Math - Programs

1. **Prime Number**

1.1 BruteForce:

i = [1,N] => N%i==0 => count++;

count ==2 ? True : False

TC : O(N)

1.2 Optimized Solution:

i = [1, sqrt(N)] => N%i==0 => return False;

TC : O(sqrt(N))

```
boolean isPrime(int N){
    if(N<=1){
       return false;
    }
    for(int i=2;i<=Math.sqrt(N);i++){ // i*i <= N
       if(N%i==0){
           return false;
       }
    }
    return true;
}
```

1.3 Sieve of Esthones

TC : O(N log log N)

```
String sieveOfEratosthenes(int N){
   boolean[] isPrime = new boolean[N+1];
   Arrays.fill(isPrime, true);
   isPrime[0] = isPrime[1] = false;
   for(int i = 2; i * i <= N; i++) {
       if(isPrime[i]) {
           for(int j = i * i; j <= N; j+=i) {
               isPrime[j] = false;
           }
       }
   }

   return isPrime[N] ? "Yes" : "No";
}
```

2. **Armstrong Number**

A number is called armstrong number if sum of each digit raised to power of number of digits of number is equal to the number.

Eg: N = 153. It is Armstrong Number as 153 = 1 + 125 + 27

```
String isArmstrongNumber(int N){
   int len = String.valueOf(N).length();
   int sum = 0, num = N;
   while(num > 0){
       int r = num % 10;
       sum += (int) Math.pow(r, len);
       num /= 10;
   }
   return sum==N ? "Yes" : "No";
}
```

Time Complexity: O(N)

3. **GCD of two numbers**

3.1 Brute Force

Get min of a,b

Iterate until min-value > 0

```
while (result > 0) {
   if (a % result == 0 && b % result == 0) {
      break;
   }
   result--;
}
```

Time Complexity: O(min(a,b))

Space Complexity: O(1)

3.2 Euclidean method subtraction

```
int gcd(int a, int b)
{
   // Everything divides 0
   if (a == 0)
      return b;
   if (b == 0)
      return a;
   // Base case
```

```
    if (a == b)
        return a;
    // a is greater
    if (a > b)
        return gcd(a - b, b);
    return gcd(a, b - a);
}
```

Time Complexity: O(min(a,b))

Space Complexity: O(min(a,b))

3.3 Euclidean method division

```
int gcd(int a, int b)
{
    if (b == 0)
      return a;
    return gcd(b, a % b);
}
```

Time Complexity: O(log(min(a,b)))

Space Complexity: O(1)

Reference - https://www.geeksforgeeks.org/program-to-find-gcd-or-hcf-of-two-numbers/

4. **Closest Number Divisible by m**

```
public int getClosetDivisible(int n, int m){
    int q = n/m;
    int n1 = m*q;
    int n2 = (n*m)>0 ? m*(q+1) : m*(q-1);
    return Math.abs(n1-n)<Math.abs(n2-n)?n1:n2;
}
```

Time Complexity : O(1)

Space Complexity : O(1)

Reference - https://www.geeksforgeeks.org/find-number-closest-n-divisible-m/

5. **Is the number power of the other**

```
public boolean isPowerOfOther(int num1, int num2) {
    double res = Math.log(num2)/Math.log(num1);
    return res == Math.floor(res);
}
```

Time Complexity : O(1)

Space Complexity : O(1)

6. **Rectangles Overlap**

```
static class Point{
   int x;
   int y;
   public Point(int x, int y){
       this.x = x;
       this.y = y;
   }
}

boolean overlap(Point l1, Point r1, Point l2, Point r2 ){
   if(l1.x>r2.x || l2.x > r1.x){
       return false;
   }
   if(r1.y>l2.y || r2.y > l1.y){
       return false;
   }
   return true;
}
```

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