Number Theory

Euclid's Algorithm

GCD (Greatest Common Divisor)
or
HCF (highest common factor)

$32 = 2 * 2 * 2 * 2 * 2$

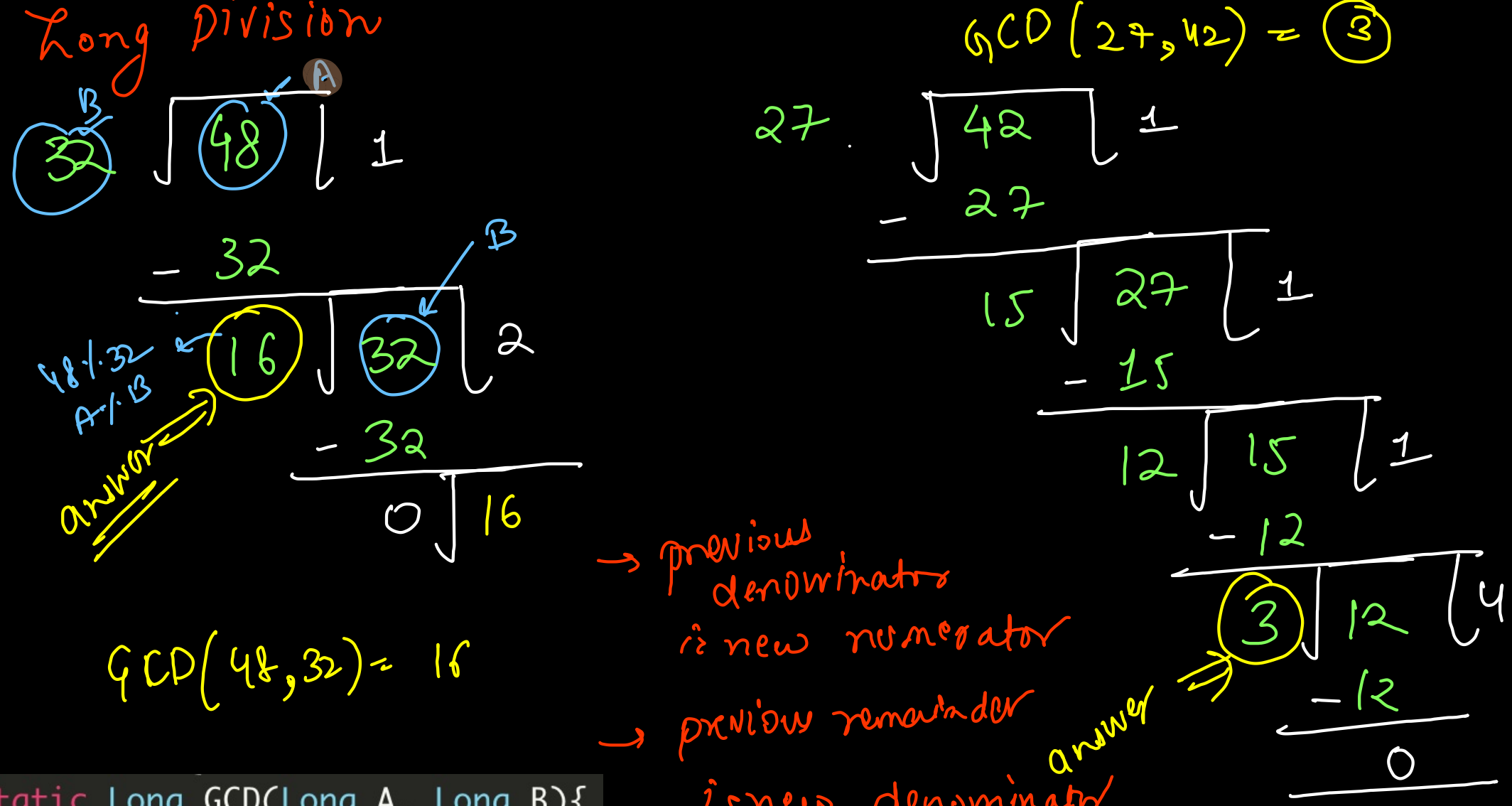
$48 = 2 * 2 * 2 * 2 * 3$

$GCD(32, 48) = 16$

$30 = 2 * 3 * 5$

$75 = 5 * 3 * 5$

$GCD(30, 75) = 15$




```
static Long GCD(Long A, Long B){
    if(B == 0) return A;
    return GCD(B, A % B);
}
```

Lowest Common multiple (Lcm)

$32 =$

$48 =$

$$Lcm = \frac{32 * 48}{GCD(32, 48) = 16} = \frac{\overset{2}{\cancel{32}} * 48}{\cancel{16}}$$

$$= 96$$

$2 * 3 = 6$

$3 * 9 = 27$

#1 $$a * b = GCD(a, b) * Lcm(a, b)$$

```java
class Solution {
    static Long GCD(Long A, Long B){
        if(B == 0) return A;
        return GCD(B, A % B);
    }

    static Long[] lcmAndGcd(Long A , Long B) {
        Long gcd = GCD(A, B);
        Long lcm = (A * B) / gcd;
        return new Long[]{lcm, gcd};
    }
}
```

Time → O(log(min(a,b)))

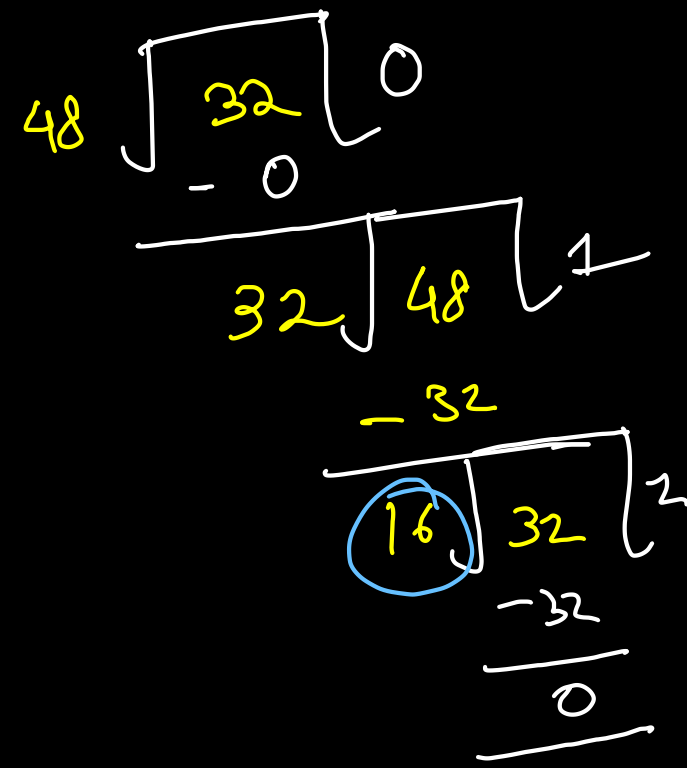
Long gcd = GCD(A, B); → O(1) const

Long lcm = (A * B) / gcd; → O(1) const

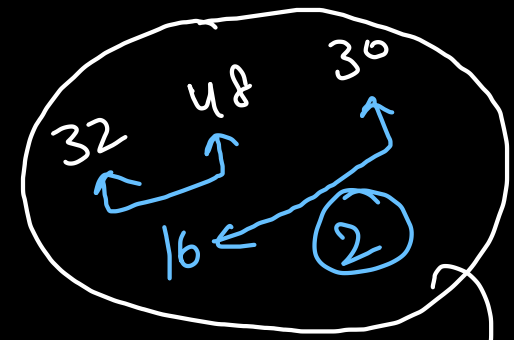
num < den

a < b

GCD (32, 48) = ?



Properties : GCD



① Symmetric $\quad gcd(a,b) = gcd(b,a)$

② $\quad gcd(a,0) = 0$

$\quad gcd(a,1) = 1$ ← min gcd

③ $\quad gcd(a,b) \Rightarrow$ max GCD possible $\Rightarrow \min(a,b)$

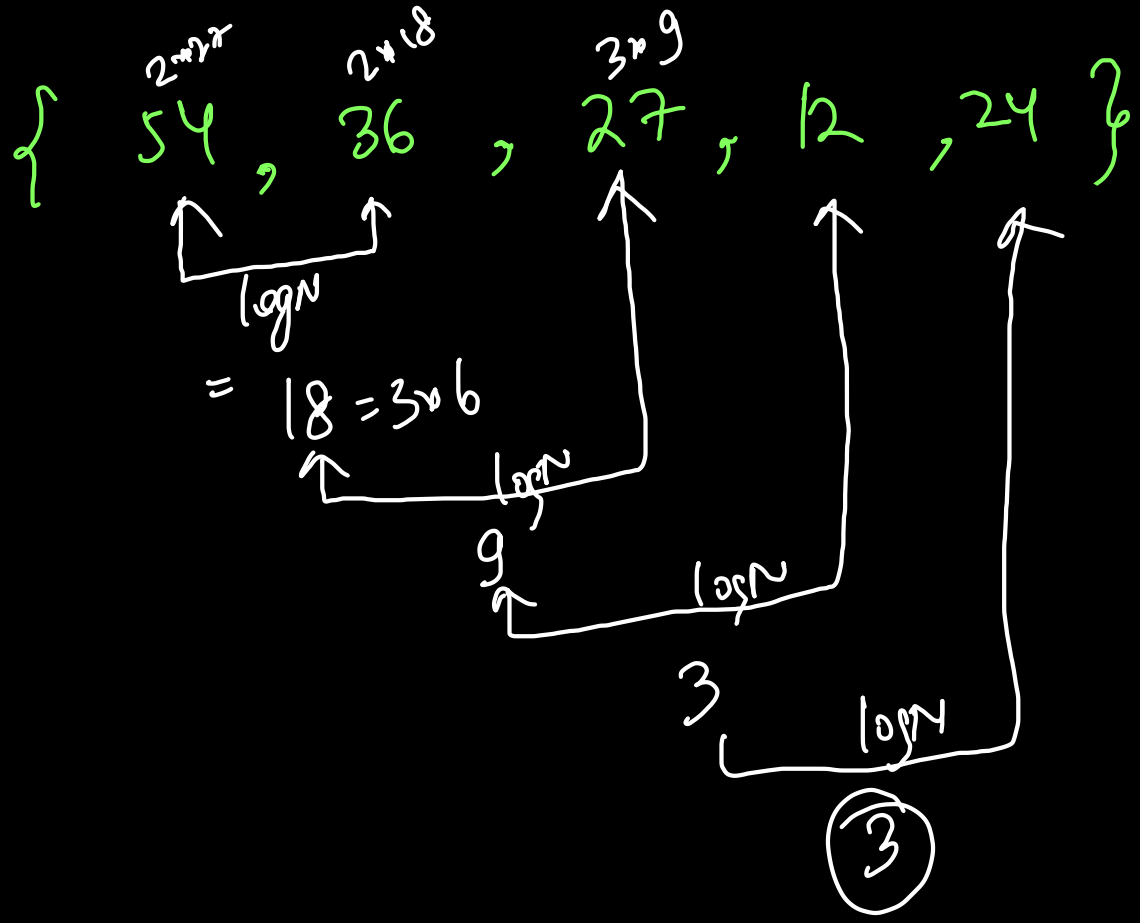
④ $\quad gcd(a,b,c) =$

$\quad gcd(\,gcd(a,b)\,, c)$

eg $\quad gcd(32,64)$

$\quad = 32$

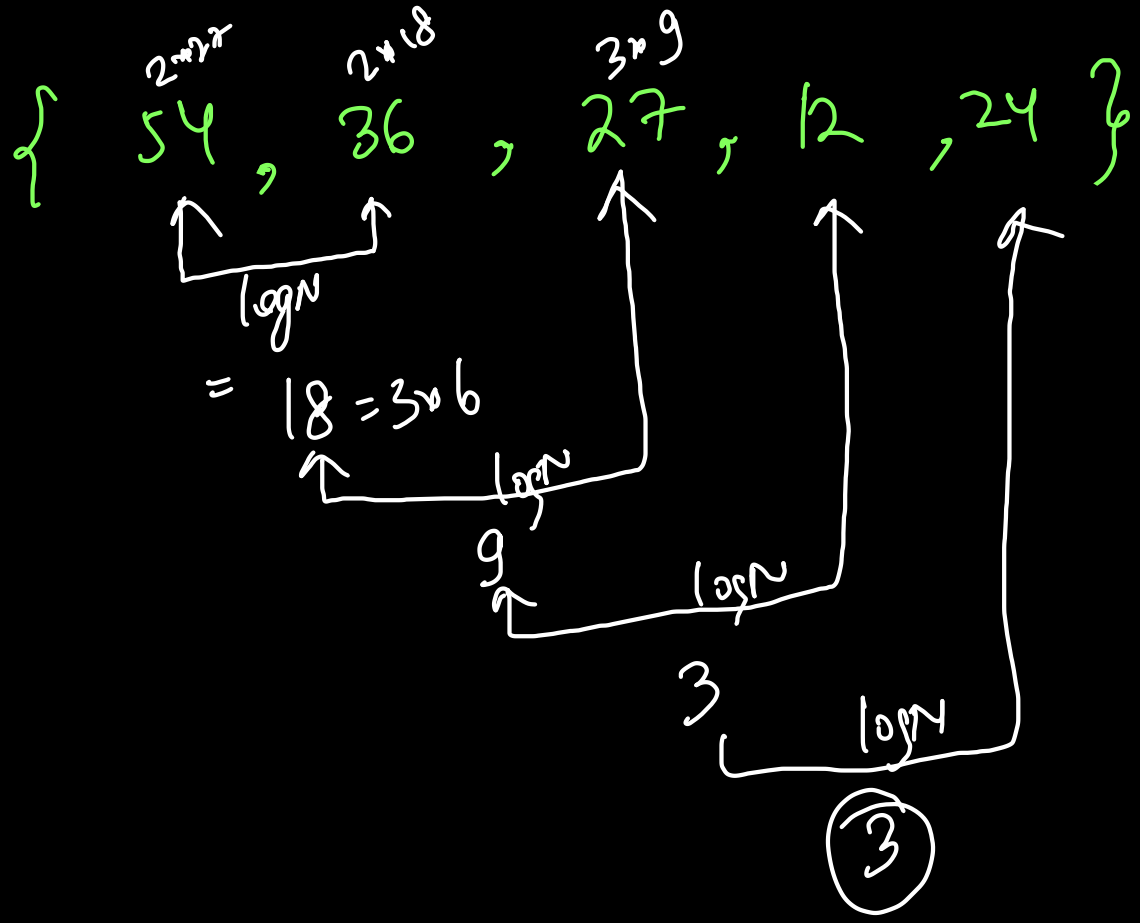
GCD of Array



Total time ⇒ NlogN

```java
class Solution
{
    public int GCD(int a, int b){
        if(b == 0) return a;
        return GCD(b, a % b);
    }
    public int gcd(int N , int arr[]) {
        int gcd = arr[0];

        for(int idx = 1; idx < N; idx++){
            gcd = GCD(gcd, arr[idx]);
        }

        return gcd;
    }
}
```

GCD of Array



Total time ⇒ NlogN

```java
class Solution
{
    public int GCD(int a, int b){
        if(b == 0) return a;
        return GCD(b, a % b);
    }
    public int gcd(int N , int arr[]) {
        int gcd = arr[0];

        for(int idx = 1; idx < N; idx++){
            gcd = GCD(gcd, arr[idx]);
        }

        return gcd;
    }
}
```

Essential maths for DSA

- → Logarithms
- → Permutation & Combination
- → AP & GP , Sequence & Series
- → Number Theory
  - Sets
  - Modular Arithmetic
  - GCD/LCM
  - Prime Numbers

optional

→ Probability

Required Rooms

→ no of foreign = A , no of Indian = B

→ each room : equal students

→ In same room, either all foreign / indian but not both

→ minimum rooms

eg a = 60, b = 84

min^m rooms = 12

gcd(60, 84) = 3 × 2 × 2 = 12

eg a = 30 b = 54

$= 10 * 3$

$= 2 * 5 * 3$

$= 27 * 2$

$= 3 * 9 * 2$

divisor, dividend

gcd(a, b) = 6 ⇒ room size (max m)

foreigner room = 30/6 = 5

indian room = 54/6 = 9

min^m rooms

5 + 9 = 14

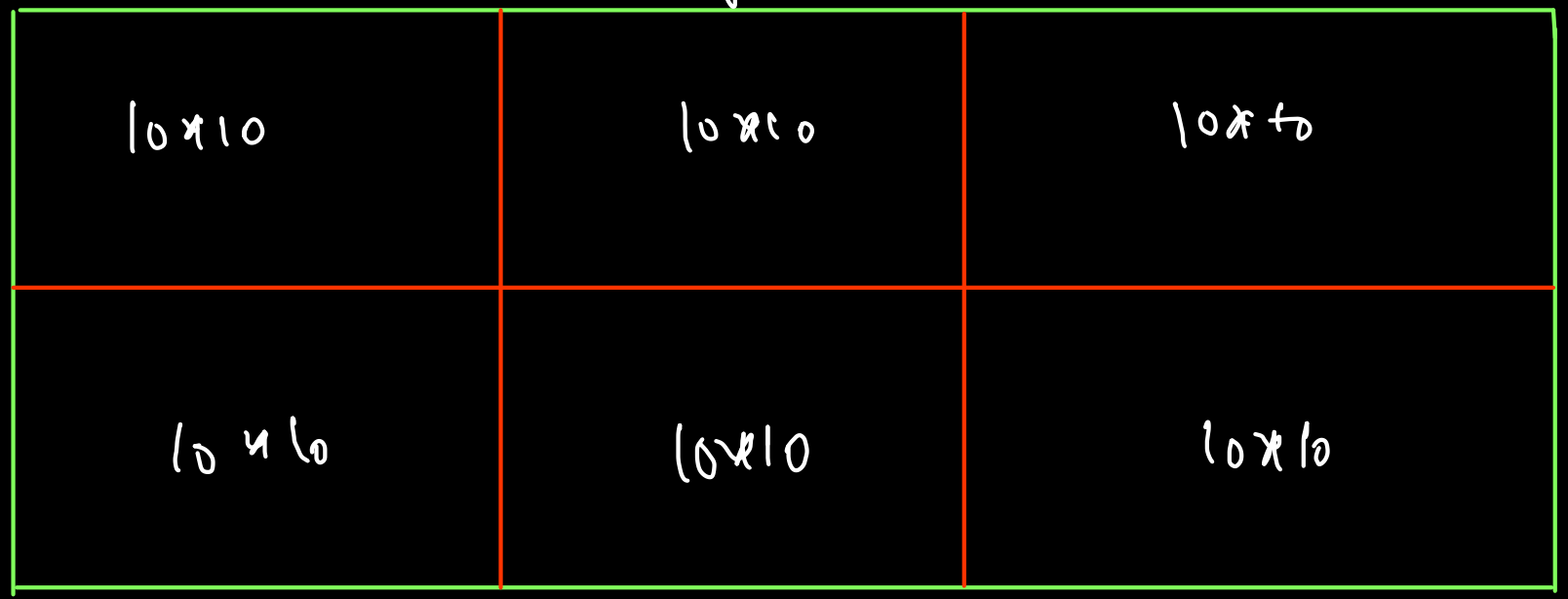
eg a = 8 b = 11

gcd(8, 11) = 1

rooms = 8/1 + 11/1

= 19

```java
class Solution{
    static int gcd(int a, int b){
        if(b == 0) return a;
        return gcd(b, a % b);
    }
    static int rooms(int a, int b){
        int roomSize = gcd(a, b);
        int minRooms = a / roomSize + b / roomSize;
        return minRooms;
    }
}
```



Cutting Squares

all squares of same length

no residue should remain

Sidelength = gcd(l,b)

no of squares = $\frac{\text{area of rectangle}}{\text{area of squares}}$

6 | 6
1 | 6

```java
class Solution{
    static Long gcd(Long A, Long B){
        if(B == 0) return A;
        return gcd(B, A % B);
    }
    static List<Long> minimumSquares(long L, long B)
    {
        Long side = gcd(L, B);
        Long squares = (L * B) / (side * side);
        // area of rectangle / area of square

        List<Long> res = new ArrayList<>();
        res.add(squares);
        res.add(side);
        return res;
    }
}
```

Extended Euclid

$$ax + by = \gcd(a,b)$$

$a = 48 \qquad b = 32$

$\gcd(48, 32) = 16$ (a, b)

$48 * x + 32 * y = 16$

$1 - \lfloor \frac{48}{32} \rfloor * 0$ ↑ ⇓ $y = x' = 0$

$32 * x' + 16 * y' = 16$

$0 - \lfloor \frac{32}{48} \rfloor * 1$ ↑ ⇓ $y' = x'' = 1$

$16 * x'' + 0 * y'' = 16$

$x'' = 1$ $\quad y'' = 0$

$\gcd(48, 32) = \gcd(32, 48 \% 32)$ (b, a % b)

$= \gcd(32, 16)$

$= \gcd(16, 0)$

$x = y' - \lfloor b/a \rfloor * x'$

$y = x'$

Prime Numbers

Check no is prime → 1, number itself

$n = 0$ ambiguous

$n = 1$ ambiguous

$n = 2$ → 1, 2 — Prime

$n = 3$ → 1, 3 — Prime

$n = 4$ → 1, 2, 4 — composite

$n = 5$ → 1, 5 — prime

$n = 6$ → 1, 2, 3, 6 — Composite

$n = 7$ → 1, 7 — prime

$n = 8$ → 1, 2, 4, 8 — composite

$n = 9$ → 1, 3, 9 — composite

$N = 77$

$2 \longrightarrow 76$
$(n-1)$

$77 \% 2 \neq 0$

$77 \% 3 \neq 0$

$77 \% 4 \neq 0$

$77 \% 5 \neq 0$

$77 \% 6 \neq 0$

$77 \% 7 = 0$

Composite

$O(n)$
linear

$N = 17$

$2 \longrightarrow 16$
$(n-1)$

$17 \% 2 \neq 0$

$\% 3 \neq 0$

$\% 4 \neq 0$

$\vdots$

$\% 16 \neq 0$

17 is prime

79

$79 \% 2$

$40 \times 1 \neq 79$

$40 \times 2 = 80$

$40 \times 3 = 120$

$\vdots$ 160 $\vdots$

Approach (1)

```
static int isPrime(int N){
    if(N <= 1) return 0;

    for(int factor = 2; factor < N; factor++){
        if(N % factor == 0) return 0; // composite
    }
    return 1; // prime
}
```

$O(n)$

$N_{max} = 10^9$

10^9 op $\Rightarrow$ 1 sec

TLE

Approach (2)

```
static int isPrime(int N){
    if(N <= 1) return 0;
    if(N <= 3) return 1;

    for(int factor = 2; factor <= N/2; factor++){
        if(N % factor == 0) return 0; // composite
    }
    return 1; // prime
}
```

$O(n/2) \simeq O(n)$

intuition

factor $> N/2$

$2 * factor > N$

Factorization of 48

1 * 48 $48|1$

2 * 24 $48|2$

3 * 16 $48|3$

4 * 12 $48|4$

6 * 8 $48|6$

$\sqrt{48} = 7$

Factorization of 64

1 * 64 $64|1$

2 * 32 $64|2$

4 * 16 $64|4$

8 * 8 $64|8$

root 64 = $\sqrt{64} = 8$

Factorization of 31

1 * 31

x is a factor,

N/x is also a factor

and x is not a factor,

N/x is not a factor

⇒ max x value

```java
class Solution{
    // O(N)
    static int approach1(int N){
        if(N <= 1) return 0;

        for(int factor = 2; factor < N; factor++){
            if(N % factor == 0) return 0; // composite
        }
        return 1; // prime
    }

    // O(N/2)
    static int approach2(int N){
        if(N <= 1) return 0;
        if(N <= 3) return 1;

        for(int factor = 2; factor <= N/2; factor++){
            if(N % factor == 0) return 0; // composite
        }
        return 1; // prime
    }

    // O(Root N)
    static int isPrime(int N){
        if(N <= 1) return 0;
        if(N <= 3) return 1;

        for(int factor = 2; factor * factor <= N; factor++){
            if(N % factor == 0) return 0; // composite
        }
        return 1; // prime
    }
}
```

Number of Divisors

Even Divisors (not a perfect square)

48

1 * 48
2 * 24
3 * 16
4 * 12
6 * 8

$x \neq N/x$

31

1 * 31

$x \neq N/x$

Odd Divisor (perfect square)

$4 = 2^2$

1 * 4
2 * 2

1, 2, 4

$9 = 3^2$

1 * 9
3 * 3

1, 3, 9

$16 = 4^2$

1 * 16
2 * 8
4 * 4

1, 2, 4, 8, 16

$49 = 7^2$

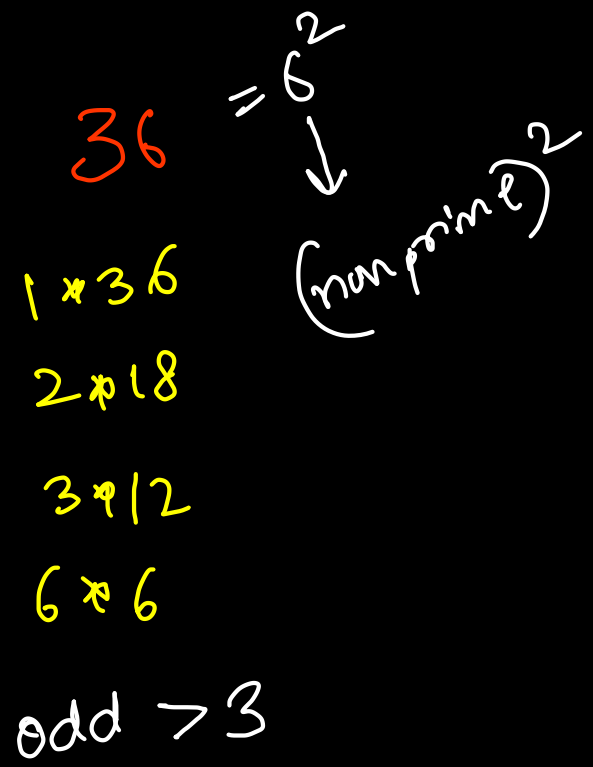
1 * 49
7 * 7

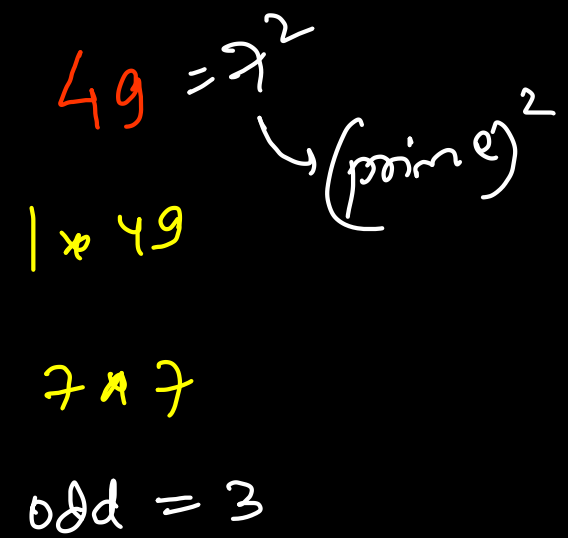
1, 7, 49

$x = N/x$

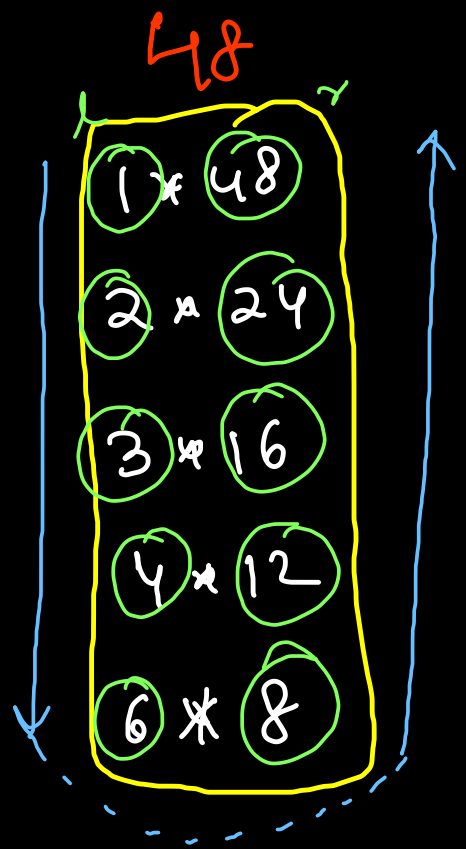
$x^2 = N$

$x = \sqrt{N}$



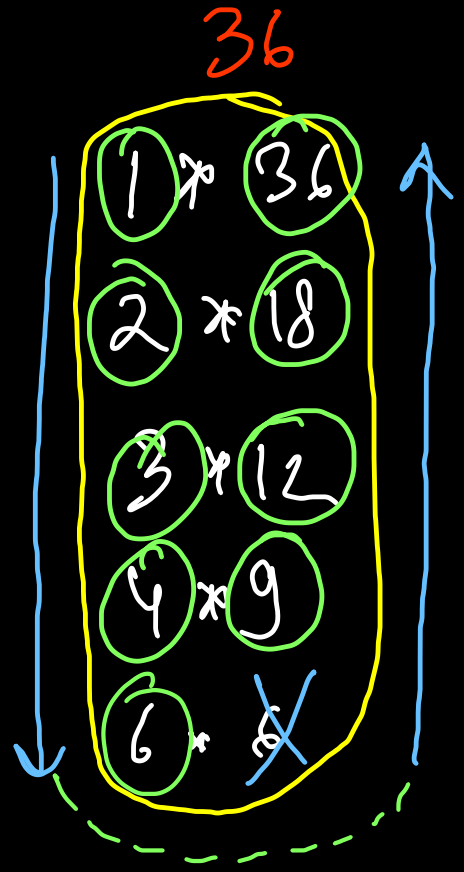



```java
class Solution {
    // O(Log N)
    public boolean isPerfectSquare(int n){
        int sqrt = (int)Math.sqrt(n);
        return (sqrt * sqrt == n);
    }

    // O(Root N)
    public boolean isPrime(int n){
        if(n == 0 || n == 1) return false;

        for(int factor = 2; factor * factor <= n; factor++){
            if(n % factor == 0) return false;
        }
        return true;
    }

    public boolean isThree(int n) {
        // Number should be perfect square (odd divisors)
        if(isPerfectSquare(n) == false) return false;

        // If it is square of prime, then odd divisors = exactly three
        int sqrt = (int)Math.sqrt(n);
        return isPrime(sqrt);
    }
}
```



Sorted order of factors

⇒ 1, 2, 3, 4, 6, 8, 12, 16, 24, 48



Sorted order of factors

1, 2, 3, 4, 6, 9, 12, 18, 36

left + reverse right

```java
public int kthFactor(int n, int k) {
    List<Integer> res = new ArrayList<>();
    List<Integer> right = new ArrayList<>();

    for(int f = 1; f * f <= n; f++){
        if(n % f == 0){
            res.add(f);
            if(f != n / f)
                right.add(n / f);
        }
    }

    Collections.reverse(right);
    res.addAll(right);

    if(k > res.size()) return -1;
    return res.get(k - 1);
}
```

$O(\sqrt{n})$ space, worst case

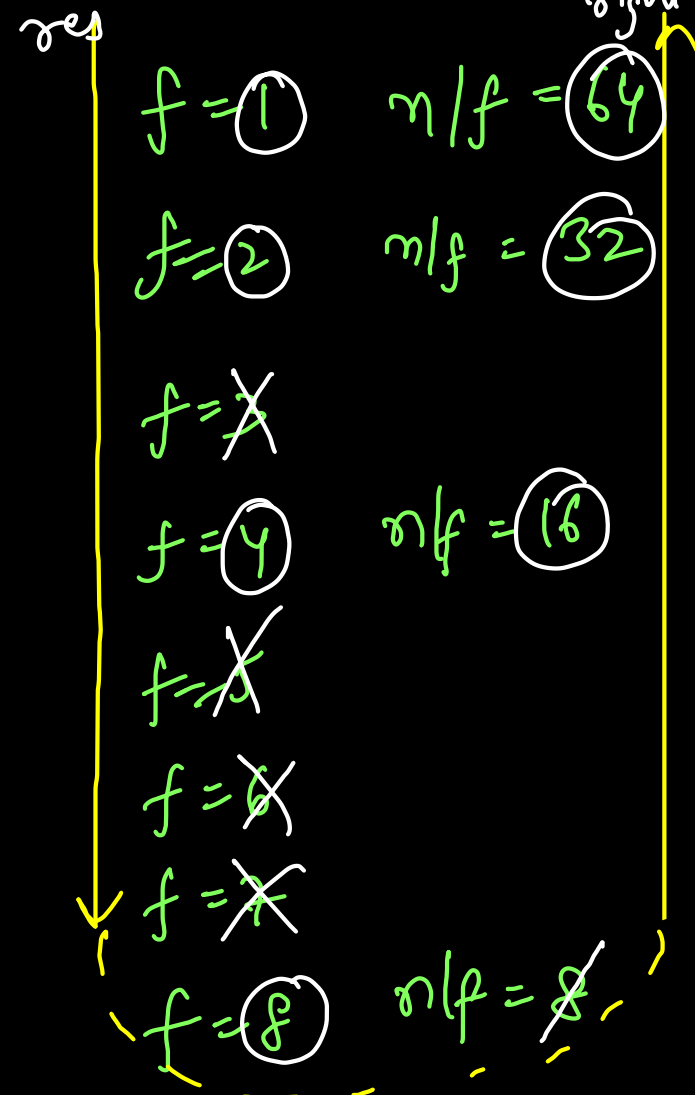
$O(\sqrt{n})$ time

generate all factors in pairs in sorted order

⇒ kth factor does not exist

return kth factor in sorted order

$n = 64$



factors: 1, 2, 4, 8, 16, 32, 64

```java
class Solution {
    public int getFactors(int n){
        int count = 0, sum = 0;
        for(int f = 1; f * f <= n; f++){
            if(n % f == 0) {
                count++;
                sum += f;
                if(f != n / f)  {
                    sum += n / f;
                    count++;
                }
            }
        }

        if(count == 4) return sum;
        return 0;
    }
    public int sumFourDivisors(int[] nums) {
        int sum = 0;
        for(int val: nums){
            sum += getFactors(val);
        }
        return sum;
    }
}
```

factorization

Sum of Divisors of Numbers who have count of divisors = exactly 4

Common Divisors

2100

prime factorizam: $2 \times 2 \times 3 \times 5 \times 5 \times 7$

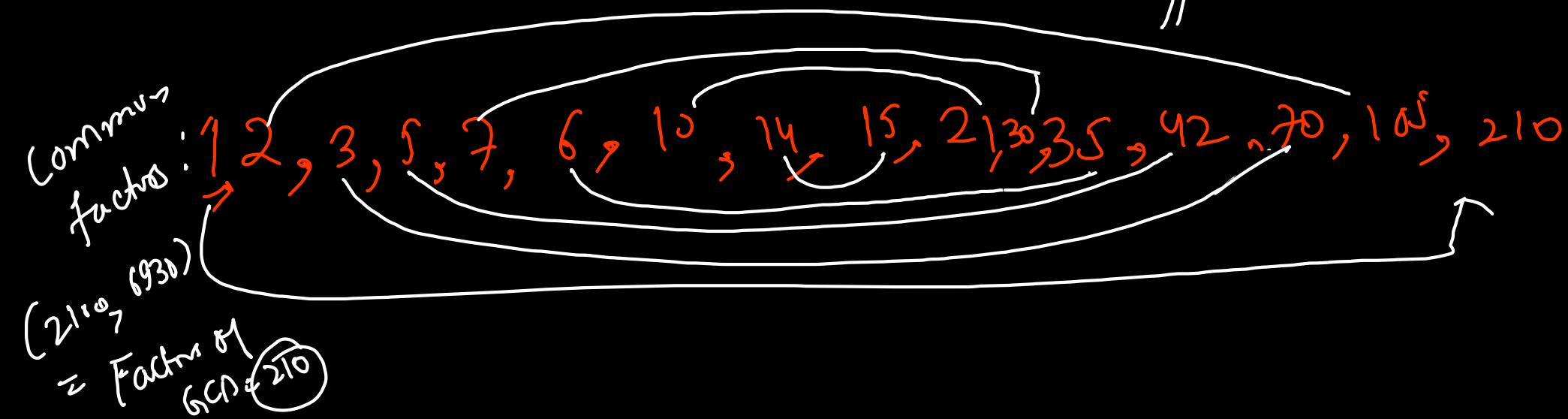
6930

$2 \times 3 \times 3 \times 5 \times 7 \times 11$ prime factorizam

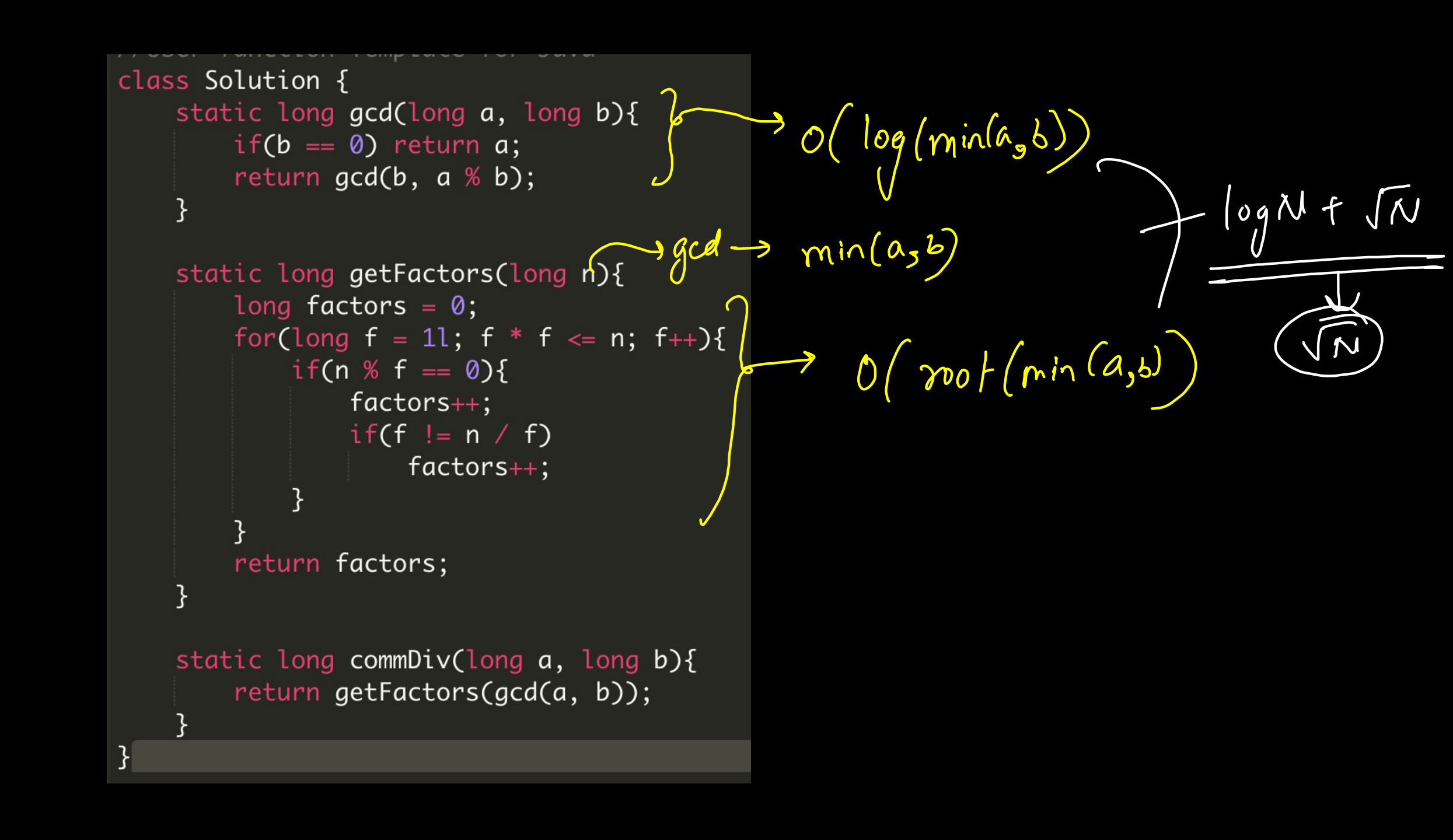
GCD

$2 \times 3 \times 5 \times 7 = 210$

= factorizam



```java
class Solution {
    static long gcd(long a, long b){
        if(b == 0) return a;
        return gcd(b, a % b);
    }

    static long getFactors(long n){
        long factors = 0;
        for(long f = 1l; f * f <= n; f++){
            if(n % f == 0){
                factors++;
                if(f != n / f)
                    factors++;
            }
        }
        return factors;
    }

    static long commDiv(long a, long b){
        return getFactors(gcd(a, b));
    }
}
```



LC 204 Count Primes

$n = 20$

(2), (3), (5), (7), (11), (13), (17), (19) count = 8

$n = 31$

2, 3, 5, 7, 11, 13, 15, 17, 19, 23, 29 count = 11

Bruteforce :⇒ Run a loop on every no from 2 to N-1, check prime

⇒ If yes count++;

$O(\text{no of } N * N) = O(N\sqrt{N}) = O(N^{3/2})$

$N = 10^6 * 5$

$(10^6)^{3/2} = 10^9$ TLE

```java
class Solution {
    public boolean isPrime(int k){
        if(k == 0 || k == 1) return false;
        if(k == 2 || k == 3) return true;

        for(int f = 2; f * f <= k; f++){
            if(k % f == 0) return false;
        }
        return true;
    }

    public int countPrimes(int n) {
        int count = 0;
        for(int k = 2; k < n ; k++){
            if(isPrime(k) == true) count++;
        }
        return count;
    }
}
```

Brute force

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

TLE

Sieve of Eratosthenes

→ channi

multiples of prime no cannot be prime

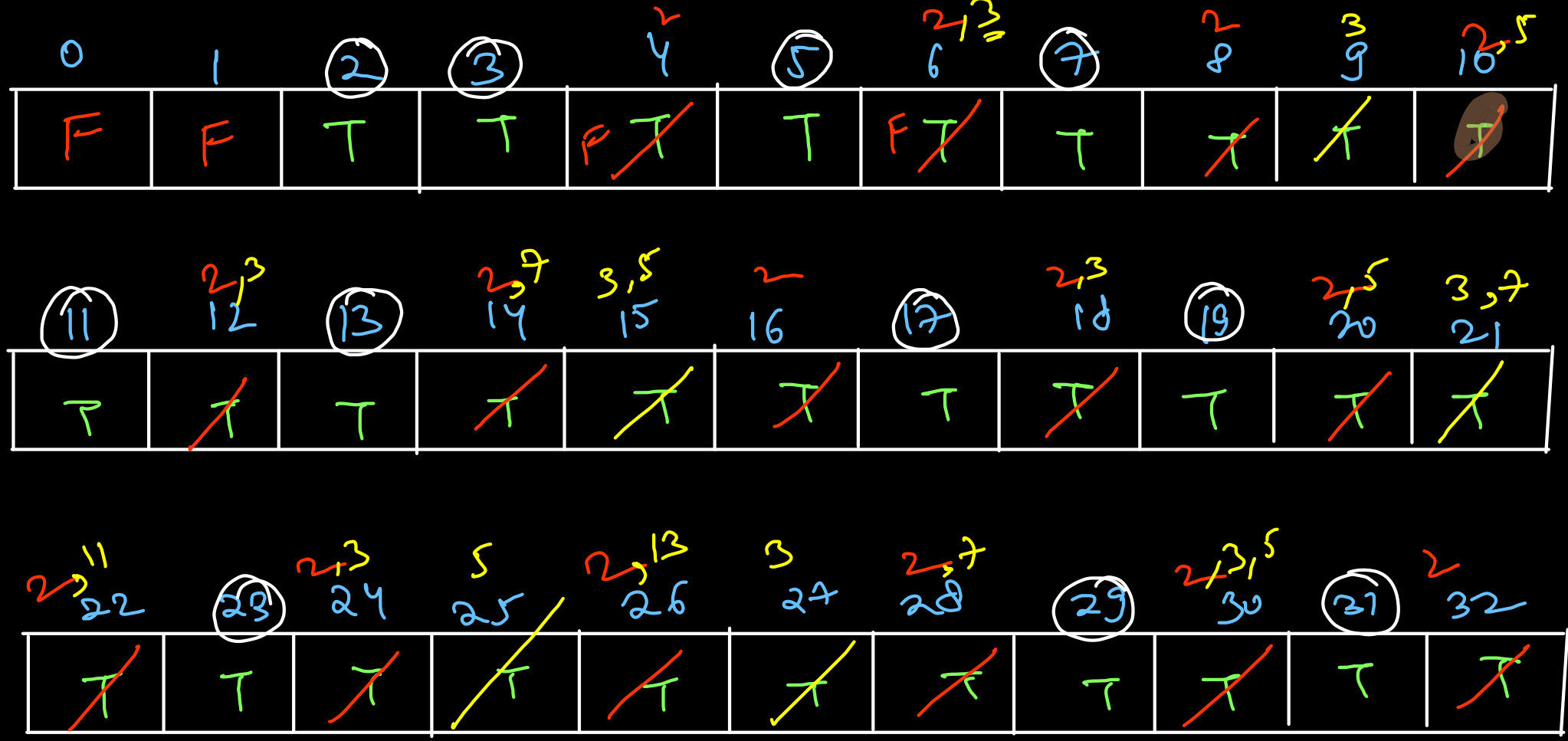
2 → 2×2 = 4, 2×3 = 6, 2×4 = 8, 2×5 = 10, 2×6 = 12, ...

3 → 3×2 = 6, 3×3 = 9, 3×4 = 12, 3×5 = 15, 3×6 = 18, ...

multiple of Composite no are already rejected

4 → 4×2 = 8, 4×3 = 12, 4×4 = 16, 4×5 = 20

6 → 12, 24, 30, 36, ...


0 1 2 3 4 5 6 7 8 9 10
F F T T F T T F T T T T
11 12 13 14 15 16 17 18 19 20 21
T T T T T T T T T T T T
22 23 24 25 26 27 28 29 30 31 32
T T T T T T T T T T T T

$n = 33$

```java
class Solution {
    public int countPrimes(int n) {
        if(n <= 2) return 0;

        boolean[] prime = new boolean[n];
        Arrays.fill(prime, true);
        prime[0] = prime[1] = false;

        int count = 0;
        for(long i = 2l; i < n; i++){
            if(prime[(int)i] == false) continue;
            // if i is itself composite, multiples already rejected

            count++;
            for(long j = i*i; j < n; j += i){
                prime[(int)j] = false;
                // rejecting multiples of prime = i
            }
        }

        return count;
    }
}
```

Time

$O(n \log\log n)$

$\approx O(n)$

$n = 10^9$

$\log n = 9 \times 3$

$= 27$

$\log\log n = \log 27 = 5$