A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself (2, 3, 5, 7, 11, 13, 17, 19, 23)

1

```
boolean isPrimeBruteForce(int n) {
  for (int i = 2; i < n ; ++i) {
    ++steps ;
    if (n % i == 0) {
      return false ;
    }
  }
  return true ;
}

 void bruteForce() {
   for (int i = 2; i <= max ; ++i) {
     if (isPrimeBruteForce(i) == true) {
       p[pkount++] = i ;
     }
   }
 }
```

$$O(n^2)$$

2

```
boolean isPrimeUptoSquareRoot(int n) {
  If n is factorisable
        n = r * q
    r or q must be <= SQRT(n)

   n  SQRT(n)  (r * q)
  -----------------------
  25    5         (5 * 5)
  18    4.2       (3 * 6)
  24   4.8        (2 * 12)
}

 void uptoSquareRoot() {
     for (int i = 2; i <= max ; ++i) {
       if (isPrimeUptoSquareRoot(i) == true) {
         p[pkount++] = i ;
       }
     }
   }
```

$$O(n\sqrt{n})$$

3

```
void uptoPrimeNumbers() {
  int pkount = 0 ;
  p[pkount++] = 2 ;
  for (int i = 3; i <= max; ++i) {
    boolean divisible = false ;
    for (int k = 0; k <= sqrt(pkount); ++k) {
        //Check if divisible
    }
    if (divisible == true) {
      p[pkount++] = i ;
    }
  }
}
```

Note this

$$\frac{i}{\log i}$$

$$O\left(\frac{n\sqrt{n}}{\log n}\right)$$