1>n2){
                n1 =n1 % n2;
        }
        else{
                n2 = n2 % n1;
        }
}
if(n1==0) return n2;
return n1;
```

2nd Approach

Time Complexity : O(min(N1,N2))
Space Complexity : O(1)

```
for(int i=1 ; i< Math.min(n1,n2); i++){
        if(n1%i==0 & n2%i==0){
                gcd = i;
        }
}
return gcd;
```

Prime Number

1st Approach

Time Complexity : O(sqrt(N))
Space Complexity : O(1)

```
class Solution {

    public boolean isPrime(int n) {

        if(n < 2) return false;
                        int count = 0;
        for(int i = 1; i <= Math.sqrt(n); ++i) {
            if(n % i == 0) {
                count = count + 1;
                if(n % i != i) {
                    count = count + 1;
                     }
            }
        }
        if(count == 2) return true;
        return false;

    }
```

2nd Approach

Time Complexity : O(N^1/2 * N^1/4) = O(N^3/4)
Space Complexity : O(1)

```
public static boolean isPrime(int n) {
        // Handle small numbers
        if (n <= 1) return false;
        if (n <= 3) return true;
        if (n % 2 == 0 || n % 3 == 0) return false;
```

```
        // Start checking from 5 using 6k ± 1 optimization
        for (int i = 5; i * i <= Math.sqrt(n); i += 6) {
            if (n % i == 0 || n % (i + 2) == 0) return false;
        }
        return true;
    }
```

Prime Till N

1st Approach

Time Complexity: O(n^3/2)
Space Complexity : O(1)

```
class Solution
{
   public boolean isPrime(int n)
    {
    for(int i = 1; i <= Math.sqrt(n); ++i) {
         if(n % i == 0) {
                count = count + 1;
                if(n % i != i) {
                    count = count + 1;
                     }
            }
        }
            return count==2
     }
    public int exactly3Divisors(int N)
    {

        int counter=0;
        for(int i=1;i<=N;i++)
        {

            if(isPrime(i))
                counter++;
        }
        return counter;
    }
}
```