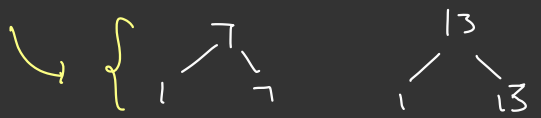
PRIME Numbers

Prime → exactly 2 divisors → 1 & N



Not Prime

Prime Check

$O(N)$

1 ——— N Count div

I)

```
cnt = 0
for(i=1 ——— N){
    if(N%i == 0){
        cnt++;
    }
}
```

II)

```
if(cnt == 2){
    Prime
}
```

3

(X, 2, 3, ---- N-1 X)

as soon as you atleast 1 div in the range (2, N-1)

break

Algo-2

O(N) in worst case

N = 7

7%2	NO
7%3	NO
7%4	NO
7%5	NO
7%6	NO

7 i=7

```
for(i=2, i<=N-1, i++){
    if(N%i==0){
        break;
    }
}
if(i==n){
    Prime
}
else{
    Not PRIME
}
```

i = 3

3

loop ends i==n

break the loop i<=n-1

2 3 4 5 6 7 (i==7)

i++

2 3 ---- 8 9

N = 9

2	9%2	No	
3	9%3	Yes	Stop

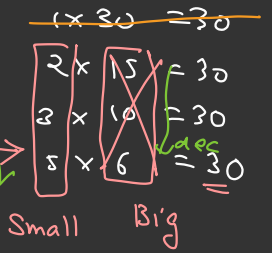
faster for non-prime No's

8

Algo -3 Root N optimisation

Examples –

N = 30 → 1, 2, 3, 5, 6, 10, 15, 30



Not Prime → any 1 divisor

Yes

$\sqrt{30} = 5.5xx$

$a == b$

$a^2 = N$

$\Rightarrow a = \sqrt{N}$

$a \cdot b = N$

$> \sqrt{N} \quad > \sqrt{N}$

$= > N$

$\Rightarrow$

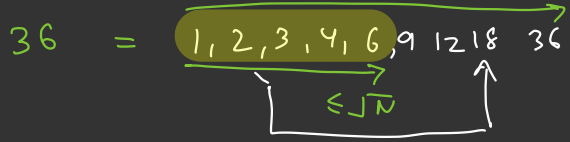
$a \times b = N$

$1 \times b = N$

$2 \times b = N$

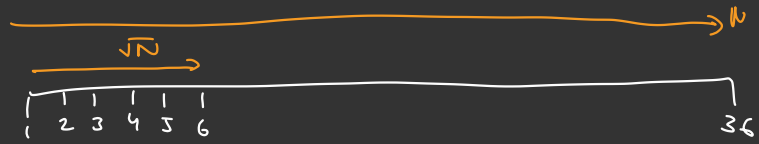
$\vdots$

$\sqrt{N}$



1 x 36 = 36
2 x 18 = 36
3 x 12 = 36
4 x 9 = 36
6 x 6 = 36

Pairs



$N = 10^4$

$\sqrt{N}$ = 100 steps

$N = 10^6$

$\sqrt{N}$ = 1000 steps

isPrime(n) {

```
isPrime = true;
for (i=2; i*i<=N ; i++ ){
    if (N%i==0) {
        isPrime = false;
        break;
    }
}
if (isPrime) → "Prime"
else → not Prime
```

$i * i \leq N$

$\Rightarrow i^2 \leq N$

$O(\sqrt{N})$

$O(1)$

}

N = 5	i=2	$2^2 \leq 5$	5 % 2 No	
	i=3	$3^2 \leq 5$	out of the loop	Prime
N=3	i=2	$2^2 \leq 3$	out of the loop	Prime
N=36	i=2	$2^2 \leq 36$	36 % 2 Yes	Not Prime
N=29	i=2	$2^2 = 4 \leq 29$	29 % 2	
	i=3	$9 \leq 29$	29 % 3	
	i=4	$16 \leq 29$	29 % 4	
$\sqrt{N}$	i=5	$25 \leq 29$	29 % 5	
Steps	i=6	$36 \leq 36$	out	[Prime]

Problems involving ⇒ Range of Prime Numbers

① Generate all Primes in Range 1 to N.

N = 100

2, 3, 5, 7, 11, ------- 97

Brute force

$O(N\sqrt{N})$

```
for(i=1, i<=N; i++){
    if(isPrime(i)){ print(i) };
```

3 ↑ $\sqrt{N}$

$d \longrightarrow \sqrt{N}$

Prime Sieve - Sieve of Eratosthenes.

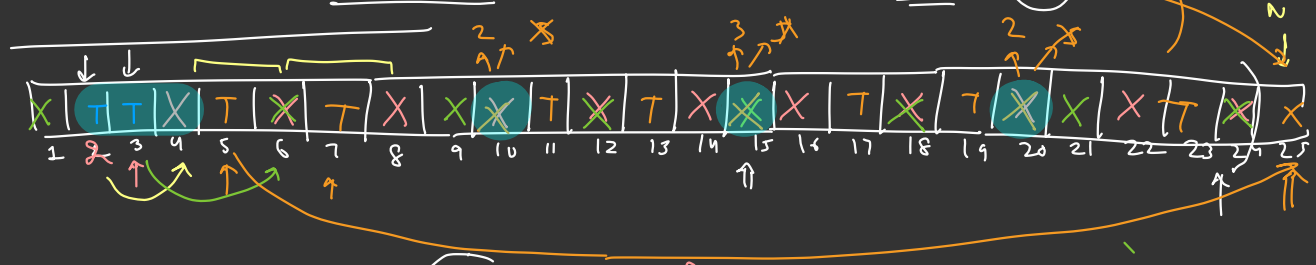
(famous technique)

{2, 3, 7, 11, 13, 17, 19, 23}

Prime (Yes)

Init ⇒ "all Prime."

boolean primes




```
for (i=2, i<=√N ; i++) {
    if (primes[i] = true){
        for (j=2*i ; j<=N, j=j+i){
            primes[j] = false ,
        }
    }
}
```

for prime no's

Mark all multiples of 2 as Not Prime

4, 6, 8, 10, --

j = ↑ ↑ ↑

2 → 4
3 → 6, 9, 12
4 → 8, 12, 16

i = 2
i = 3
i = 4

3 → 15 : 3 → 6, 9, 12, 15, --
5 → 15 : 5 → 10, 15, 20, 25, --

optimise it

$j = i^2$

```
Print-
for(i=2 ___ N){
    if(isPrime(i)) print i
}
```

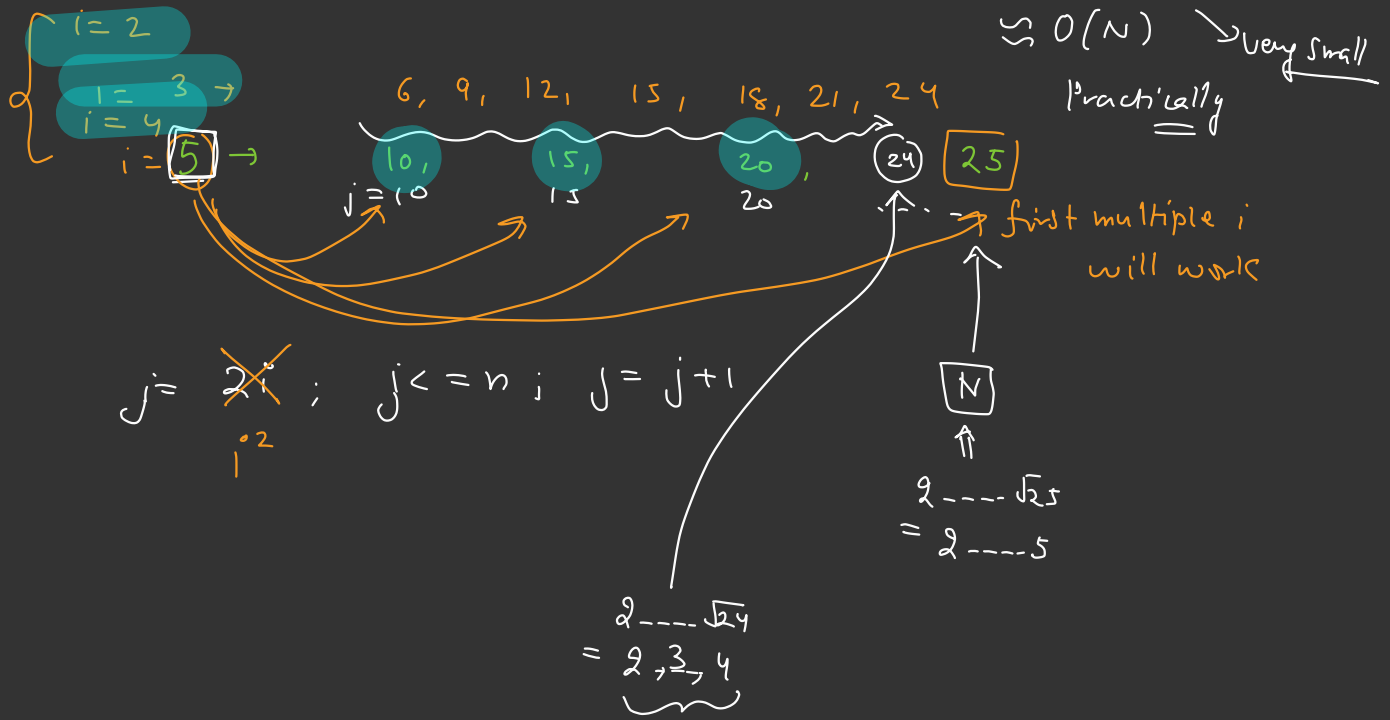
2 ___ N

Work = $\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \cdots + \frac{N}{p}$ (+ i+, up to $+\sqrt{N}$)

3, 5, 7, 9, 11, 13, 15, 17, ... ≤ N

$= O(N \log\log N)$

largest prime ≤ $\sqrt{N}$


i = 2
i = 3
i = 4
i = 5
6, 9, 12, 15, 18, 21, 24
10, 15, 20, 24
25
j = 10
15
20
first multiple i will work
j = i^2 ; j <= n ; j = j+1
N
2 ---- √25
= 2 ---- 5
2 ---- √24
= 2, 3, 4
≈ O(N)
Very small
Practically
=

```
primes[N+1] = true,

for (i = 2, i <= √N, i++) {

    if ( isPrime[i] ) {
        for (j = i^2; j <= N; j = j+i) {
            isPrime[j] = false;
        }
    }
}

for(i = 2 ——— N){
    if (isPrime(i)) { Print(i) };
}
```

(Q) given Q Queries, of the form [a,b] find out the count of primes in range [a,b]. $a, b \leq 10^6$

optimised Algo →

Q = 5

a b

[3, 10]

[10, 100]

[5, 20]

[2000, 10000]

[1500, 3500]

① Precompute sieve + prefix array — N log log N + N

② for each query. iterate over 3 to 10 and count (b − a) ✓

prefix[i] = count of primes till i

Q(a,b) = prefix[b] − prefix(a−1) → O(1) per query

Overall: N log log N + Q — done

Prime Factorisation

10.25

↳ No = Product of Powers of Primes.

$48 = 2^4 \cdot 3^1$

$51 = 3 \cdot 17$

$20 = 2, 2, 5$

2	48
2	24
2	12
2	6
3	3
	1

Algorithm - 1

```
for (i=2; i<=n; i++) {
    if (n%i == 0){
        cnt = 0
        while(n%i == 0){
            n = n/i
            cnt = cnt + 1;
        }
        print (i, cnt);
    }
}
```

$2^1 \cdot 3^3$

2	54
3	27
3	9
3	3
	1

N

(2,1) (3, 3)

Worst case

no is prime → O(N)

n = 11

2x, 3x, 4x, 5x, 6x, 7x, 8x, 9x, 10x, 11✓

4 steps.

11/11 = 1

N			
i = 2	27	54	yes
i = 3	1	27	cnt = 1
		yes	cnt = 0
		9	cnt = 1
		3	cnt = 2
		1	cnt = 3
i = 4	1		

n = 72

i = 2

36 cnt = 1

18 cnt = 2

9 cnt = 3

$2^3 \cdot 3^2$

$= 8 \times 9$

$= 72$

i = 3

n = 9

3 cnt = 1

1 cnt = 2

i = 4

n = 1

Stop

Algo-2

optimised

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

Not True $\sqrt{N}$ WHY ?

$4 \le \sqrt{11}$

```
for (i=2; i<=n; i++) {
    if (n%i == 0){
        cnt = 0
        while(n%i == 0){
            n = n/i
            cnt = cnt + 1;
        }
        print (i, cnt);
    }
}
if(n>1) { print(n, 1); }
```

outside for loop

$\sqrt{11}$ prime 1

$i \le \sqrt{N}$ N

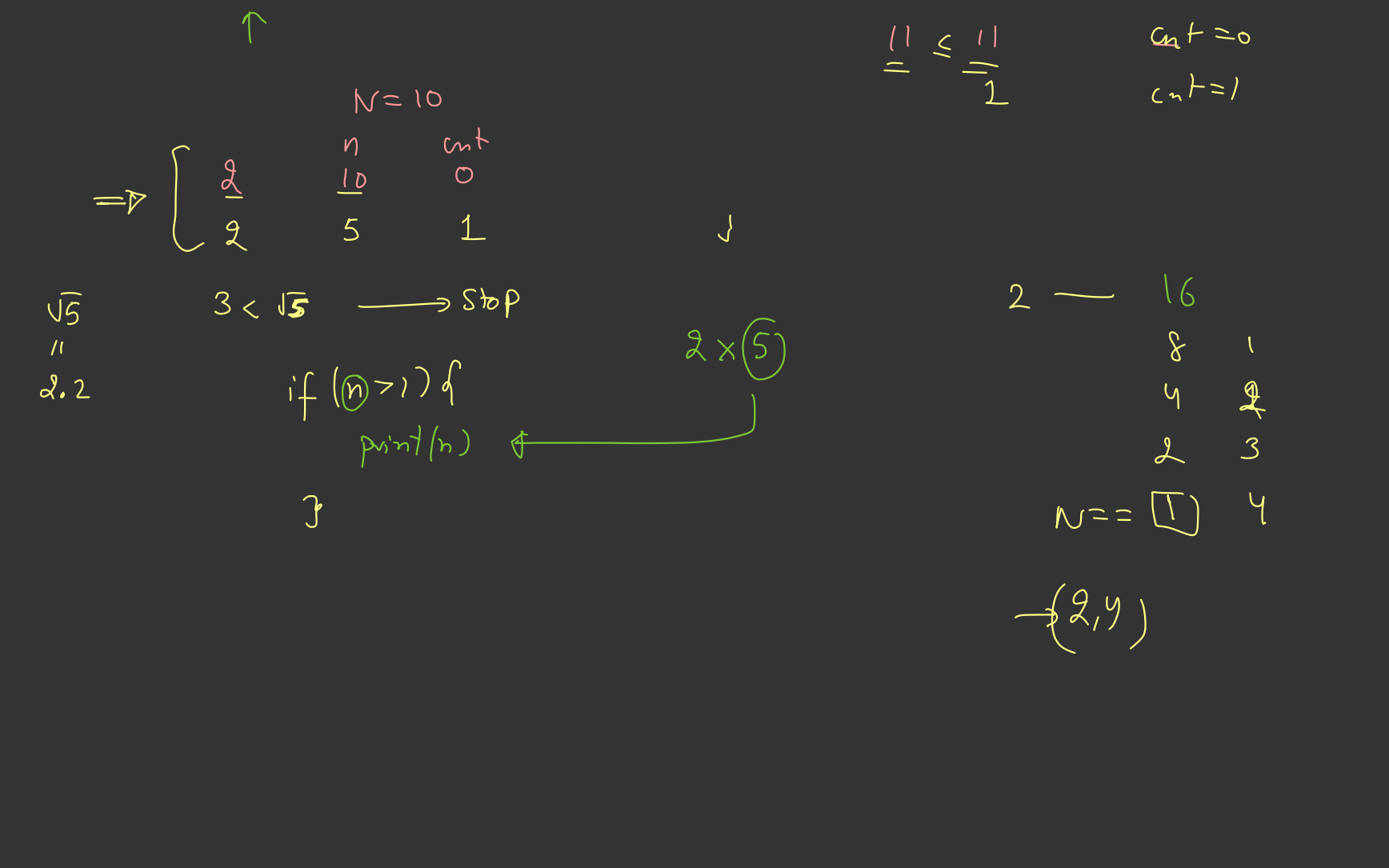
N = 44 $= 2^2 \cdot 11$

i	n	cnt
2	44	0
2	22	1
2	11	2
3	11	—
4	11	—
5	11	—
6	11	—
7	11	—
8	11	—
9	11	—
10	11	—

Stop at $\sqrt{N}$

cnt = 4 x 11 = 44

must be prime no as we couldn't find any more after $\sqrt{N}$


N = 10
n cnt
2 10 0
2 5 1
3 < √5 → stop
√5 = 2.2
if (n > 1) {
print(n)
}
2 × 5
11 ≤ 11
1
cnt = 0
cnt = 1
2 — 16
8 1
4 2
2 3
N == 1 4
(2, 4)

$5^2 \leq 55 \Rightarrow 25 \leq 55$

$6^2 \not\leq 36 \leq 11$

$\boxed{6^2 \leq 11}$

for (i=2, $i^2 <= 11$; i++){

Stop

3

if (n>1){

print(11,1) ;

3

i=2

i=3

i=4

i=5

~~i=6~~

i=6

N = 110

110	cnt=0	
55	cnt=1	(2', 5', 11')
55	x	= 2x5x11
55	x	= (110)
55	cnt=0	
11	cnt=1	
→ (11)	x	No $\sqrt{11} = 3$

↑ Prime No

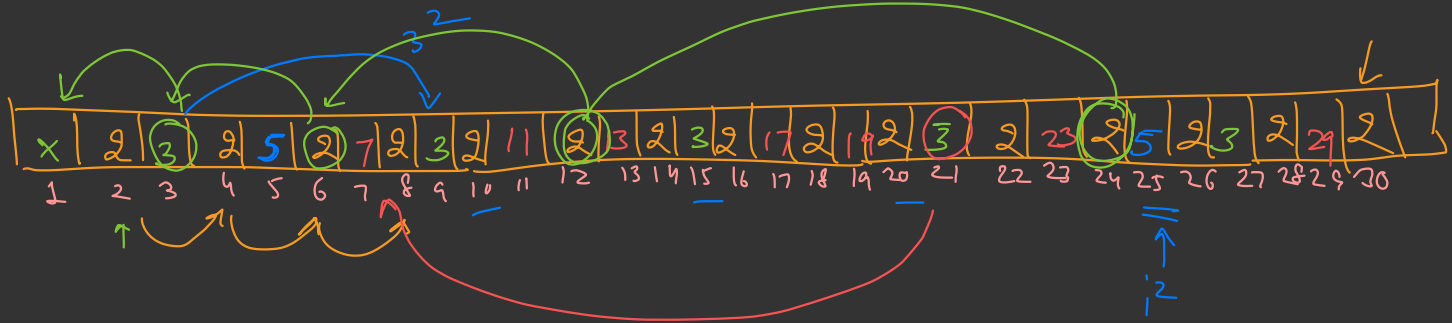
$\sqrt{N}$ → single

N $\sqrt{N}$ → N inputs

Prime Factorisation for a Range

for any 'no' store the smallest p.f

N = 30



1. Build a sieve storing the smallest pf at a every i
2. Prime factorisation

```
no = input()
while(no > 1){
    print(pfa[no]),
    no = no / pfa[no];
}
```

3

N = 24

24 /2

12 /2

6 /2 → 3 /3 → 1 stop

$2^3 \cdot 3 = 24$

21 → 3

1

Sieve → $N \log\log N + N \log_2 N$ (for N Numbers)

$21 = 3 \times 7$

100 → 13 → ... 1
(100/7)

100 → 50 → 25 → 12 → 6 → 3 → 1

X
- X/6
- X/7
- X/8

} by same No 7, 2 in each step.

Applications

=) Count the no divisors of 24

& divisors

1, 2, 3, 4, 6, 8, 12, 24

$\sqrt{N}$ Brute force

log N

$24 = 2^3 \cdot 3^1$

$\| (0,1,2,3) \quad (4) \cdot (2) = 8$

2 x 3 = 6

2 x 1 = 2

1 x 3 = 3

8 x 3 = 24

8 x 1 = 8

$N = p1^{a_1} \cdot p2^{a_2} \cdots pk^{a_k}$

divisors = $(a_1+1)(a_2+1)\cdots(a_k+1)$

0 to a_1 ~~a to a2~~

formula

$(3+1)(1+1) = 4 \times 2 = 8$ ways

$56 = \rightarrow 2^{3} \cdot 7^{1}$ (Prime factorisation)

$= 2^0 \cdot 7^0 = 1$

$2^1 \cdot 7^0 = 7$

$2^2 \cdot 7^0 = 2$

$2^3 \cdot 7^0 = 14$

$2^2 \cdot 7^1 = 4$

$2^3 \cdot 7^1 = 28$

$2^1 \cdot 7^1 = 8$

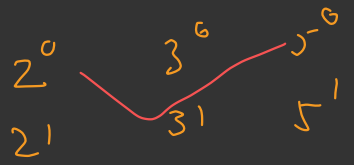
$2^2 \cdot 7^2 = 56$

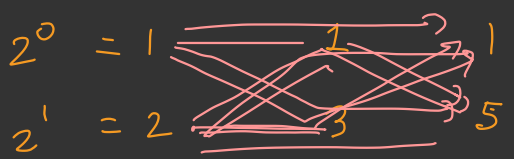
5 x 3 x 2

$30 = 2^1 \cdot 3^1 \cdot 5^1$

$= (1+1)(1+1)(1+1)$

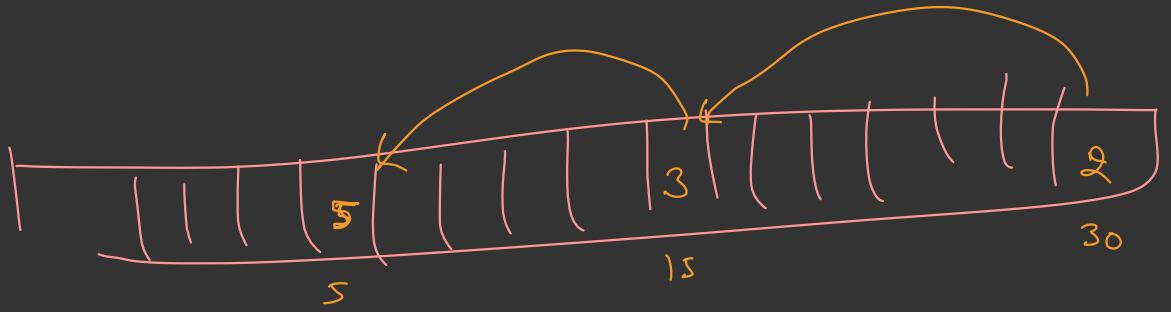
$= 8$





$= 1, 3, 15, 5$

$= 30, 10, 6, 2$



$30 = 2 \times 3 \times 5$