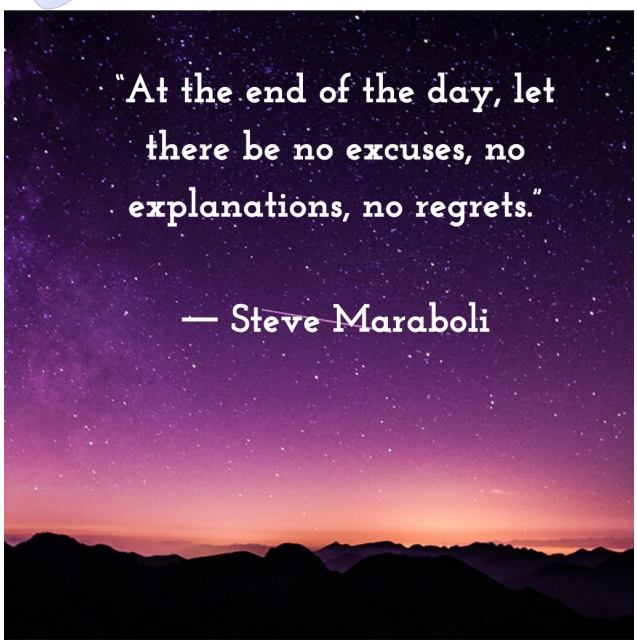
Prime Numbers



Today's content

- → Prime number Intro
- → Get all primes from 1 to N
- → Print smallest prime factor for 2 to N
- → Prime factorisation
- → Get the no. of factors/divisors
- → Sorted permutation Rank

Prime number → No. having only two factors

1 & itself

Eg: 2, 3, 5, 7, 11 ...

Q Given a no. we need to check if it is prime or not.

No. = 11 ⟶ True

No. = 14 ⟶ False

Ans = Count the no. of factors

if (count == 2) → prime

if (count > 2) → not prime

```
boolean checkprime (int n)
{
    count = 0
    for (i=1 ; i*i ≤ n ; i++){
        if (n % i == 0){
            // i & n/i are two factors
            if (i == n/i) count ++;
            else count += 2
        }
    }
    if (count == 2) return true
    else return false
}
```

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

SC : $O(1)$

Scenerio

SecurePrime Inc. a renowned cybersecurity firm, is on a mission to upgrade its **encryption** techniques to outsmart potential attackers. Their strategy involves utilizing **prime numbers**, known for their pivotal role in strengthening cryptographic systems

Problem

SecurePrime Inc. plans to enhance its **encryption keys** by incorporating **random prime numbers** from the **1 to A** into their algorithms. This approach significantly complicates any brute force attempts by attackers, making the encryption much more robust and reliable.

Task

As the requirement , you need to generate prime numbers** from the **1 to A**

N = 10 → 2, 3, 5, 7

N = 20 → 2, 3, 5, 7, 11, 13, 17, 19

Idea → Iterate from 1 to N & check if a number is prime or not.

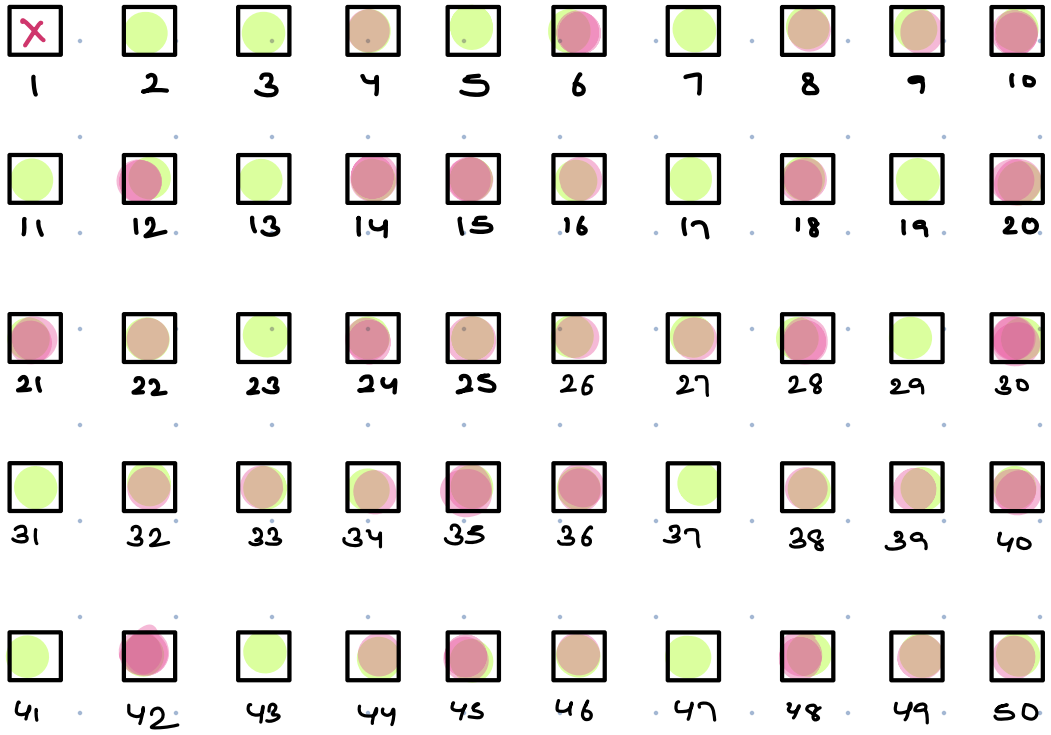
```
void printall (int n){
    for (i=1; i≤n; i++){
        if (checkprime(i) == true) print(i);
    }
}
```

TC: O(N√N)
SC: O(1)

* Idea 2 = Sieve of Eratosthenes

N = 50 ⇒ Find all prime no. from 1 to 50

Assume → Every no. is a prime no.



If 2 is a prime no., what all no. can never be prime

↳ Multiples of 2 will never be prime

Mark them as false

& we have to follow this process for each & every no. in this array.

```
void printallprimes (int n)
    boolean [] prime = new boolean[n+1]
    // Mark every no. as true:
    for (i=0; i≤n; i++){
        prime[i] = true;
    }
```

```
    prime[0] = false;
    prime[1] = false;

    for (i = 2 ; i*i<=n ; i++){
        if (prime[i] == true){
            for (j = i*i ; j<=n ; j=j+i){
                prime[j] = false;
            }
        }
    }

    for (i=2; i<=n; i++){
        if (prime[i] == true) print(i);
    }
}
```

TC: $O(N \log(\log N))$

SC: $O(N)$

i	Multiples of i & mark them as False
2	2*2 2*3 2*4 2*5 2*6 . . .
3	3*2 3*3 3*4 3*5 3*6 . . .
4	
5	5*2 5*3 5*4 5*5 5*6 . . .

Observation → For an element i, start marking multiples

false from i * i

*

$i\ [2 \to n]$

$j\ [i*i \to n]$

2 → $\frac{n}{2}$ iterations

3 → $\frac{n}{3}$

4 → ×

5 → $\frac{n}{5}$ iterations

.

.

.

$\sqrt{n}$ → $j = [(\sqrt{n})^2 \to n]$ 1 iteration

$\sqrt{n}+1$ → $j = [(\sqrt{n}+1)^2 \to n]$ 0 iterations

.

.

.

n → $j = [n^2 \to n]$ 0 iterations

0

0

TC : $\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} + \cdots$

$= n\left(\underbrace{\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \cdots}_{\text{sum of reciprocal of all prime no}}\right)$

$= n(\log(\log n))$

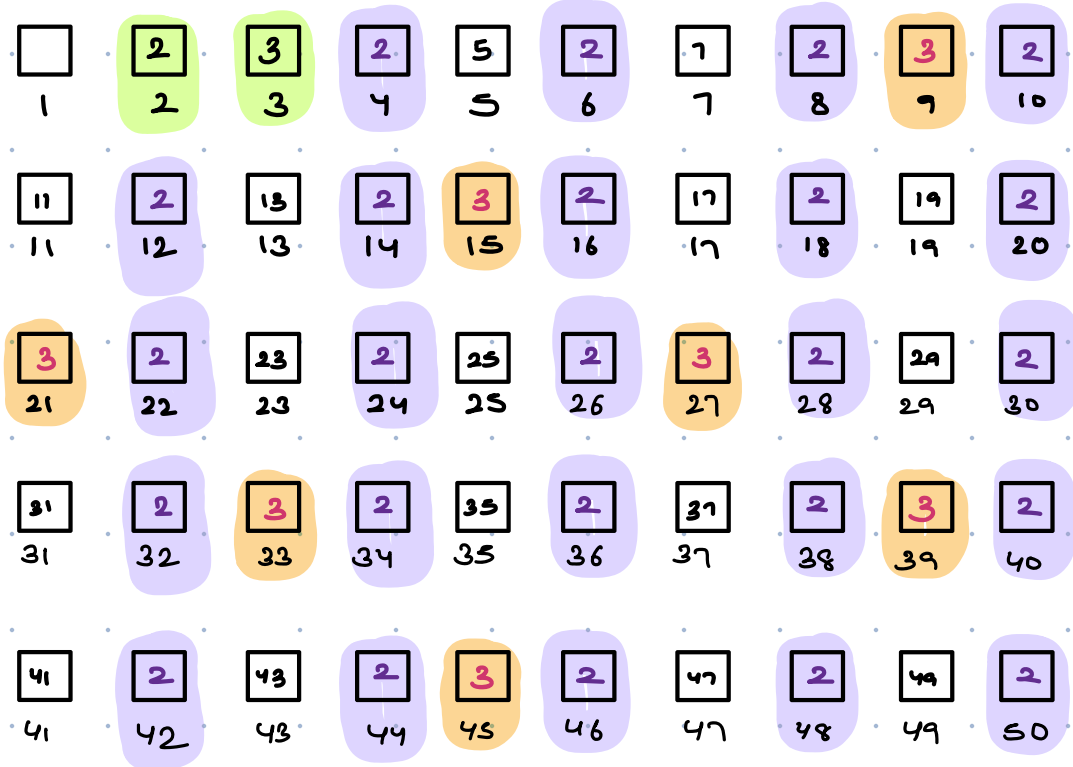
Segmented Sieve → [a b] → provide prime no. from a to b in better SC

Q3 Given N. Return the smallest prime for all no. from 2 to N

N = 10 → 2 3 4 5 6 7 8 9 10

Ans → 2 3 2 5 2 7 2 3 2

Assumption → Every no. is spf of itself



Observations → If (A[i] == i) → prime no.

→ If (A[i] != i) → this index is already updated & don't update it again.

```
int [] getspf (int n){
    int [] spf = new int [n+1];
    // all i, spf[i] = i
    for (i = 1; i <= n; i++){
        spf[i] = i;
    }
    for (i = 2; i*i <= n; i++){
        if (spf[i] == i){
            for (j = i*i; j <= n; j+=i){
                if (spf[j] == j){
                    spf[j] = i;
                }
            }
        }
    }
    return spf;
}
```

* Prime factorisation

n = 48

2	48
2	24
2	12
2	6
3	3
	1

$48 = 2 * 2 * 2 * 2 * 3$

$= 2^4 * 3^1$

No. of factors = $(4+1) * (1+1)$

$= 5 * 2 = 10$

48 = 1, 2, 3, 4, 6, 8, 12, 16, 24, 48

N = 300

2	300
2	150
3	75
5	25
5	5
	1

$300 = 2^2 * 3^1 * 5^2$

No. of factors = $(2+1) * (1+1) * (2+1)$

$= 3 * 2 * 3$

$= 18$

* Given a no. & its prime factorisation

$$n = i^a * j^b * k^c$$

No. of factors = $(a+1) * (b+1) * (c+1)$

Q Given a no. N, for all the elements from 1 to n, count the no. of factors

N = 10

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
	2	3	2	5	2	7	2	3	2
			4		3		4	9	5
					6		8		10
Ans → 1	2	2	3	2	4	2	4	3	4

Idea 1 → Create an array & initialise it with 1 (1 is a factor of every no.)

After that increase the count of that particular ele by 1 ($x \% x$), along with this go on every multiple of x & increase count by 1

TC : $O(N \log(\log N))$

Idea 2

$N = 360$

2	360
2	180
2	90
3	45
3	15
5	5
	1

$N = 360 \rightarrow 2^3 * 3^2 * 5^1$

No. of factors

HM

spf key		freq value
2	→	3
3	→	2
5	→	1

No. of divisors = $(3+1) * (2+1) * (1+1)$

$= 4 * 3 * 2$

$= \underline{24}$

Implement 1

```
int [] ans = new int [n+1]

for (i=1; i≤n; i++){
    ans[i]++;
    for (j=2i; j≤n; j+=i){
        ans[j]++;
    }
}
return ans;
```

TC: O(N (log N))
SC: O(1)

$$N + \frac{N}{2} + \frac{N}{3} + \frac{N}{4} + \cdots$$

$$N\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots\right)$$

$\log N$

Idea 2

```
int [] spf = getspf[n];
```
→ O(N log(log N))

```
for (i=1; i≤n; i++){
    HM<I, I> map = new HashMap<>();
    fill (spf, i, map)
```
→ $\log_2 i$

```
    ans=1
    for ( int key: map.keySet()){
        int val= map.get(key)+1;
        ans= ans*val;
    }
    print(ans);
}
```
O(1)

N * log i

```
public void fill ( int [] spf, int x, HM<I,I> map){
    while (x>1){
        // divide with spf[x]
        if ( hm.containskey (spf[x])
            of = hm.get(spf[x]);
            hm.put ( spf[x], of +1 );
        else { hm.put (spf[x], 1) ; }
        x = x/spf[x];
    }
```

Sorted permutation Rank

Given a string str. Find the rank of str in all the permutations formed in the lexicographical order.

str = "bac" = 3! = 6

str = "cab" = 5

a b c →
a c b
b a c → Ans = 3
b c a
c a b
c b a

str = "y o g i"

Rank = 3 * 3! +
2 * 2! +
0 * 1! +
0 * 0!

3 * 3! + 2 * 2! + 0 * 0!

= 18 + 4 = 22 + 1

↓

yogi

g - - - = 3!
i - - - = 3!
o - - - = 3!
y g - - = 2!
y i - - = 2!
y o g -
y o g i →

str = "abc"

a b c

0 * 2! + 0 * 1! + 0 * 0!

0 + 1

str = " r e d " n = 3

i → d

0 1 2

d _ _

e _ _

r d _

r e d

rank = 1 +

2 * [2]!

1 * [1!]

0 * [0!]

sum = 1 + 4 + 1 = 6

n-1-i

rank = count of smaller el on RHS * [n-1-i]!

str = " p e x t "

e _ _ _ ⇒ 3!

p e t _

p e x t

rank = 1 +

1 * 3! +

0 * 2! +

1 * 1! +

0 * 0!

ans = 8

```
rank = 1

long [] fact ;

fact [0] = 1

for ( i = 1 ; i ≤ n ; i++) {
    fact [i] = i * fact [i-1] ;
}
```

% modulo

```
for ( i = 0 ; i < str.length ; i++) {
    // count of smaller ele on RHS
    rank = rank + count * fact [n-1-i] ;
}

return rank ;
```

Explore this ↓

* Sorted permutation Rank with Repeat