Agenda :

Sieve of Erasthosthenes and its applications

Prime numbers ⟶

N

2 factors —— 1 and N

2 5 7 11 29 13 17

Q. Check if a number is prime number

⟶ find count of factors and if count is 2, it is prime

N = 3 Prime / True

N = 4 Not Prime / False

Approach :—

find count of factors and if count is 2, it is prime

N = 29 ⟶ 1 to 29

Can 30 be a factor for it?

Range : 1 to N

```
if (N == 1) { return false }
if (N == 2) { return true }
for ( i = 2 ; i < N ; i++) {
    if ( N % i == 0) {
        return false
    }
}
return true
```

Q. 11 ⟶ 1 & 11 = 2 factors ↓ Prime

N = 36

→ 1 to 36

1 and 36

2 to 35

N = 36

1 × 36
2 × 18
3 × 12
4 × 9
6 × 6 ---- LINE
9 × 4
12 × 3
18 × 2
36 × 1

} i x j

j x i

$\sqrt{N}$

1 to $\sqrt{N}$

N = 50

$\sqrt{50} = 7.1$

1 to 7.1 / 1 to $\sqrt{50}$

1 × 50
2 × 25
5 × 10 ↑ LINE
10 × 5
25 × 2
50 × 1

Count / Populate all my factors

C == 2 → Prime

C > 2 → NOT Prime

Pseudocode:

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

SC: $O(1)$

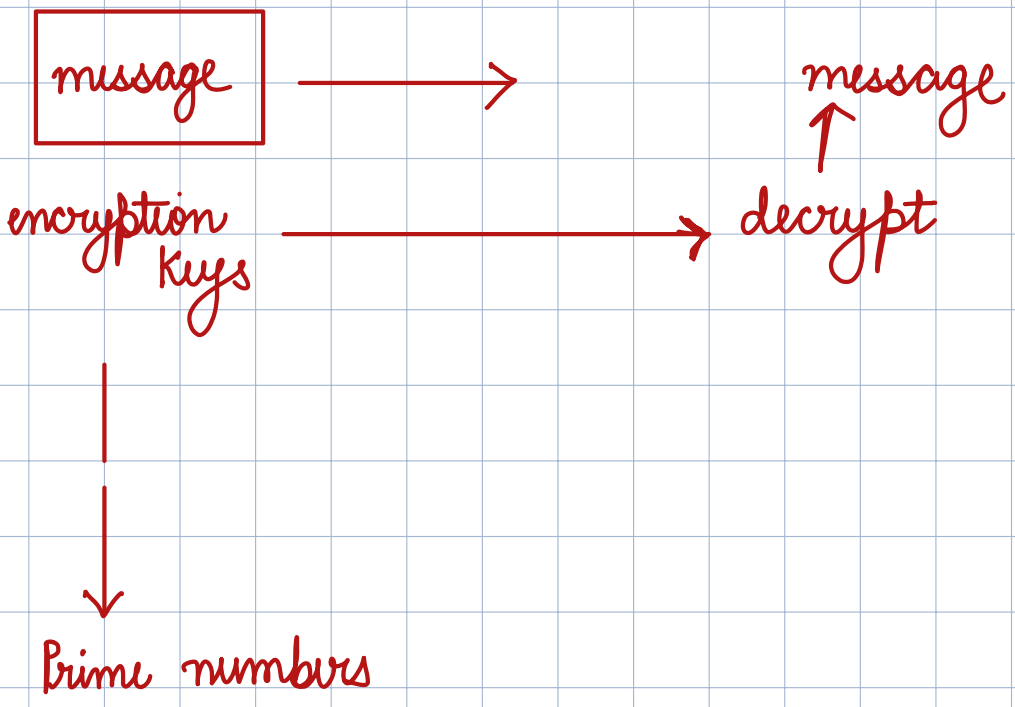
1 to $\sqrt{N}$

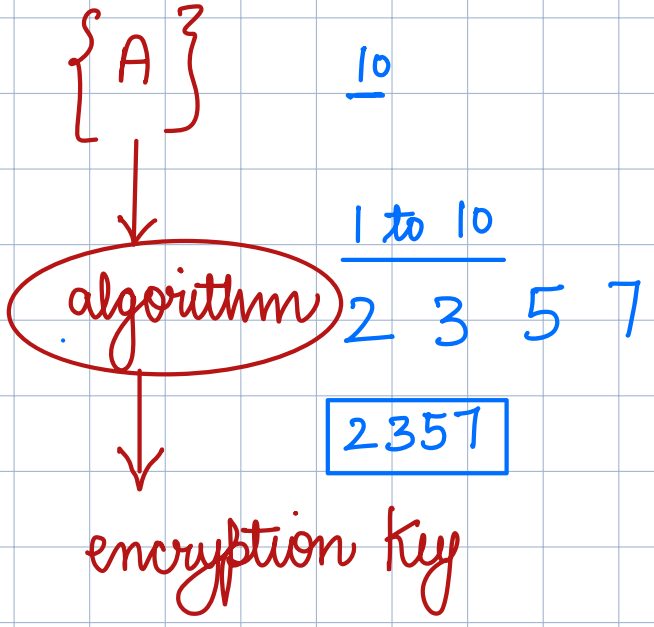
```
int c=0;
for( i=1    i*i <= N    i++) {
        if( N%i == 0) {
            if( i == (N/i)) {
                c = c+1
            }else {
                c = c+2
            }
        }
}
```

c

```
if( c == 2) {return true;}
else { return false }
```

Q2. Prime numbers ⟶ secure combination





Build best algo ⟶ to figure out what are the prime numbers 1 to A

↓ variable

A = 20

2 3 5 7 11 13 17 19

A = 10

2 3 5 7

---

for (j = 1 to A) {

N = j

int c = 0;

for (i = 1 i * i <= N i++) {

if (N % i == 0) {

if (i == (N/i)) {

c = c + 1

} else {

c = c + 2

}

}

}

TC: $A \times \sqrt{A}$

$O(A\sqrt{A})$

SC: O(1)

c

if (c == 2) {Print j ;}

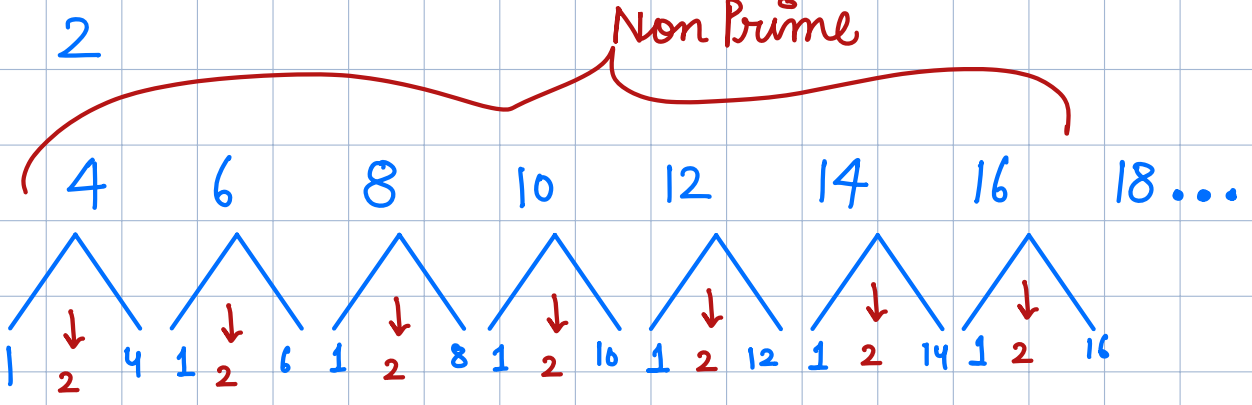
}

Sieve of Eratosthenes

Observation :

If 2 is a prime number

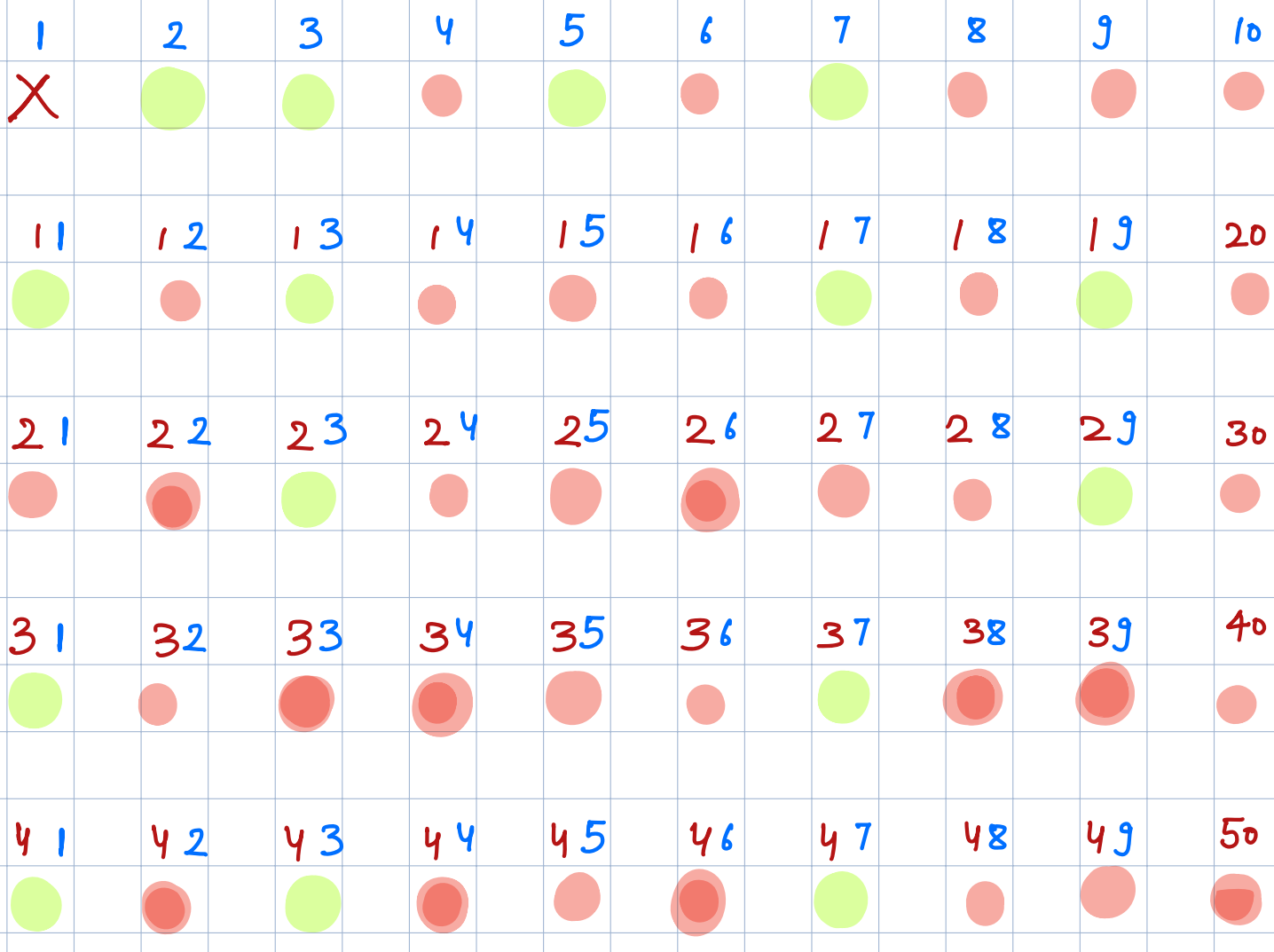
Its multiples can never be a prime number.



1 to A

A = 50

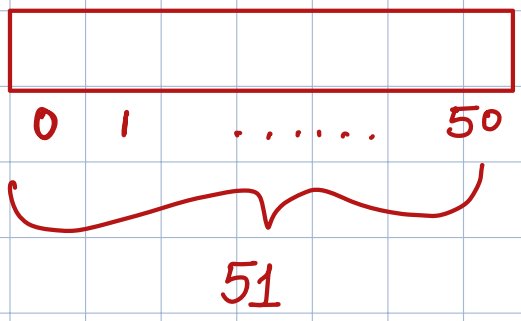
1 to 50 ⟶ find all prime no's in 1 to 50


1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
Prime
Not Prime

Create a big boolean array — 1 to N → N+1

1 to A = 1 to 50

A+1



F F T
0 1 2 ——— N
↑

green — Prime = True

2 — green (True)
- multiples of 2 - Red - False

3 — green (True)
- multiples of ~~2~~ 3 - Red - False

4 ← Red (ignore) False

5 — green (True)
- multiples of ~~2~~ 5 - Red - False

⋮

Pseudo Code:

boolean

isPrime [N+1] ·····> 0 to N

isPrime[0] = false
isPrime [1] = false

fill all values 2 to N = true ← in begin everyone isprime

```
for ( i=2 ; i<=N ; i++) {
    if ( isPrime[i] == true) {
        j = 2×i
        while ( j <= N) {
            isPrime[j] = false
            j = j + i
        }
    }
}

for ( i=0 ; i<=N ; i++) {
    if ( isPrime [i] == true) {
        Print(i)
    }
}
```

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

SC: $O(N)$

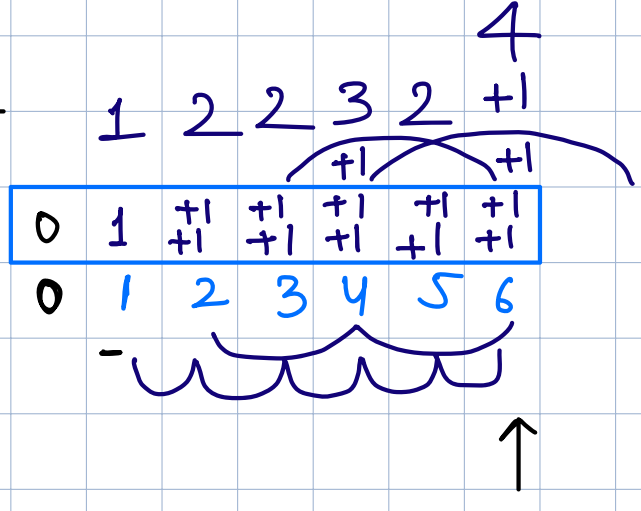
Count of Divisors

N=6

1	2	2	3	2	4
1	2	3	4	5	6

{1, 2, 3, 6}

N=6



Pseudo Code:

```
int isPrime [N+1]  ·····>  0 to N

isPrime[0] = 0
isPrime [1] = 0

fill all values 2 to N = 0

for ( i = 1 ; i <= N ; i++) {
{
    j = 2 x i
    while ( j <= N) {
        isPrime[j] += 1
        j = j + i
    }
}

isPrime
```

1 to N

0	1	2	3	4	5	6	7	8	9	10
0	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

Rank → {NAAGI} — Dictionary A, A, G, I, N

Sorted Permutation Rank

- CAB ✓

ABC
ACB
BAC
BCA
→ CAB 5th
CBA

Ans = 5

- PLAY

A L P Y — 3! = 6 words with A

L A P Y — 3! = 6 words with L

13 =

P ✓ L A Y → A _ _ — 2! = 2 with A

15 L _ A Y

A ↕ Y

15

PLAY ⟶

1. A L P Y

what is the rank of Play

15. PLAY

A (LPY)

L (PAY)

P L A Y

3

4-0-1

2 letters

$2 \times (3!)$

12 + 1 → 13th

12

LAY 3

$1 \times 2! = 2! = 2 + 1 \longrightarrow L$

4-1-1

2

A Y
↑

$0 \times 1! = 0$

0

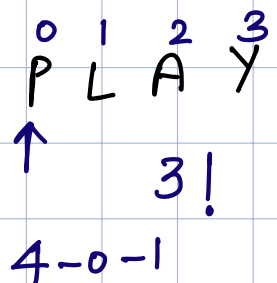
Y
↑
STOP

0

$(12 + 2 + 0 + 0) + 1$

P L A Y

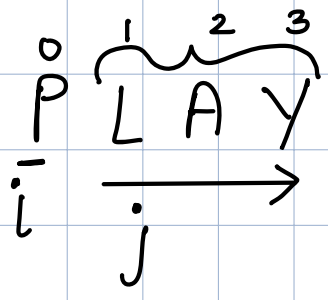
```
rank = 0
for ( i = 0    to   N-1 ) {
        ch = s[i]
        cnt = 0
        for ( j = i+1    j<N    j++ ) {
            if ( s[j] < ch ) { cnt++ }
        }
    rank = ( rank + cnt x fact[N-i-1] ) % M
}
return rank+1
```



TC: $O(N^2)$

SC: $O(N)$

```
fact[0] = fact[1] = 1
for ( i = 2    to   N ) {
        fact[i] = ( fact[i-1] * i ) % M
}
```



$S[i] = P = ch$

$S[j]$ ⟶