# Today's Agenda

- Introduction to Prime Numbers
- Get all primes from 1 to N
- Print smallest prime factor for 2 to N
- Prime Factorization
- Get the number of factors/divisors

what are prime numbers?

↓ (N>0)

numbers having only 2 factors

1 → 1 ×

2 → 1, 2

5 → 1, 5

7 → 1, 7

11 → 1, 11

Ques. check prime ?

T.C → O(√N)

S.C → O(1)

```
count = 0;
for (i=1; i*i<=N; i++) {
    if (N%i==0) {
        if (i == N/i) {
            count += 1;
        } else { count += 2 }
    }
}
if (count == 2) {
    return True;
} else {
    return false;
}
```

Ques 2) Given a number N, print all prime numbers from 1 to N.

N = 10, 2, 3, 5, 7

N = 20, 2, 3, 5, 7, 11, 13, 17, 19

Brute force :-

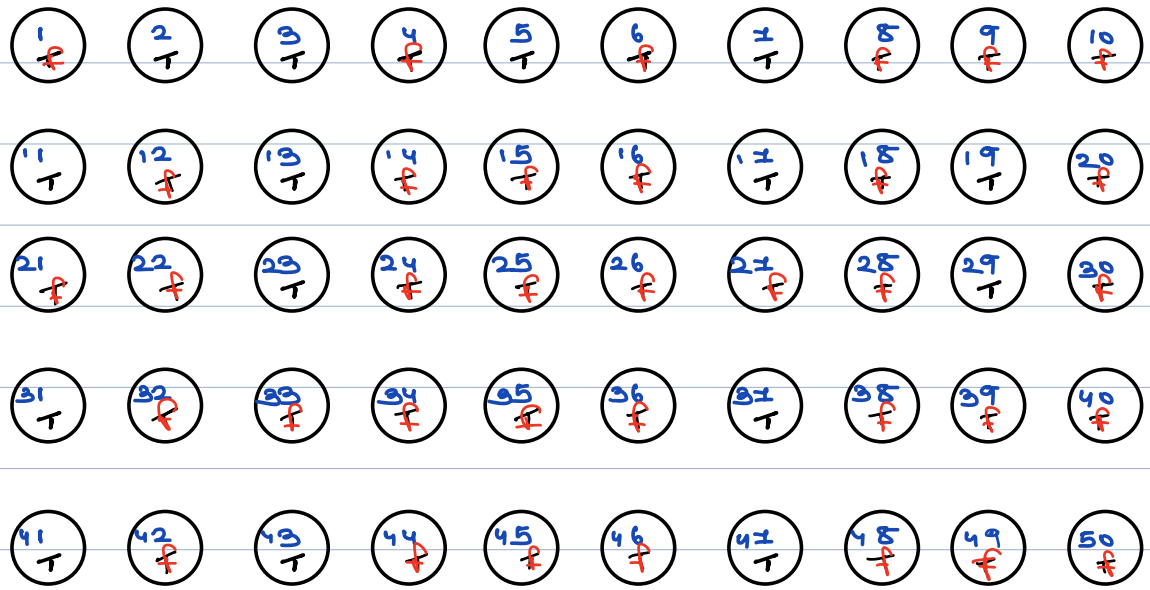
T.C → $N\sqrt{N}$

S.C → O(1).

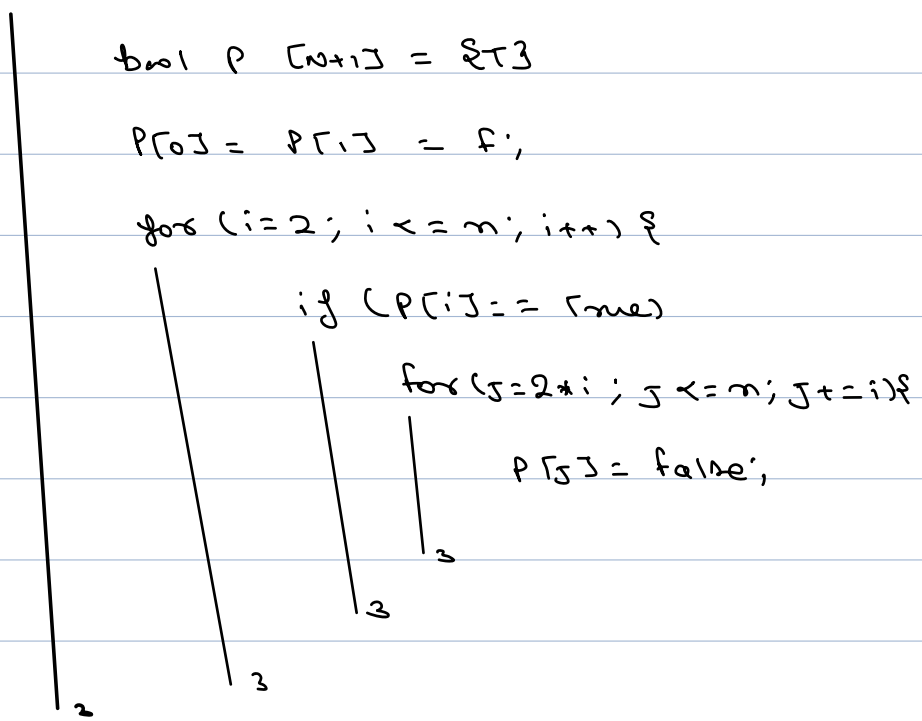
```
for (i=2; i<=n; i++) {
    if (checkPrime(i)) {
        print(i);
    }
}
```

Optimized Approach :-

N = 50, [1 50]




```
getallprimes(N){
    bool P[N+1] = {T}
    P[0] = P[1] = f;
    for(i=2; i<=n; i++){
        if(P[i]==true)
            for(j=2*i; j<=n; j+=i){
                P[j] = false;
            }
    }
}
```



2 → 4, 6, 8, 10 . . .

3 → 6, 9, 12 . . . .

5 → 5x2, 5*3, 5x4, 5x5

7 → 7*2, 7*3, 7*4, 7*5, 7*6, 7*7

11 → 11*2, 11x3, 11x4 - - - 11x11 . . .

Sieve of Eratosthenes.

```
getallprimes(n){                    ← correct code.
    bool p[n+1] = {T}
    p[0] = p[1] = f;
    for(i=2; i<=√n; i++){
        if(p[i]==true)
            for(j=i*i; j<=n; j+=i){
                p[j] = false;
            }
    }
}
```

n = 50

√50 = 7

1 to 50.

| outer loop | | inner loop |
|---|---|---|
| 2 | ~ | $n/2$ |
| 3 | ~ | $n/3$ |
| 5 | ~ | $n/5$ |

T.C → $\left(\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \dots\right)$

T.C → $(n \log(\log n))$

S.C → $O(n)$

Sum of reciprocals of all the primes.

$n = 2^{64} \simeq 10^{18}$.

| Previous idea | Current idea |
| --- | --- |
| T.C → $n\sqrt{n}$ | T.C → $O(n \log\log n)$ |
| $10^{18} \times \sqrt{10^{18}}$ | $10^{18}(\log \log 2^{64})$ |
| $\Rightarrow 10^{18} \times 10^{9}$ | $10^{18} \times \log 64$ |
| $\Rightarrow 10^{27}$ | $10^{18} \times 6$ |
| | $\Rightarrow 6 \times 10^{18}$ |

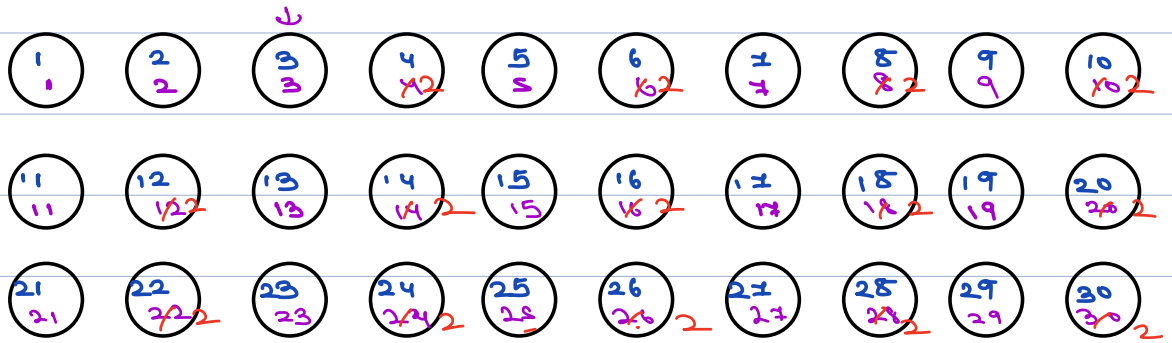
## Ques Smallest prime factors:-

**Given N, return the smallest prime factors for all numbers from 2 to N**

N = 10,

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| spf → | 2 | 3 | 2 | 5 | 2 | 7 | 2 | 3 | 2 |

for prime no. → spf will be no. itself.




T.C → O(n log log n), S.C → O(n)

```
— spf creation (n) {

    int spf[n+1]   // initialize spf[i] = i.
    P[0] = P[1] = 0;
    for (i=2; i<=√n; i++) {
        if (spf[i]==i)        // check if is prime
            for (j=i*i; j<=n; j+=i){
                if (spf[j]==j) {
                    spf[j]=i;
                }
            }
    }
```

# Prime factorization :-

↳ Representing a no' in multiples of powers of unique prime numbers.

| | |
|---|---|
| 2 | 48 |
| 2 | 24 |
| 2 | 12 |
| 2 | 6 |
| 3 | 3 |
| | 1 |

$n = 48 = 2^4 \times 3^1 \Rightarrow (4+1) \times (1+1) \Rightarrow 10$

1, 3, 5, 9, 15, 45

$n = 45 \Rightarrow 3^2 \times 5^1 \Rightarrow (2+1) \times (1+1) = 6$

$n = 300 \Rightarrow 2^2 \times 3^1 \times 5^2 \Rightarrow (2+1) \times (1+1) * (2+1)$
$\Rightarrow 18$

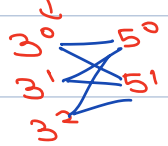
if you have a number n, whose prime factorization is,

$p_1^{a_1} \times p_2^{a_2} \times p_3^{a_3} \times \ldots p_y^{a_y}$

no of factors $\Rightarrow (a_1+1) * (a_2+1) \ldots (a_y+1)$

$n = 45 \Rightarrow 3^2 \times 5^1 \Rightarrow (2+1)(1+1) = 6$

1, 3, 5, 9, 15, 45



## Ques

**Given a number N. For all the numbers from 1 to N, get the number of factors/divisors**

N=10. →

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 2 | 4 | 2 | 4 | 3 | 4 |

Brute force:- for all numbers from 1 to n, find count of factors by $\sqrt{n}$ method.

T.C ⇒ $O(n\sqrt{n})$

Optimized.

⇒ n = 49. $\frac{49}{7}^{7} \Rightarrow \frac{7}{7}^{1} \Rightarrow 1$

7 → 7 x 2

no. of factors = 3.

n = 48 $\frac{48}{2}^{24} \Rightarrow \frac{24}{2}^{12} \Rightarrow \frac{12}{2} \Rightarrow \frac{6}{2}^{3} \Rightarrow \frac{3}{3} 1$

2 → 2 x 2 x 2 x 2

3 → 1

⇒ 10 factors.

n = 54 $\frac{54}{2}^{27} \Rightarrow \frac{27}{3}^{9} \Rightarrow \frac{9}{3}^{3} \Rightarrow \frac{3}{3} 1$

2 → 1

3 → 3 x 3 x 3

⇒ no. of factors = 8.

// create the spf array → $n \log \log n$

```
for (i = 2; i <= n; i++) {
    mm<int, int>
    x = i;
    while (x > 1) {
        if (spf[x] is in mm) {
            mm[spf[x]] += 1
        } else {
            mm[spf[x]] = 1
        }
        x = x / spf[x]
    }
    ans = 1;
    for (x : mm.keySet()) {
        ans = ans * (mm[x] + 1)
    }
    print(ans);
}
```

$\log(n)$

$\log n$

spf array.

T.C → $n \log(\log n) + n \log n$

S.C → $O(n) + O(\log n)$

Views.