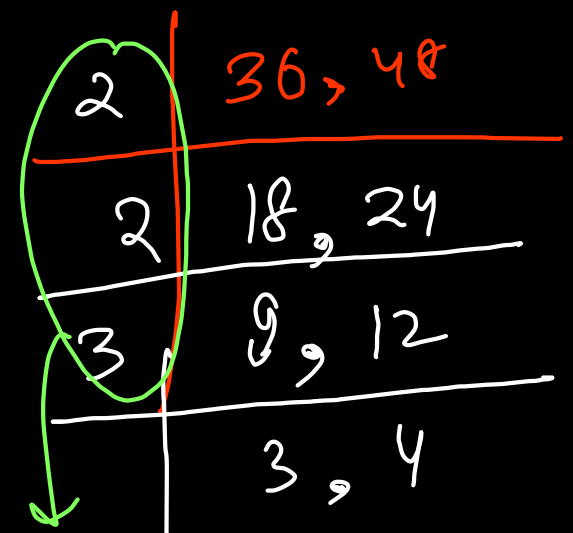
Number Theory

GCD ⟹ Greatest Common Divisor

or

HCF ⟹ Highest Common Factor



GCD(36, 48) = 12

LCM(36, 48) = 144

$2^3 \times 3^2 \times 5^1$

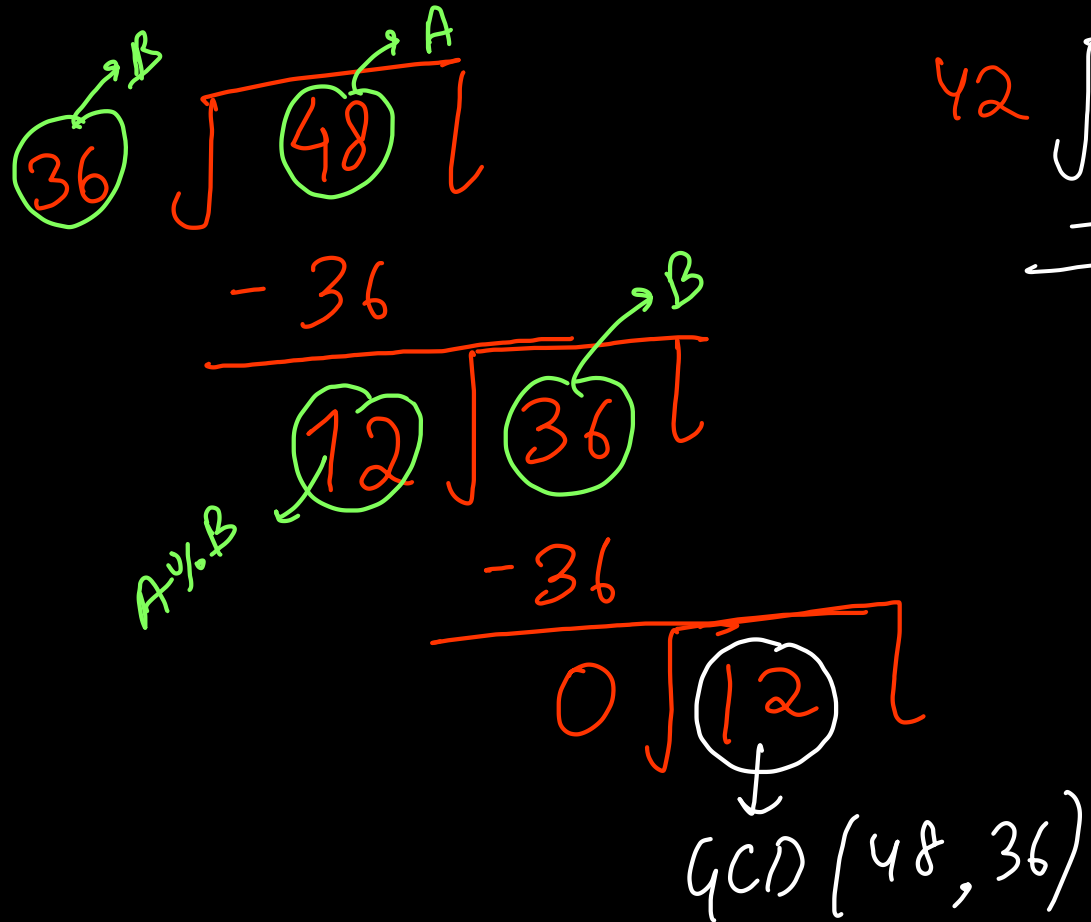
$2^2 \times 3^5 \times 7^2$

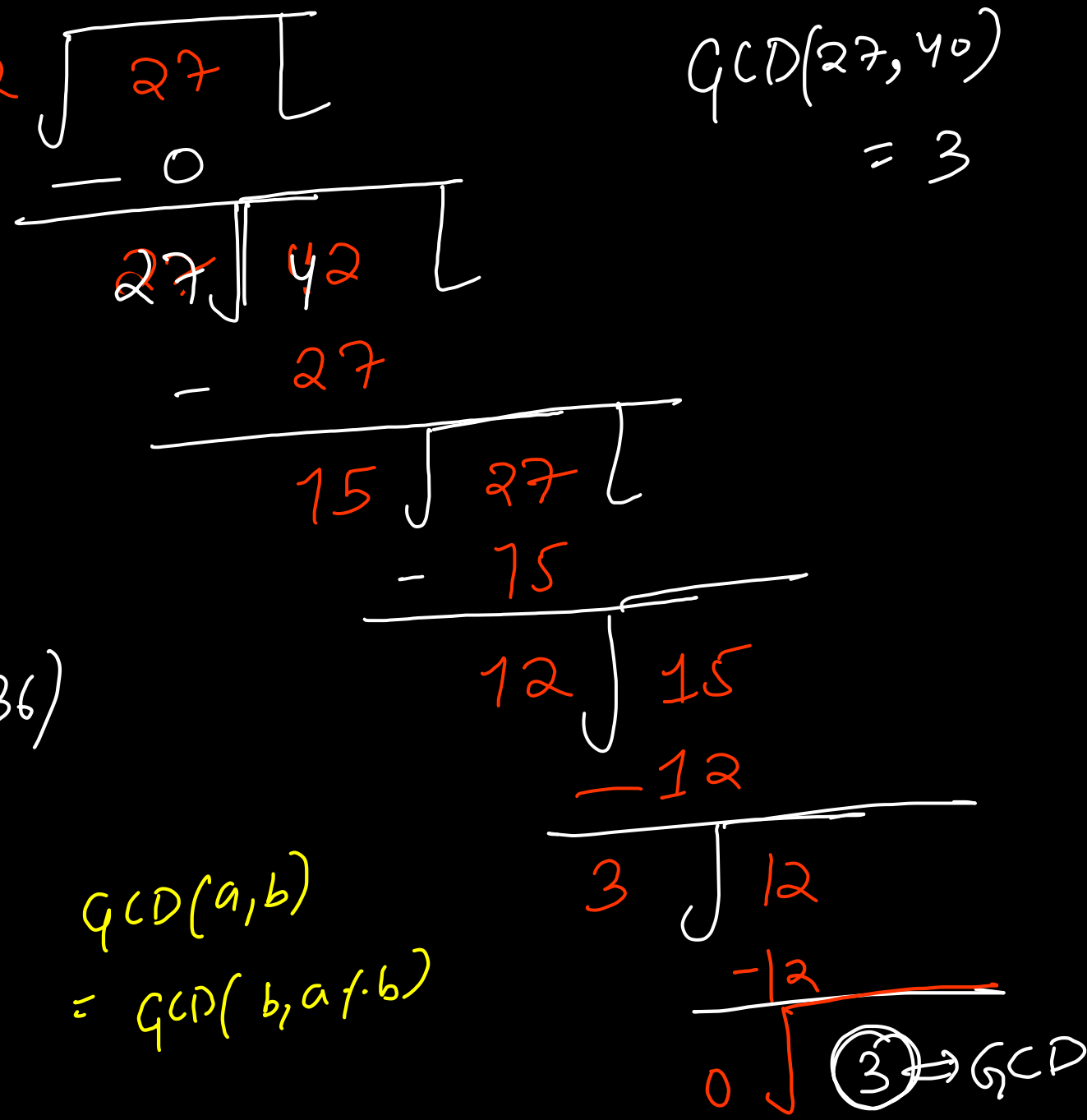
→ HCF → $2^2 \times 3^2 = 4 \times 9$

$= \underline{36}$

LCM = $2^3 \times 3^5 \times 5^1 \times 7^2$

$$\text{LCM} = \frac{a \times b}{\text{GCD}}$$





GCD(27, 40)
= 3

Euclid's Algorithm
(long Division)

GCD(a, b)
= GCD(b, a%b)

```
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 HCF = GCD(A, B);
    Long LCM = (A * B) / HCF;
    return new Long[]{LCM, HCF};
}
```

→ Euclid's Algorithm

⇒ Time ⇒ O(log(min(a,b))

⇒ Space ⇒ O(logarithmic)

GCD of Array

{ 48, 36, 92, 16, 10 }

gcd(a, b, c) = gcd(gcd(a, b), c)

gcd = 48, 12, 4, 1, 2

(Worst case O(N log N))

```
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 ans = arr[0];
    for(int i = 1; i < N; i++){
        ans = gcd(ans, arr[i]);
    }
    return ans;
}
```

Minimum Rooms

foreigners = 36

indians = 48

room-size = gcd(36, 48) = 12

no-of-rooms = $\frac{36}{12} + \frac{48}{12} = 3 + 4 = 7$

Constraint 1) Different rooms for foreigners & indians

Constraint 2) Participants should be divided equally in each room.

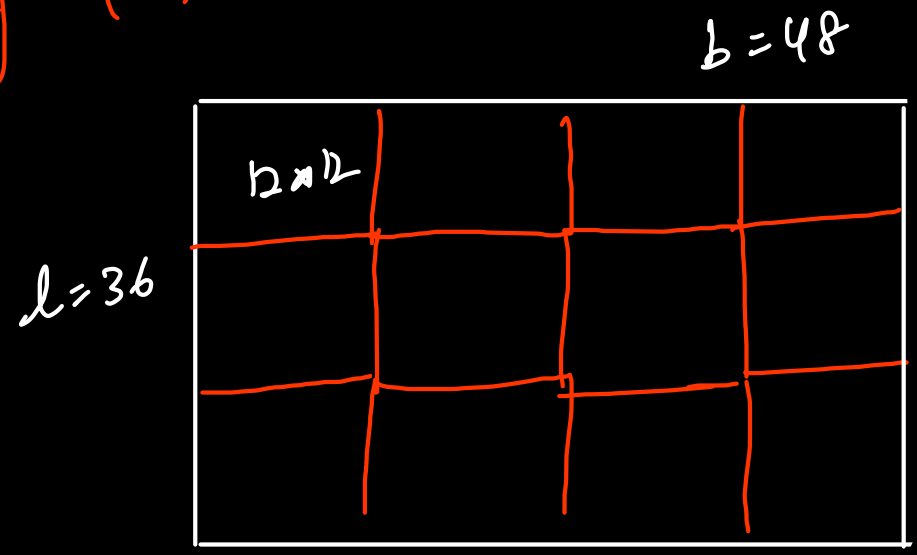
answer = (a+b)/gcd(a,b)

```
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 + B) / roomSize;
    return minRooms;
}
```

Cutting Rectangles

Length = 36
Breadth = 48

$\gcd(36, 48) = 12$



each square should be of same size

→ minimum square
→ maximum side

Side = 12 = gcd(36, 48)

area of rectangle / area of square

= (36 * 48) / (12 * 12)

= 12

```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)
    {
        List<Long> res = new ArrayList<>();

        Long side = GCD(L, B);
        Long count = (L * B) / (side * side);

        res.add(count); res.add(side);
        return res;
    }
}
```

Time => O(log N)

Space => O(log N)

Extended Euclid's Algorithm

$$a * x + b * y = \gcd(a, b)$$

$(x, y) \Rightarrow$ integral pair

$35x + 15y = 5$

$a = 35, \quad b = 15$

$\gcd(a, b) = 5$

$x = 1 \quad y = -2$

$35 * 1 + 15 * -2 = 35 - 30 = 5$

$$GCD(42,27) = GCD(27,15) = GCD(15,12) = GCD(12,3) = GCD(3,0)$$

$a = 42$

$b = 27$

$$x = y'$$

$$y = x' - \left\lfloor \frac{a}{b} \right\rfloor * y'$$

$$\begin{aligned}
42x + 27y &= gcd(42,27) && (x = 2,\ y = -3)\\
27x + 15y &= gcd(27,15) && (x = -1,\ y = 2)\\
15x + 12y &= gcd(15,12) && (x = 1,\ y = -1)\\
12x + 3y &= gcd(12,3) && (x = 0,\ y = 1)\\
3x + 0y &= gcd(3,0) && (x = 1,\ y = 0)
\end{aligned}$$

$ax + by = gcd \longrightarrow$

$bx' + (a\%b)y' = gcd \longrightarrow$

$a\%b = a - \lfloor a/b \rfloor * b$

Eg $15\%4 = 15 - \lfloor 15/4 \rfloor * 4$

$= 15 - 3*4 = 15-12$

$= 3$

$ax + by = gcd$

$bx' + \left(a - \left\lfloor \frac{a}{b} \right\rfloor * b\right) y' = gcd$

$\Downarrow$

$ax + by = a(y')$

$+ b\left(x' - \left\lfloor \frac{a}{b} \right\rfloor y'\right)$

```java
// ans[0] = gcd, ans[1] = x, ans[2] = y

static int[] gcd(int a,int b){
    if(b == 0)
        return new int[]{a, 1, 0};

    int[] ans = gcd(b, a % b);
    long newX = ans[2];
    long newY = ans[1] - ((long)a / b) * ans[2];

    ans[1] = (int)newX; ans[2] = (int)newY;
    return ans;
}
```

Extended Euclid

Time ⇒ $O(\log(\min(a,b)))$

Space ⇒ $O(\log(\min(a,b)))$

Applicam

⇒ Linear Diophantine Eqn

⇒ Multiplicative modulo Inverse (MMI)

↳ modular division

Extended Euclid Algorithm

↳ $ax + by = \gcd(a,b)$

$x = k_1, \quad y = k_2$

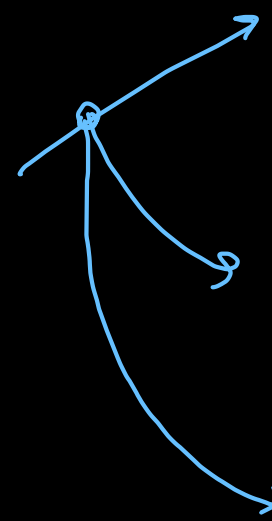
given a, b
find $x, y \in$ integral

Linear Diophantine Eqn

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

$a\left(\frac{x}{k}\right) + b(y/k) = \gcd(a,b)$

$x = k \cdot k_1 \qquad y = k \cdot k_2$



1) Find any x, y integral pair

2) Check if pair exist or not

3) Find all x, y integral pairs

$$\underset{a}{42}x + \underset{b}{27}y = \underset{k}{\underset{\gcd}{3}} * 10;$$

$x = k \cdot (2) = 10 * 2 = 20$

$y = k(-3) = 10 * -3 = -30$

} find any soln

find all integral solns

$(20, -30)$

$-b/g$ ↓ ↓ $+a/g$

$(11, -16)$

$-b/g$ ↓ ↓ $+a/g$

$(2, -2)$

$ax + by = g * k;$

$a(x + \frac{b}{g}) + b(y - \frac{a}{g})$

$= g * k$

$\frac{b}{g} = \frac{27}{3} = 9$

$\frac{a}{g} = \frac{42}{3} = 14$

no integral sols

$ax + by = c$

cannot represent

$ax + by = k * gcd(a,b)$

if C is not a multiple of $gcd(a,b)$

$\Rightarrow$ then no integral soln of (x,y) is possible

$C \% gcd(a,b) == 0$

```
class Solution{
    static int gcd(int A, int B){
        if(B == 0) return A;
        return gcd(B, A % B);
    }

    static int isPossible(int A,int B, int C){
        int g = gcd(A, B);
        if(C % g == 0) return 1; // solutions exits -> c is multiple of gcd
        else return 0; // else no integral solutions
    }
}
```

Prime Numbers

smallest prime No = (2)

↳ only even prime no

0 → even

→ neither prime nor composite

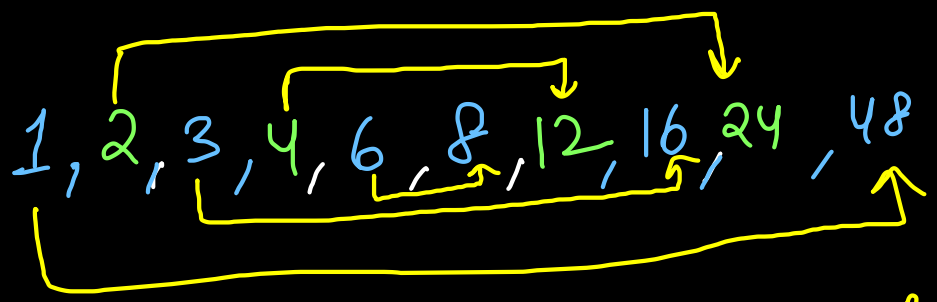
1 → odd

→ prime

3, 5, 7, 11, 13, 17, 19, 23, 29

↳ only 2 factors (1 & itself)

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



all factors are in pairs

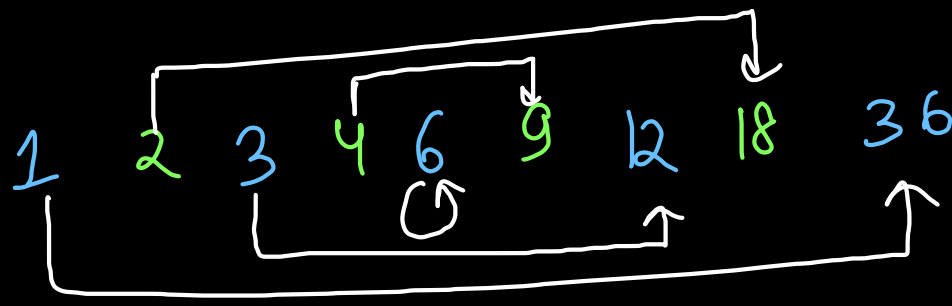
$48 = x^2$

$x = 6 \cdots$

23

2 3 4 5 | ---->

factors of 36



all factors except one factor will be in pairs

$6 = \frac{36}{6} \Rightarrow 36 = 6^2$

36 perfect squar

```java
// Time = O(N), Space = O(1), TLE
static int approach1(int N){
    if(N == 0 || N == 1) return 0;

    for(int f = 2; f < N; f++){
        if(N % f == 0) return 0;
    }
    return 1;
}

// Time = O(N/2), Space = O(1)
static int approach2(int N){
    if(N == 0 || N == 1) return 0;

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

// Time = O(Root N), Space = O(1), Nmax = 10 ^ 16 = Long Value
static int approach3(int N){
    if(N == 0 || N == 1) return 0;

    // all factors are in pairs
    for(int f = 2; f * f <= N; f++){
        if(N % f == 0) return 0;
    }
    return 1;
}
```

Factorization

48 : (1, 48), (2, 24), (3, 16), (4, 12), (6, 8)

{1, 2, 3, 4, 6} + { 48, 24, 16, 12, 8 }

reverse

36 : (1, 36), (2, 18), (3, 12), (4, 9), (6, 6)

$x = \sqrt{x}$

{ 1, 2, 3, 4, 6 } + { 36, 18, 12, 9 }

reverse

```java
public long factorSum(int N)
{
    long sum = 0;
    for(long i = 1; i * i <= N; i++){
        if(N % i != 0) continue;

        sum = sum + i;
        if(i != N / i)
            sum = sum + N / i;
    }

    return sum;
}
```

$O(\sqrt{N})$

```java
static int countFactors(int N) {
    int count = 0;
    for(long i = 1; i * i <= N; i++){
        if(N % i != 0) continue;
        count++;
        if(i != N / i) count++;
    }

    return count;
}
```

$O(\sqrt{N})$

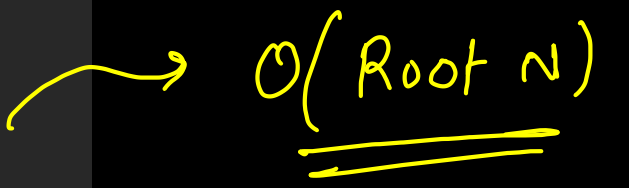
$N = 36$

$i^{\circ} =$

1 ✓	$36 \% 1 = 0$	$36/1 = 36$ ✓
2 ✓	$36 \% 2 = 0$	$36/2 = 18$ ✓
3 ✓	$36 \% 2 = 0$	$36/3 = 12$ ✓
4 ✓	$36 \% 4 = 0$	$36/4 = 9$ ✓
5 ✗	$36 \% 5 \neq 0$	
6 ✓	$36 \% 6 = 0$	$36/6 = 6$ ✗

LC 367)
Perfect Squares

```java
class Solution {
    public int countFactors(int N) {
        int count = 0;
        for(long i = 1; i * i <= N; i++){
            if(N % i != 0) continue;
            count++;
            if(i != N / i) count++;
        }

        return count;
    }

    public boolean isPerfectSquare(int N) {
        int count = countFactors(N);
        if(count % 2 == 1) return true; // perfect square
        return false;
    }
}
```



```java
class Solution {
    public int kthFactor(int N, int k) {
        List<Integer> left = new ArrayList<>();
        List<Integer> right = new ArrayList<>();

        for (long i = 1; i * i <= N; i++) {
            if (N % i != 0) continue;

            left.add((int)i);
            if (i != N / i)
                right.add((int)(N / i));
        }

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

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

→ Time ⇒ O(Root N)

Space ⇒ O(Root N)

Three Divisors

```java
class Solution {
    public boolean isPrime(int n){
        if(n == 0 || n == 1) return false;
        for(int i = 2; i * i <= n; i++){
            if(n % i == 0) return false;
        }
        return true;
    }
    public boolean isThree(int n) {
        // 1, root n, n
        // Numbers which are squares of prime
        //  are having three divisors

        // 4: 1, 2, 4
        // 9: 1, 3, 9
        // 25: 1, 5, 25
        // 49: 1, 7, 49
        // 121: 1, 11, 149

        int sqrt = (int)Math.sqrt(n);
        if(sqrt * sqrt != n) return false;
        return isPrime(sqrt);
    }
}
```

$O(\sqrt{N})$

10:55 PM

```java
int countFactors(int N) {
    int count = 0;
    for (long i = 1; i * i <= N; i++) {
        if (N % i != 0)
            continue;
        count++;
        if (i != N / i)
            count++;
    }

    return count;
}
long factorSum(int N) {
    long sum = 0;
    for (long i = 1; i * i <= N; i++) {
        if (N % i != 0)
            continue;

        sum = sum + i;
        if (i != N / i)
            sum = sum + N / i;
    }

    return sum;
}
```

Four Divisors

```java
public int sumFourDivisors(int[] nums) {
    long sum = 0;
    for(int val: nums){
        if(countFactors(val) == 4){
            sum = sum + factorSum(val);
        }
    }
    return (int)sum;
}
```

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

$a = 2^3 \times 3^2 \times 5^1$ $b = 2^5 \times 3^3 \times 7^2$

$gcd = 2^3 \times 3^2$

$= 8 \times 9 = 72$

factors(72)

Common Divisors / Factors

36 48

(1) (12)
(2) (6)
(3) (4)

factors(12)

= factors(gcd(36, 48))

Count Common Divisors

```java
static long countFactors(long N) {
    long count = 0;
    for (long i = 1; i * i <= N; i++) {
        if (N % i != 0)
            continue;
        count++;
        if (i != N / i)
            count++;
    }

    return count;
}

static long gcd(long a, long b){
    if(b == 0) return a;
    return gcd(b, a % b);
}
static long commDiv(long a, long b){
    return countFactors(gcd(a, b));
}
```

gcd → $O(\log(\min(a,b)))$

+

factors → $O(\sqrt{\min(a,b)})$

Product of all factors

$48 = 1 \times 2 \times 3 \times 4 \times 6 \times 8 \times 12 \times 16 \times 24 \times 48$

$O(\sqrt{N})$ Time

```
class Solution {
    int m = 1000000007;

    long multiplyFactors(int N){
        long res = 1;
        for (long i = 1; i * i <= N; i++) {
            if (N % i != 0)
                continue;

            res = (res * i) % m;
            if (i != N / i)
                res = (res * (N / i)) % m;
        }

        return res;
    }

    int factorProduct(int N) {
        return (int)multiplyFactors(N);
    }
}
```

Prime Factorization

$$n = p_1^{e_1} * p_2^{e_2} * p_3^{e_3} * p_4^{e_4} * \cdots$$

number of factors

$$d(n) = (e_1 + 1) \cdot (e_2 + 1) \cdots (e_k + 1)$$

Sum of factors

$$\sigma(n) = \frac{p_1^{e_1+1} - 1}{p_1 - 1} \cdot \frac{p_2^{e_2+1} - 1}{p_2 - 1} \cdots \frac{p_k^{e_k+1} - 1}{p_k - 1}$$

```java
// https://leetcode.com/problems/perfect-number/description/
// Time = O(root N), Space = O(1)

class Solution {
    public boolean checkPerfectNumber(int N) {
        long sum = 0;
        for (long i = 1; i * i <= N; i++) {
            if (N % i != 0)
                continue;

            sum = sum + i;
            if (i != N / i)
                sum = sum + N / i;
        }

        return (sum - N == N);
    }
}
```

Leetcode 204) Count Primes

$n = 30$ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 ans = 10

Brute force :→ 2 to N-1 loop : Every no check if it is prime or not

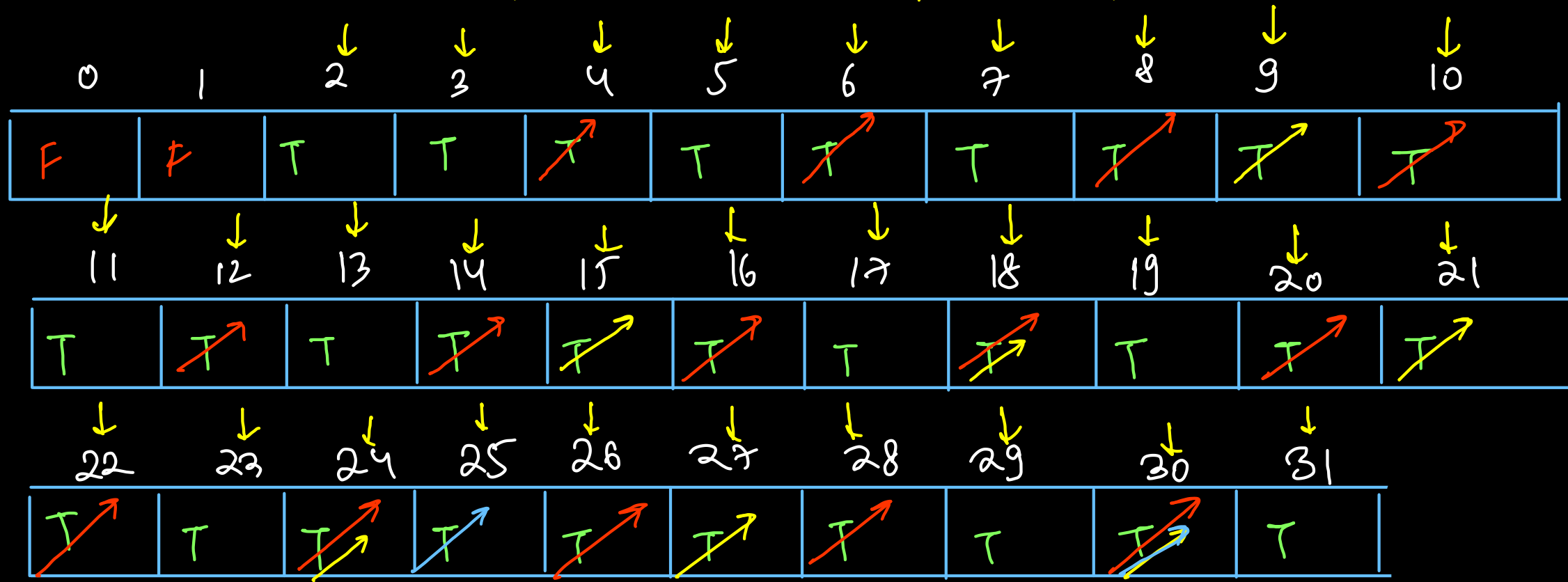
↳ $O(N \times \sqrt{N}) = O(N^{3/2})$

$10^{6 \times 3/2} = 10^9$ TLE

Prime Seive Of Eratosthenes

$n = 32$

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31



Time complexity = $O(n \log\log n) \simeq O(n)$

```java
public int countPrimes(int n) {
    boolean[] vis = new boolean[n + 1];
    Arrays.fill(vis, true);
    int count = 0;
    for(long i = 2; i < n; i++){
        if(vis[(int)i] == false) continue;
        count++;
        for(long j = i * i; j < n; j += i)
            vis[(int)j] = false;
    }

    return count;
}
```

Pre-optimizatn technique

$O(n \log\log n) \approx O(n)$ time

$N = 10^8$ max

$O(n)$ space

$N = 10^6$ max (array)

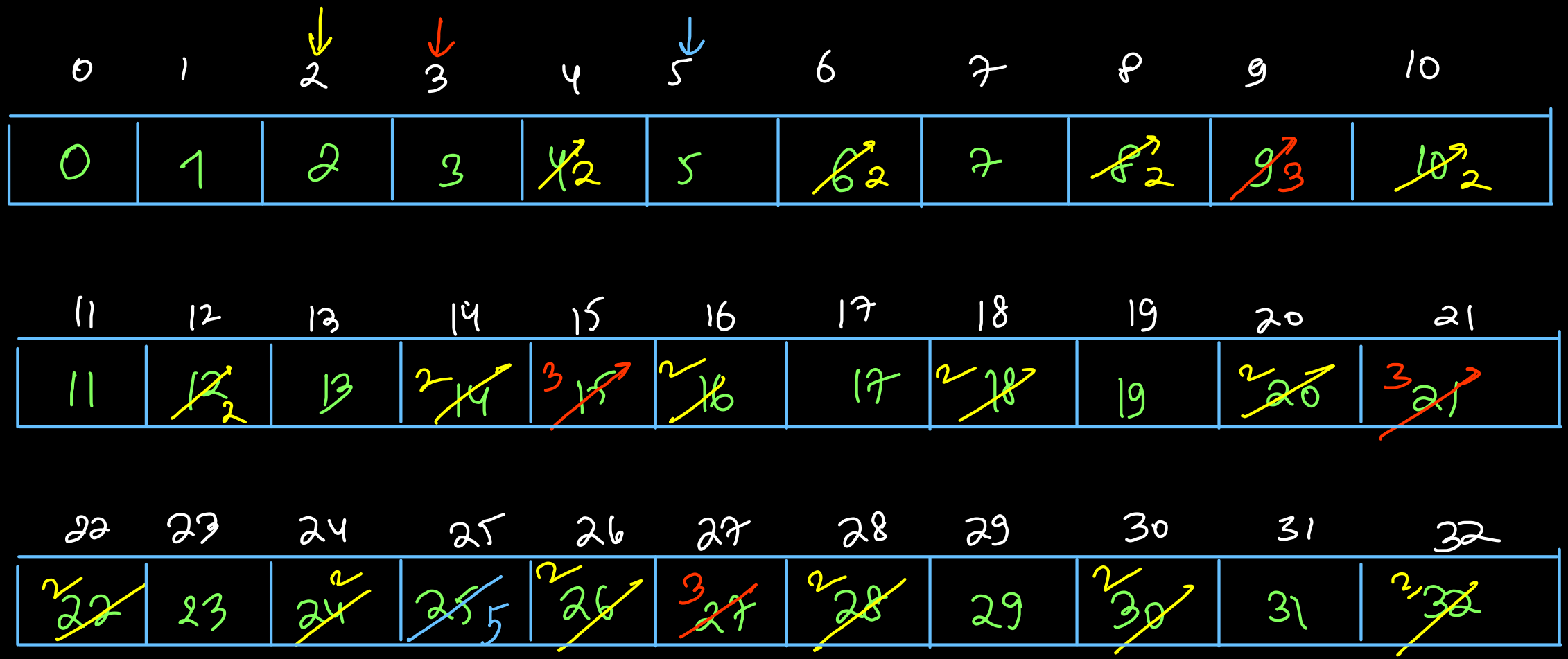
```java
public int countPrimes(int n) {
    if(n <= 2) return 0;
    BitSet vis = new BitSet(n + 1);
    vis.set(2, n + 1);

    int count = 0;
    for(long i = 2; i < n; i++){
        if(vis.get((int)i) == false) continue;
        count++;
        for(long j = i * i; j < n; j += i)
            vis.clear((int)j);
    }

    return count;
}
```

→ set all bits except 0 & 1

→ check ith bit is set or not

↳ unset jth bit (mark multiply of prime false)

least prime Factor



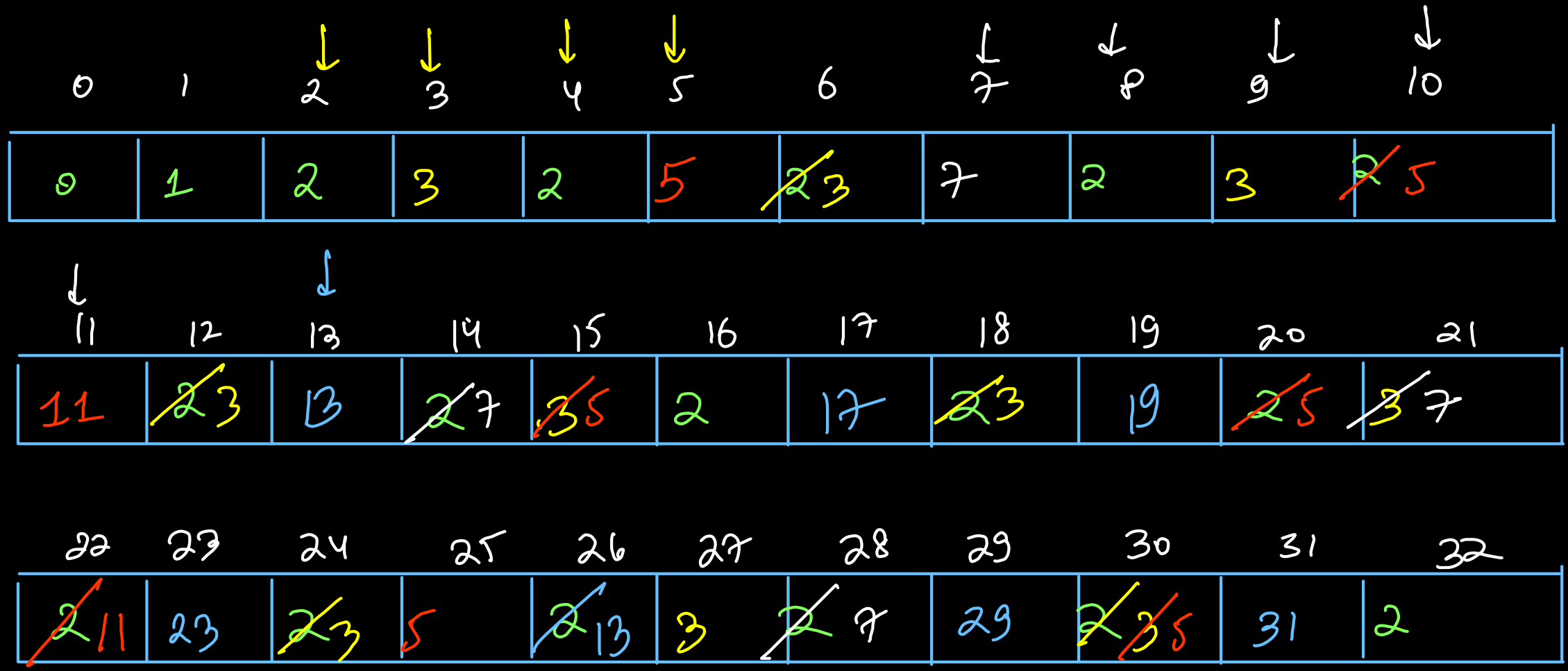
```java
public int[] leastPrimeFactor(int n)
{
    int[] vis = new int[n + 1];
    vis[1] = 1;

    for(long i = 2; i <= n; i++){
        if(vis[(int)i] > 0) continue;
        vis[(int)i] = (int)i;

        for(long j = i * i; j <= n; j += i){
            if(vis[(int)j] == 0)
                vis[(int)j] = (int)i;
        }
    }
    return vis;
}
```

multiples of 'i'

lowest prime factor

Time = $n \log \log n$

Space = $O(n)$


0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 2 5 23 7 2 3 25
11 12 13 14 15 16 17 18 19 20 21
11 23 13 27 35 2 17 23 19 25 37
22 23 24 25 26 27 28 29 30 31 32
211 23 23 5 213 3 7 29 35 31 2

```java
public int[] highestPrimeFactor(int n){
    int[] vis = new int[n + 1];
    vis[1] = 1;

    for(long i = 2; i <= n; i++){
        if(vis[(int)i] > 0) continue;
        vis[(int)i] = (int)i;

        for(long j = 2 * i; j <= n; j += i){
            vis[(int)j] = (int)i;
        }
    }
    return vis;
}
```

Time ⇒ O(n log n)

Space ⇒ O(n)

2^0 3 5^2, 7^3, 11^4, 13^5, 17^6, 19^7, 23^8, 29^9, 31^{10}

```java
import java.util.*;
0 references
public class Solution {
    static List<Integer> primes;

    0 references
    static void primesieve(int n){
        boolean[] vis = new boolean[n + 1];
        Arrays.fill(vis, true);

        primes = new ArrayList<>();
        for(long i = 2; i < n; i++){
            if(vis[(int)i] == false) continue;

            primes.add((int)i);
            for(long j = i * i; j < n; j += i)
                vis[(int)j] = false;
        }
    }

    0 references
    public static int findNthPrime(int n) {
        if(primes == null){          NloglogN
            primesieve(10000005); // precomputation
        }

        return primes.get(n - 1);
    }
}
```

Number of Queries = Q

Number = N

Precomputation → NloglogN

Q = 5

n = 10 → 29 → O(1)

n = 5 → 11 → O(1)

n = 7 → 17 → O(1)

n = 1 → 2 → O(1)

n = 9 → 23 → O(1)

→ O(Q)

Total time = O(NloglogN + Q)

Prime Factorisation

$504 = 2^3 * 3^2 * 7^1$

$n = p_1^{a_1} \times p_2^{a_2} \times p_3^{a_3} \ldots$

2	504
2	252
2	126
3	63
3	21
7	7
	1

```java
public int[] AllPrimeFactors(int N) {
    ArrayList<Integer> primes = new ArrayList<>();

    for(int f = 2; f <= N; f++){
        if(N % f == 0) primes.add(f);
        while(N % f == 0){
            N = N / f;
        }
    }

    int[] res = new int[primes.size()];
    for(int i=0; i<res.length; i++) res[i] = primes.get(i);
    return res;
}
```

↓ optimization

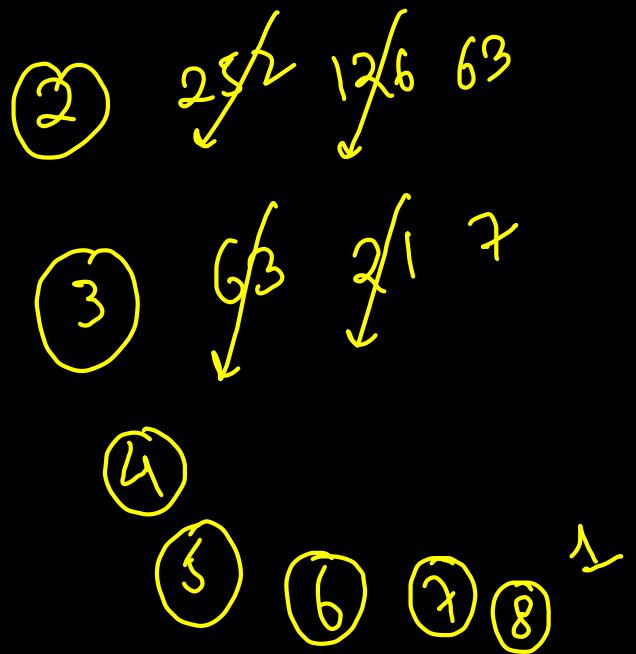
```java
public int[] AllPrimeFactors(int N) {
    ArrayList<Integer> primes = new ArrayList<>();

    for(int f = 2; f * f <= N; f++){
        if(N % f == 0) primes.add(f);
        while(N % f == 0){
            N = N / f;
        }
    }
    if(N != 1) primes.add(N);

    int[] res = new int[primes.size()];
    for(int i=0; i<res.length; i++) res[i] = primes.get(i);
    return res;
}
```

Time → O(n) worst case
Time → O(√n) avg case

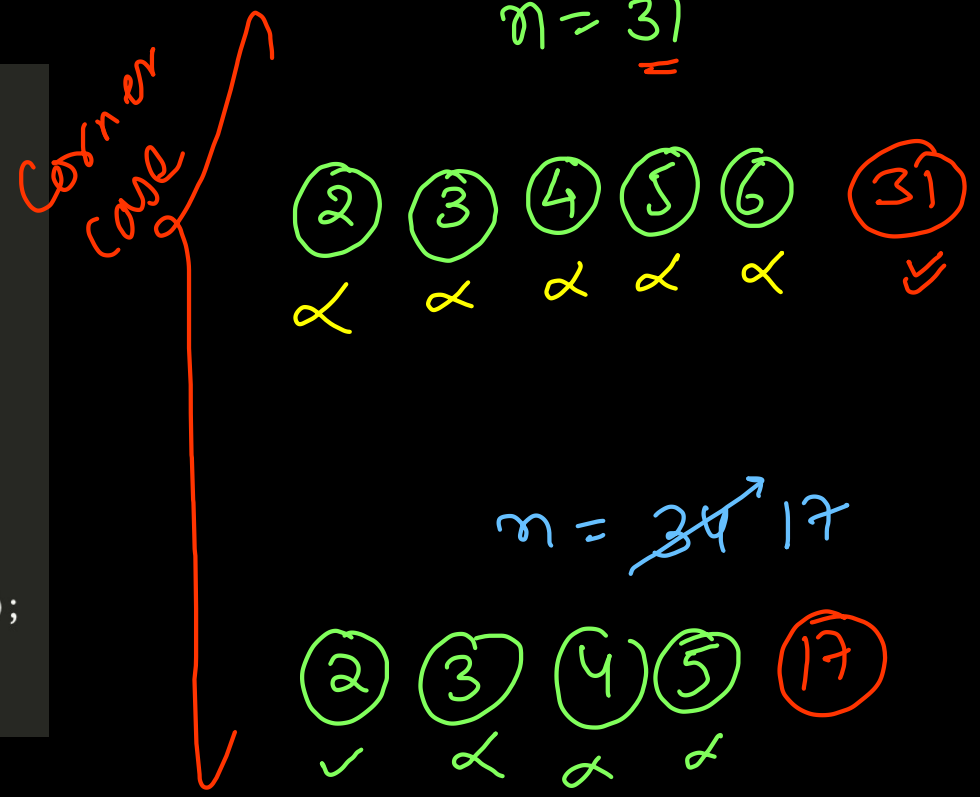
Space ⇒ O(1)
no extra space

504

N! = 1



```java
public int[] AllPrimeFactors(int N) {
    ArrayList<Integer> primes = new ArrayList<>();

    for(int f = 2; f * f <= N; f++){
        if(N % f == 0) primes.add(f);
        while(N % f == 0){
            N = N / f;
        }
    }
    if(N != 1) primes.add(N);

    int[] res = new int[primes.size()];
    for(int i=0; i<res.length; i++) res[i] = primes.get(i);
    return res;
}
```



Prime Factorization Queries

Brute force → Nested loop → $O(\sqrt{N} * q)$

$\sqrt{10^6} * 10^5$

$10^3 * 10^5 = 10^8$

Precomputation ⇒ fill the least prime factor array

Precomputation (least prime factor)

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

11	12	13	14	15	16	17	18	19	20	21
11	12 2	13	14 2	15 3	16 2	17	18 2	19	20 2	21 3

22	23	24	25	26	27	28	29	30	31	32
22 2	23	24 2	25 5	26 2	27 3	28 2	29	30 2	31	32 2

(2) (3) (5)

Input: {30, 28}

n = 30 15 5 1

(2) (2) (7)

n = 28 14 7 1

Output: { {2, 3, 5}, {2, 2, 7} }

```java
public static int[] leastPrimeFactor(int n) {
    int[] vis = new int[n + 1];
    vis[1] = 1;

    for (long i = 2; i <= n; i++) {
        if (vis[(int) i] > 0)
            continue;
        vis[(int) i] = (int) i;

        for (long j = i * i; j <= n; j += i) {
            if (vis[(int) j] == 0)
                vis[(int) j] = (int) i;
        }
    }
    return vis;
}

public static ArrayList<ArrayList<Integer>> primeFactorization(ArrayList<Integer> arr, int q){
    int[] lpf = leastPrimeFactor(1000005); // precomputation

    ArrayList<ArrayList<Integer>> res = new ArrayList<>();
    for(int N: arr){
        ArrayList<Integer> primeFactors = new ArrayList<>();
        while(N != 1){
            primeFactors.add(lpf[N]);
            N /= lpf[N];
        }
        res.add(primeFactors);
    }
    return res;
}
```

precomputation
$\Rightarrow n \log\log n$

queries = q

$O(\log N)$ per query

Total time
$= n \log\log n + q \log n$
$10^6 + 10^5$

Divisors / Factors queries

Brute force ⇒ find all factors

$O(\sqrt{n}) \times q = O(\sqrt{n} * q)$

$\sqrt{5 * 10^5} * 2 * 10^5 \Rightarrow$ TLE

precomputation :→

→ j is multiple of i

→ i is a factor/divisor of j

```
for(int i=1; i<=MAX; i++)
    for(int j=2*i; j<=MAX; j+=i)
        divisors[j].add(i)
```

```
for(int i=1; i<=MAX; i++)
    for(int j=2*i; j<=MAX; j+=i)
        divisors[j].add(i)
```

i=1 i=4
i=2 i=5
i=3

$\Downarrow$ $O(n \log n)$

query → O(1) constant per query

Total time = $O(n \log n + q)$

1 → 0
2 → 0+1
3 → 0+1
4 → 0+1+2
5 → 0+1
6 → 0+1+2+3
7 → 0+1
8 → 0+1+2+4
9 → 0+1+3
10 → 0+1+2+5
11 → 0+1
12 → 0+1+2+3+4

13 → 0+1
14 → 0+1+2
15 → 0+1+3+5
16 → 0+1+2+4
17 → 0+1
18 → 0+1+2+3
19 → 0+1
20 → 0+1+2+4+5
21 → 0+1+3
22 → 0+1+2
23 → 0+1
24 → 0+1+2+3+4

```java
public class Solve {
    static class FastReader { ... }

    static class FastWriter { ... }

    static FastReader scn = new FastReader();
    static FastWriter out = new FastWriter();

    static { ... }

    public static void main(String[] args) {
        try {
            solve();
            out.close();
        } catch (Exception e) {
            System.out.println(e);
        }

    }

    public static void solve() throws Exception {
        int n = 500000;
        long[] factors = new long[n + 1];

        for(long i=1; i<=n; i++){
            for(long j=2*i;j<=n; j+=i){
                factors[(int)j] += i;
            }
        }

        int t = scn.nextInt();
        while(t-- > 0){
            int i = scn.nextInt();
            out.println(factors[i]);
        }
    }
}
```

main

→ while submitting

DIVSUM (spoj)

Sum of proper divisors queries!

→ Space = O(n)

→ Precomputation ⇒ nlogn

+

→ queries → q

Segmented Sieve

Prime numbers b/w L to R

$1 \le L \le R \le 10^{10}$ → $R - L \le 10^6$

Sieve of eratosthenes

↳ time = $n \log\log n$

$N_{max} = 10^8$

↳ space = n

$N_{max} = 10^6$

Example $L = 11$ $R = 30$

11, 13, 17, 19, 23, 29

(1) primes (precomputation)

2, 3, 5 → all prime nos $\leq \sqrt{30}$

$\leq \sqrt{R}$

11	12	13	14	15	16	17	18	19	20	21
0	1	2	3	4	5	6	7	8	9	10
✓	F	✓	F	F	F	✓	F	✓	F	F

22	23	24	25	26	27	28	29	30
11	12	13	14	15	16	17	18	19
F	✓	F	F	F	F	F	✓	F

logical value = idx + left

2 → 12, 3 → 12, 5 → 25

$\left(\frac{11}{2}\right)^{\times 2}$ 10+2 $\left(\frac{11}{3}\right)^{\times 3}$ 9+3 $(11/5)^{\times 5} = 10+5 = 15$

```java
static List<Long> primesieve(int n){
    boolean[] vis = new boolean[n + 1];
    Arrays.fill(vis, true);

    List<Long> primes = new ArrayList<>();
    for(long i = 2; i < n; i++){
        if(vis[(int)i] == false) continue;

        primes.add(i);
        for(long j = i * i; j < n; j += i)
            vis[(int)j] = false;
    }

    return primes;
}
```

```java
public static void solve() throws Exception {
    List<Long> primes = primesieve(1000005);

    int t = scn.nextInt();
    while(t-- > 0){
        long left = scn.nextLong();
        long right = scn.nextLong();

        boolean[] vis = new boolean[(int)(right - left + 1)];
        Arrays.fill(vis, true);

        for(long i: primes){
            if(i * i > right) break;

            long start = (left / i) * i;
            if(start < left) start += i;
            start = Math.max(start, i * i);

            for(long j = start; j <= right; j += i){
                vis[(int)(j - left)] = false;
            }
        }

        for(long i = left; i <= right; i++){
            if(i >= 2 && vis[(int)(i - left)] == true){
                System.out.println(i);
            }
        }
        System.out.println();
    }
}
```

Total time

$q(R-L) + \sqrt{R}$

$10 * 10^6 + \sqrt{10^{10}}$

$10^7 + 10^5$ ✓

Segmented Sieve

$q * (R-L) \log\log(R-L)$

↳ precomputation

↳ $O(N \log\log N)$

dry run ??

Modular Arithmetic

$m = 7$

$n \% m = [0, 7)$

$= 0, 1, 2, 3, 4, 5, 6$

% → modulus operator → remainder

$0 \% 7 = 0 \xrightarrow{+7} 7 \% 7 = 0$

$1 \% 7 = 1 \xrightarrow{+7} 8 \% 7 = 1$

$2 \% 7 = 2 \xrightarrow{+7} 9 \% 7 = 2$

$3 \% 7 = 3 \xrightarrow{+7} 10 \% 7 = 3$

$4 \% 7 = 4 \xrightarrow{+7} 11 \% 7 = 4$

$5 \% 7 = 5 \xrightarrow{+7} 12 \% 7 = 5$

$6 \% 7 = 6 \xrightarrow{+7} 13 \% 7 = 6$

$-1 \% 7 \equiv 6 \% 7$

$-2 \% 7 \equiv 5 \% 7$

$-3 \% 7 \equiv 4 \% 7$

$-4 \% 7 \equiv 3 \% 7$

$-5 \% 7 \equiv 2 \% 7$

$-6 \% 7 \equiv 1 \% 7$

$-7 \% 7 \equiv 0 \% 7$

Modulus Properties

① modular Addition

$$(a+b) \% m = (a\%m + b\%m) \% m$$

eg $(36+40) \% 7 = (36\%7 + 40\%7) \% 7 = (1+5) \% 7 = 6$

② modular multiplication

$$(a*b) \% m = (a\%m * b\%m) \% m$$

eg $(44*32) \% 7 = (44\%7 * 32\%7) \% 7 = (2*4) \% 7 = 1$

③ Modular Subtraction

$$(a-b)\%m = (a\%m - b\%m + m)\%m$$

eg: $(10-15)\%7 = (10\%7 - 15\%7)\%7$

$-5\%7 = (3-1)\%7$

$= 2 = 2\%7$

eg: $(15-10)\%7 = (15\%7 - 10\%7)\%7$

$5\%7 = (1-3+7)\%7$

$5 = 5\%7$

$m = 10^9 + 7 = 1000000007$

1) Within integer range $\longrightarrow$ rem < m

2) prime number $\longrightarrow$ MMI

3) close to Integer.MAX-VALUE $\longrightarrow$ Spacial resultant

Large Modular Factorial

$m = 10^9 + 7$

$5! \% m = (5 * 4 * 3 * 2 * 1) \% m$

resultant $= (1 * 5) \% m$

$= (5 * 4) \% m$

$= (20 * 3) \% m$

$= (60 * 2) \% m$

$= (120 * 1) \% m$

$= 120$

$(a * b * c) \% m$

$= ((a \% m) * (b \% m) * (c \% m)) \% m$

$10^4 \quad * \quad 10^4 \quad * \quad 10^4$

```java
public long[] factorial(long arr[], int n) {
    long[] fact = new long[100006];
    fact[0] = fact[1] = 1;
    for(int f = 2; f < fact.length; f++)
        fact[f] = (fact[f - 1] * f) % 1000000007;

    long[] ans = new long[n];
    for(int i = 0; i < n; i++)
        ans[i] = fact[(int)arr[i]];

    return ans;
}
```

10^9+7

O(N) + O(N)

$0! = 1$

$1! = 1$

$2! = (1! \times 2) \% m$

$3! = (2! \times 3) \% m$

$n! = ((n-1)! \times n) \% m$

Time ⇒ O(n) linear

Space ⇒ O(n) dp (fact) table

```cpp
unsigned long long factorial(int n)
{
    const unsigned int M = 1000000007;
    unsigned long long f = 1;

    for (int i = 1; i <= n; i++)
        f = f * i;  // WRONG APPROACH as
                    // f may exceed (2^64 - 1)

    return f % M;
}

// This code is contributed by Shubham Singh
```

wrong

Time Complexity: O(n)

Auxiliary Space: O(1)

C++ C Java Python3 C# Javascript

```cpp
unsigned long long factorial(int n)
{
    const unsigned int M = 1000000007;

    unsigned long long f = 1;
    for (int i = 1; i <= n; i++)
        f = (f*i) % M;  // Now f never can
                        // exceed 10^9+7
    return f;
}
```

correct

Modular Exponentiation (Power)

$x = 2, \quad n = 10, \quad m = 1003$

$x^n \% m \Rightarrow 2^{10} \% 1003 = 1024 \% 1003 = 21$

Divide & Conquer (Recursive)

$x^n = x^{n/2} * x^{n/2}$ (even n)

$= x^{n-1} * x$ (odd n)

$x^n \% m = (x^{n/2} \% m * x^{n/2} \% m) \% m$

even $\Rightarrow n$

$2^{10} \% m = (2^5 * 2^5) \% m$

$x^n \% m = (x^{n-1} \% m * x \% m) \% m$

odd $\Rightarrow n$

$2^5 \% m \quad (2^4 * 2) \% m$

```java
public class Solution {
    0 references
    public static int modularExponentiation(int x, int n, int m) {
        if(n == 0) return 1;
        if(n % 2 == 1){
            long ans = modularExponentiation(x, n - 1, m);
            return (int)((ans * x) % m);
        }

        long ans = modularExponentiation(x, n / 2, m);
        return (int)((ans * ans) % m);
    }
}
```

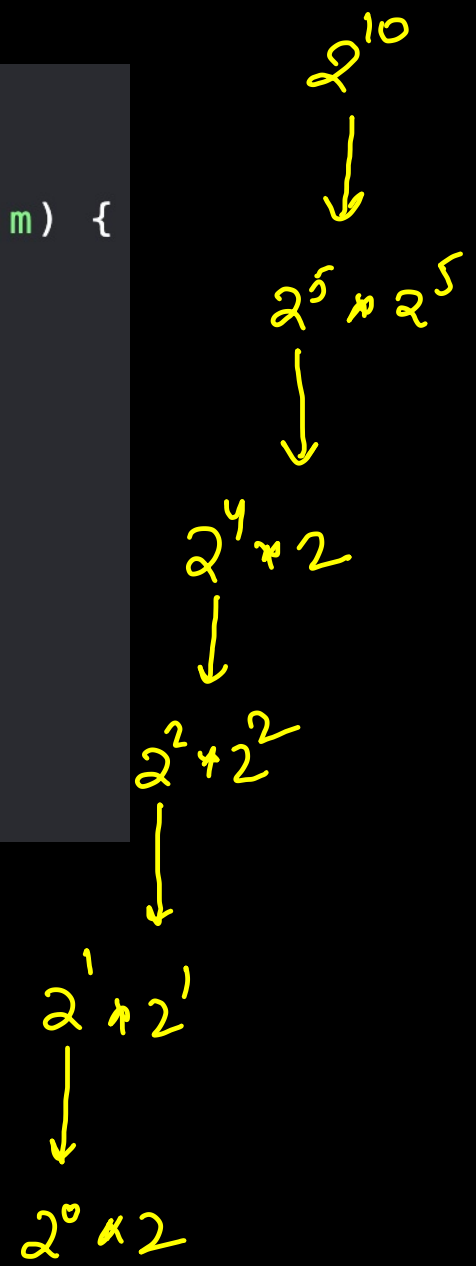
linear

logarithmic

Time ⇒ O(log n)

Space ⇒ O(log n)

} → Recursive (D & C)



Euler Totient Function

$\phi(n)$ = count of numbers from $[1, n)$ which are co-prime with n.

co-prime ↓

$gcd(a, b) = 1$

$gcd(2, 3) = 1$

$gcd(7, 10) = 1$

$\phi(1) = 0$

$\phi(2) = \{1\} = 1$

$\phi(3) = \{1, 2\} = 2$

$\phi(4) = \{1, 3\} = 2$

$\phi(5) = \{1, 2, 3, 4\} = 4$

$\phi(6) = \{1, 5\} = 2$

$\phi(7) = \{1, 2, 3, 4, 5, 6\} = 6$

$\phi(8) = \{1, 3, 5, 7\} = 4$

Properties

1) $\phi(p) = p-1$ where p = prime no

$\phi(10^9+7) = 10^9+6$

2) $\phi(a*b) = \phi(a) * \phi(b)$ where a, b are coprime

$\phi(10) = \phi(2) * \phi(5) = 1*4 = 4 = \{1, 3, 7, 9\}$

3) generically, $\phi(a*b) = \dfrac{\phi(a) * \phi(b) * g}{\phi(g)}$ where $g = gcd(a, b)$

$\phi(16) = \dfrac{\phi(4) * \phi(4) * 4}{\phi(4)} = 2*4 = 8$ $\{1, 3, 5, 7, 9, 11, 13, 15\}$

4) $\phi(p^k) = p^k - p^{k-1} = p^k\left(1-\frac{1}{p}\right)$

$a = p_1^{a_1} * p_2^{a_2} * p_3^{a_3} * \cdots$

$\phi(a) = \phi(p_1^{a_1}) * \phi(p_2^{a_2}) * \phi(p_3^{a_3}) \cdots$

$= p_1^{a_1}\left(1-\frac{1}{p_1}\right) * p_2^{a_2}\left(1-\frac{1}{p_2}\right) * p_3^{a_3}\left(1-\frac{1}{p_3}\right) * \cdots$

$\phi(a) = a * \left(1-\frac{1}{p_1}\right) * \left(1-\frac{1}{p_2}\right) * \left(1-\frac{1}{p_3}\right)$ (prime factorization)

$$\phi(a) = a * \left(1 - \frac{1}{b_1}\right) * \left(1 - \frac{1}{b_2}\right) * \left(1 - \frac{1}{b_3}\right) \quad \text{(prime factorization)}$$

eg $\phi(2^2 * 3^1 * 5^3) = 2^2 * 3^1 * 5^3 \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{5}\right)$

```
static long ETF(long N) {
    long ans = N;

    for(int f = 2; f * f <= N; f++){
        if(N % f == 0)
            ans = ans - ans / f;
        while (N % f == 0)
            N = N / f;
    }

    if(N != 1) ans = ans - ans / N;
    return ans;
}
```

↝ Time Complexity = Prime Fact
O(root N)
{using prime factorization}

- If p is a prime number, then $\gcd(p, q) = 1$ for all $1 \leq q < p$. Therefore we have:

$$\phi(p) = p - 1.$$

- If p is a prime number and $k \geq 1$, then there are exactly p^k / p numbers between 1 and p^k that are divisible by p. Which gives us:

$$\phi(p^k) = p^k - p^{k-1}.$$

- If a and b are relatively prime, then:

$$\phi(ab) = \phi(a) \cdot \phi(b).$$

  This relation is not trivial to see. It follows from the Chinese remainder theorem. The Chinese remainder theorem guarantees, that for each $0 \leq x < a$ and each $0 \leq y < b$, there exists a unique $0 \leq z < ab$ with $z \equiv x \pmod{a}$ and $z \equiv y \pmod{b}$. It's not hard to show that z is coprime to ab if and only if x is coprime to a and y is coprime to b. Therefore the amount of integers coprime to ab is equal to product of the amounts of a and b.

- In general, for not coprime a and b, the equation

$$\phi(ab) = \phi(a) \cdot \phi(b) \cdot \frac{d}{\phi(d)}$$

  with $d = \gcd(a, b)$ holds.

Thus, using the first three properties, we can compute $\phi(n)$ through the factorization of n (decomposition of n into a product of its prime factors). If $n = {p_1}^{a_1} \cdot {p_2}^{a_2} \cdots {p_k}^{a_k}$, where p_i are prime factors of n,

$$\begin{aligned}
\phi(n) &= \phi({p_1}^{a_1}) \cdot \phi({p_2}^{a_2}) \cdots \phi({p_k}^{a_k}) \\
&= \left({p_1}^{a_1} - {p_1}^{a_1-1}\right) \cdot \left({p_2}^{a_2} - {p_2}^{a_2-1}\right) \cdots \left({p_k}^{a_k} - {p_k}^{a_k-1}\right) \\
&= p_1^{a_1} \cdot \left(1 - \frac{1}{p_1}\right) \cdot p_2^{a_2} \cdot \left(1 - \frac{1}{p_2}\right) \cdots p_k^{a_k} \cdot \left(1 - \frac{1}{p_k}\right) \\
&= n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right)
\end{aligned}$$

Euler's Theorem

$a^{\phi(n)} \equiv 1 \bmod m \Rightarrow a^{\phi(n)} \% m = 1$

where $\gcd(a, n) = 1$

Fermat's Little Theorem

n = prime $\Rightarrow \phi(n) = p-1$

$a^{p-1} \% p = 1$ where a, p are co-prime

eg $a = 2, b = 11, m = 1003$

$2^{\phi(11)} \% 1003 = 2^{10} \% 1003 = 1024 \% 1003 = 21$

Modular Division

$$(a/b) \% m \neq (a \% m / b \% m) \% m$$

$$= (a \% m * (\frac{1}{b}) \% m) \% m$$

$$= (a \% m * b^{-1} \% m) \% m$$

where b^{-1} = Multiplicative modular inverse of b (wrt m)

eg $(15/10) \% 11 \neq ((15 \% 11) / (10 \% 11)) \% 11$

$(4 / 10) \% 11 = 0$

For simplicity m = prime ($\leq 10^9 + 7$)

$a + ? = 0$ additive$(a) = -a$

$a * ? = 1$ multiplicate$(a) = 1/a$

$b * \frac{1}{b} = 1 \bmod m \Rightarrow (b * b^{-1}) \,\%\, m = 1$

using Fermat's little theorem $\Rightarrow$ $b * b^{-1} = b^{m-1}$

$b^{-1} = b^{m-2} \,\%\, m$

Short formula (m = prime**) on basis of fermat

b & m are co-prime

$$(a/b) \% m = (a \% m * b^{m-2} \% m) \% m$$

b^{m-2} → modular exponentiation (power)

eg $(15/10) \% 11 = ((15 \% 11) * (10^{9} \% 11)) \% 11$

$b^{-1} = b^{m-2}$

$= 10^{11-2} = 10^{9}$

$= (4 * 10) \% 11$

$= 40 \% 11 = 7$

Generic formula

MMI for any pair of (a,m)

$$a * a^{-1} \equiv 1 \bmod m$$

① Extended Euclid Algo.

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

② lets take $b = m$

$$\Rightarrow ax + my = \gcd(a, m)$$

③ Taking modulus both side

$$(ax + my) \% m = \gcd(a,m) \% m$$

$$ax \% m = \gcd(a,m) \% m$$

④ Comparing $\Rightarrow \gcd(a,m) == 1$

$\Rightarrow$ a,m are co-prime then only MMI exist

⑤ $x = a^{-1}$

$2 * ? \equiv 1 \bmod 6$ $\quad \gcd(2,6) \neq 1$

mmi doesn't exist

$2 * ? \equiv 1 \bmod 3$ $\quad \gcd(2,3) = 1$

$ans = 2^{-1} = 2 \bmod 3$

$2 * ? \equiv 1 \bmod 5$

$ans = 2^{-1} = 3 \bmod 5$

```java
int[] gcd(int a, int b) {
    if (b == 0)
        return new int[] { a, 1, 0 }; // gcd, x, y

    int[] ans = gcd(b, a % b);
    long newX = ans[2];
    long newY = ans[1] - ((long) a / b) * ans[2];

    ans[1] = (int) newX;
    ans[2] = (int) newY;
    return ans;
}

public int modInverse(int a, int m)
{
    int[] ans = gcd(a, m);
    if(m == 1 || ans[0] != 1) return -1;
    // if a & m are not co-prime, MMI does not exist
    return (ans[1] % m + m) % m;
}
```

→ extended euclid algorithm

$\log(\min(a, m))$

make the MMI positive

$-3 \% 7 \equiv 4 \% 7 \equiv 11 \% 7$

$+7$ $+7$

Modular Binomial Coefficient

$$nC_r \,\%\, m = \frac{n!}{(n-r)! * r!} = \left(\frac{n}{b * c}\right) \%\, m$$

$$= (n/d) \%\, m$$

$$= (n \%\, m * d^{m-2} \%\, m) \%\, m$$

MMI

$n!$ $\quad$ $(n-r)! \times r!$

```java
public long power(long x, long n, long m) {
    if (n == 0)
        return 1;
    if (n % 2 == 1) {
        long ans = power(x, n - 1, m);
        return ((ans * x) % m);
    }

    long ans = power(x, n / 2, m);
    return ((ans * ans) % m);
}
```

$x^n \% m = (den)^{m-2} \% m$

$O(\log m)$

```java
public long fact(long a, long m){
    long ans = 1;
    for(long f = 1; f <= a; f++){
        ans = (ans * f) % m;
    }
    return ans % m;
}
```

$O(a)$

```java
public int nCr(long n, long r)
{
    long m = 1000003;

    long num = fact(n, m);
    long d1 = fact(n - r, m);
    long d2 = fact(r, m);

    long den = (d1 * d2) % m;
    long mmi = power(den, m - 2, m);

    return (int)((num % m * mmi % m) % m);
}
```

$num \rightarrow O(n)$

$d1 \rightarrow O(n-r)$

$d2 \rightarrow O(r)$

TLE

```java
class Solution {
    public static long power(long x, long n, long m) {
        if (n == 0)
            return 1;
        if (n % 2 == 1) {
            long ans = power(x, n - 1, m);
            return ((ans * x) % m);
        }

        long ans = power(x, n / 2, m);
        return ((ans * ans) % m);
    }

    public static long fact(long a, long m) {
        long ans = 1;
        for (long f = 1; f <= a; f++) {
            ans = (ans * f) % m;
        }
        return ans % m;
    }

    public static int P(int n, int r) {
        long m = 1000000007;
        long num = fact(n, m);
        long den = fact(n - r, m);
        long mmi = power(den, m - 2, m);

        return (int) ((num % m * mmi % m) % m);
    }
}
```

prime no → d^{m-2}

Permutation coefficient

Time ⇒ O(linear) (10^8)

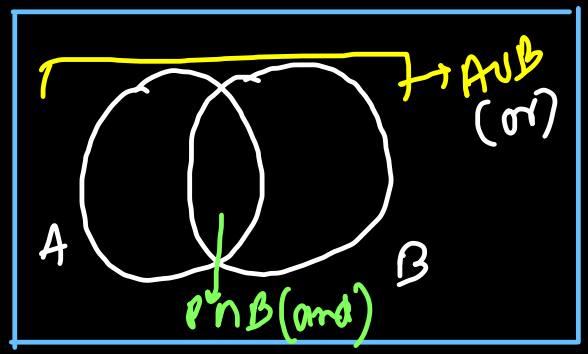
Space ⇒ O(logarithmic)

DP based soln → Time ⇒ O(n*r) (10^4)

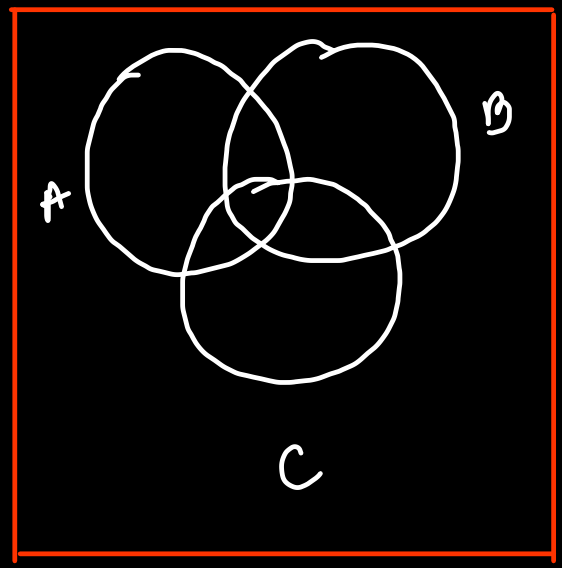
Space ⇒ O(n)

Set Theory

Inclusion Exclusion Principle



$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$



$$P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(B \cap C) - P(C \cap A) + P(A \cap B \cap C)$$

Applicatn 1) Count of numbers which are multiples of both 2 or 3 in b/w [1, 500]

Answer) $P(2 \cup 3) = P(2) + P(3) - P(2 \cap 3)$

$= \frac{500}{2} + \frac{500}{3} - \frac{500}{6}$

2, 4, 6, 8, 10, 12 — 3, 6, 9, 12, 15, 18 — 6, 12, 18

Applicatn 2) Count binary strings of length n which will start at 0 or end at 1.

Answer) Answer = P(start at 0) + P(end at 1) − P(start at 0 AND end at 1)

A = P(start at 0), B = P(end at 1), C = P(start at 0 AND end at 1)

A = 0 0/1 0/1 0/1 0/1

$= 2^{n-1}$

B = 0/1 0/1 0/1 0/1 1

$= 2^{n-1}$

C = 0 0/1 0/1 0/1 1

$= 2^{n-2}$

$= 2^{n-1} + 2^{n-1} - 2^{n-2}$

$= 2^n - 2^{n-2}$

Leetcode 878) Nth magical no.

A positive integer is *magical* if it is divisible by either `a` or `b`.

Given the three integers `n`, `a`, and `b`, return the `n^th` magical number. Since the answer may be very large, **return it modulo** `10^9 + 7`.

a = 2, b = 3, n = 10

2	3	4	6	8	9	10	12	14	15
1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th

a = 2, b = 3, n = 10

left = 0 right = 50

mid = 25 ⇒ count = 25/2 + 25/3 − 25/6

= 12 + 8 − 4 = 16 > 10

left = 0 right = 24

mid = 12 ⇒ count = 12/2 + 12/3 − 12/6

= 6 + 4 − 2 = 8 < 10

left = 13 right = 24

mid = 18 ⇒ count = 18/2 + 18/3 − 18/6

= 9 + 6 − 3 = 12 < 10

left = 13 right = 17

mid = 15 ⇒ count = 15/2 + 15/3 − 15/6

= 7 + 5 − 2 = 10 == 10

left = 13 right = 14

mid = 13

count = 13/2 + 13/3 − 13/6

= 6 + 4 − 2

= 8

left = 14 right = 14

mid = 14

count = 14/2 + 14/3 − 14/6

= 9

left = 15 right = 14

l > r

```java
class Solution {
    public long count(long mid, long a, long b){
        return (mid / a)  + (mid / b) - (mid / (a * b));
    }
    public int nthMagicalNumber(int n, int a, int b) {
        long left = 0, right = Long.MAX_VALUE;

        while(left <= right){
            long mid = left + (right - left) / 2;

            if(count(mid, a, b) < n)
                left = mid + 1;
            else right = mid - 1;
        }

        return (int)(left % 1000000007);
    }
}
```

This code will work only if a & b are coprime

a = 6, b = 10, n = 5

6, 12, 18, 24, 30, 36, 42, 48, 54, 60

10, 20, 30, 40, 50, 60, 70, 80

mid = 60

$\frac{60}{10} + \frac{60}{6} - \frac{60}{30}$

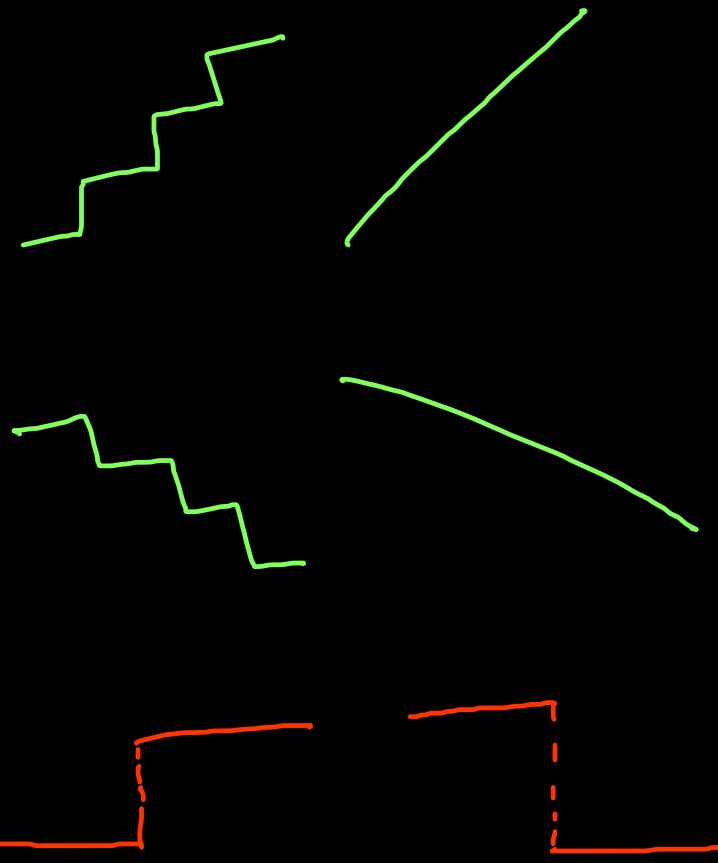
$10 + 6 - 2$

$= 14$

```java
class Solution {
    public long count(long mid, long a, long b, long lcm){
        return (mid / a)  + (mid / b) - (mid / lcm);
    }

    public long gcd(long a, long b){
        return (b == 0) ? a : gcd(b, a % b);
    }

    public int nthMagicalNumber(int n, int a, int b) {
        long lcm = (a * b) / gcd(a, b);
        long left = 0, right = Long.MAX_VALUE;

        while(left <= right){
            long mid = left + (right - left) / 2;

            if(count(mid, a, b, lcm) < n)
                left = mid + 1;
            else right = mid - 1;
        }

        return (int)(left % 1000000007);
    }
}
```

Binary Search on Answer

$O(\log 10^{16})$



Matrix Exponentiation — N^{th} fibonacci no.

Leetcode 509

$f(0) = 0$, $f(1) = 1$, $f(n) = f(n-1) + f(n-2)$

0, 1, 1, 2, 3, 5, 8, ----->

0 1 2 3 4 5 6

(1) Recursive Sol^n

TC = $O(2^N)$, SC = $O(N)$

Exponential

```java
// Time = O(2 ^ N), Space = O(N), N max <= 30
public int rec(int n){
    if(n <= 1) return n;
    return rec(n - 1) + rec(n - 2);
}

// Time = O(N), Space = O(N), DP Based (Memo/Tabulation)
// N max <= 10 ^ 6 Due to Array Constraint
public int tabu(int n){
    if(n <= 1) return n;
    int[] dp = new int[n + 1];
    dp[0] = 0; dp[1] = 1;

    for(int i = 2; i <= n; i++){
        dp[i] = dp[i - 1] + dp[i - 2];
    }
    return dp[n];
}
```

```java
// Time = O(N), Space = O(1), DP Space Optimized
// N max <= 10 ^ 8 due to TLE over linear time
public int twoptr(int n){
    if(n <= 1) return n;
    int a = 0, b = 1, c = 1;

    for(int i = 2; i < n; i++){
        a = b; b = c; c = a + b;
    }
    return c;
}
```

Most optimized soln

↳ matrix exponentiation

↳ Time = $O(\log N)$, Space = $O(\log N)$

Nmax = long value

$\approx 10^{15}, 10^{16}$

fibonacci → matrix exponentiation

recurrence relation

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}$$

$n=2$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{1} = \begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix}$$

$n=3$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{2} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} F_3 & F_2 \\ F_2 & F_1 \end{bmatrix}$$

$n=4$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^3 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} F_4 & F_3 \\ F_3 & F_2 \end{bmatrix}$$

$n=21$

$$M^{20} = \begin{bmatrix} F_{21} & F_{20} \\ F_{20} & F_{19} \end{bmatrix}_{2\times 2}$$

$$M^{10} \times M^{10} \Rightarrow M^5 \times M^5 \Rightarrow M^4 \times M \Rightarrow M^2 \times M^2 \Rightarrow M^1 \times M^1$$

$\log 20$

```java
public int[][] mult(int[][] m1, int[][] m2){
    int n = m1.length;
    int[][] res = new int[n][n];

    for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
    for(int k = 0; k < n; k++)
        res[i][j] += m1[i][k] * m2[k][j];

    return res;
}

public int[][] power(int[][] m, int n){
    if(n <= 1) return m;
    if(n % 2 == 1)
        return mult(power(m, n - 1), m);

    int[][] res = power(m, n / 2);
    return mult(res, res);
}

public int fib(int n) {
    if(n <= 1) return n;
    int[][] m = {{1, 1}, {1, 0}};
    m = power(m, n - 1);
    return m[0][0];
}
```

is square matrix

$m_{2\times 2} * m_{2\times 2}$

$O(1)$

$O(\log n)$

Time

$n = 5$

$$\begin{bmatrix} F_5 & F_4 \\ F_4 & F_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{5-1}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$$

linear recurrence relation

$$f(n) = c_1 * f(n-1) + c_2 * f(n-2) + c_3 * f(n-3) + \cdots + c_k * f(n-k)$$

Matrix exponentiation

$$F_i_{k\times 1} = T_{k\times k} * F_{i-1}{}_{k\times 1}$$

$$\Rightarrow F_N = T^{N-1} * F_1$$

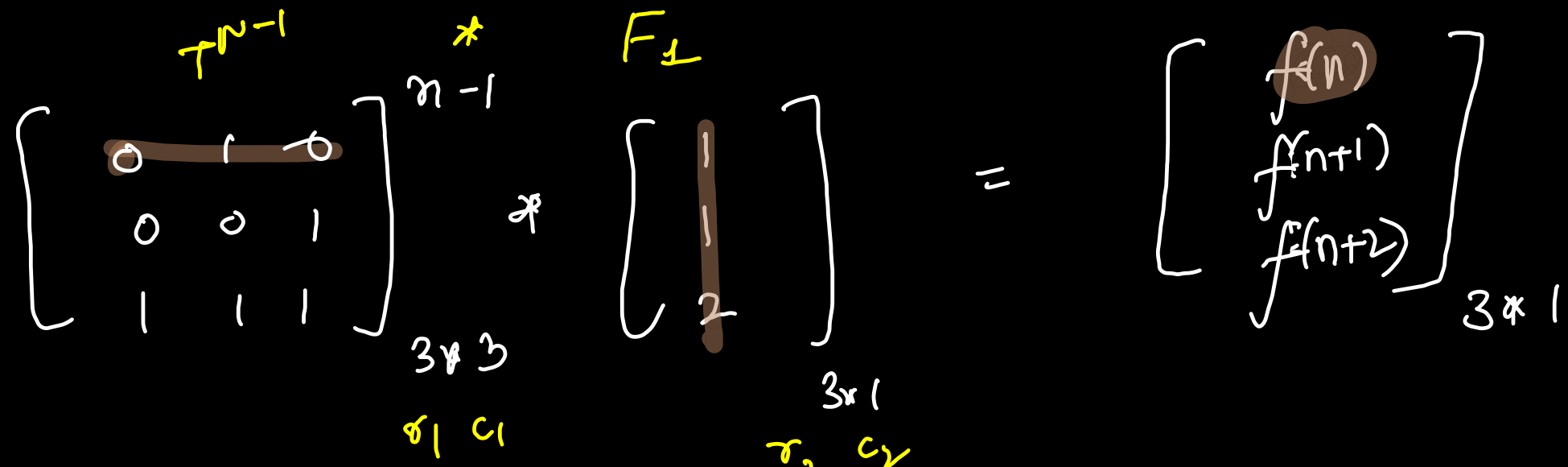
$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_K & c_{K-1} & c_{K-2} & c_{K-3} & \cdots & c_1 \end{bmatrix}$$

T = Transition matrix

fibonacci

$$= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}_{3\times 3}$$

$$F_1 = \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ \vdots \\ f(k) \end{bmatrix} \qquad F_i = \begin{bmatrix} f(i+0) \\ f(i+1) \\ f(i+2) \\ \vdots \\ f(i+k) \end{bmatrix}_{k\times 1}$$




```java
public int tribonacci(int n) {
    if(n == 0) return 0;
    if(n == 1 || n == 2) return 1;

    int[][] t = {{0, 1, 0}, {0, 0, 1}, {1, 1, 1}};
    t = power(t, n - 1);

    int[][] f1 = {{1}, {1}, {2}};

    int res = 0;
    for(int i = 0; i < 3; i++)
        res += t[0][i] * f1[i][0];

    return res;
}
```

$F_1 =$ $\begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ \vdots \\ f(k) \end{bmatrix}_{k*1}$

Time $\Rightarrow O(\log_2 N) * k$

Space $\Rightarrow O(\log_2 N) * k$

Throwing Dice

$n = 4$

1+1+1+1

1+1+2

1+2+1

2+1+1

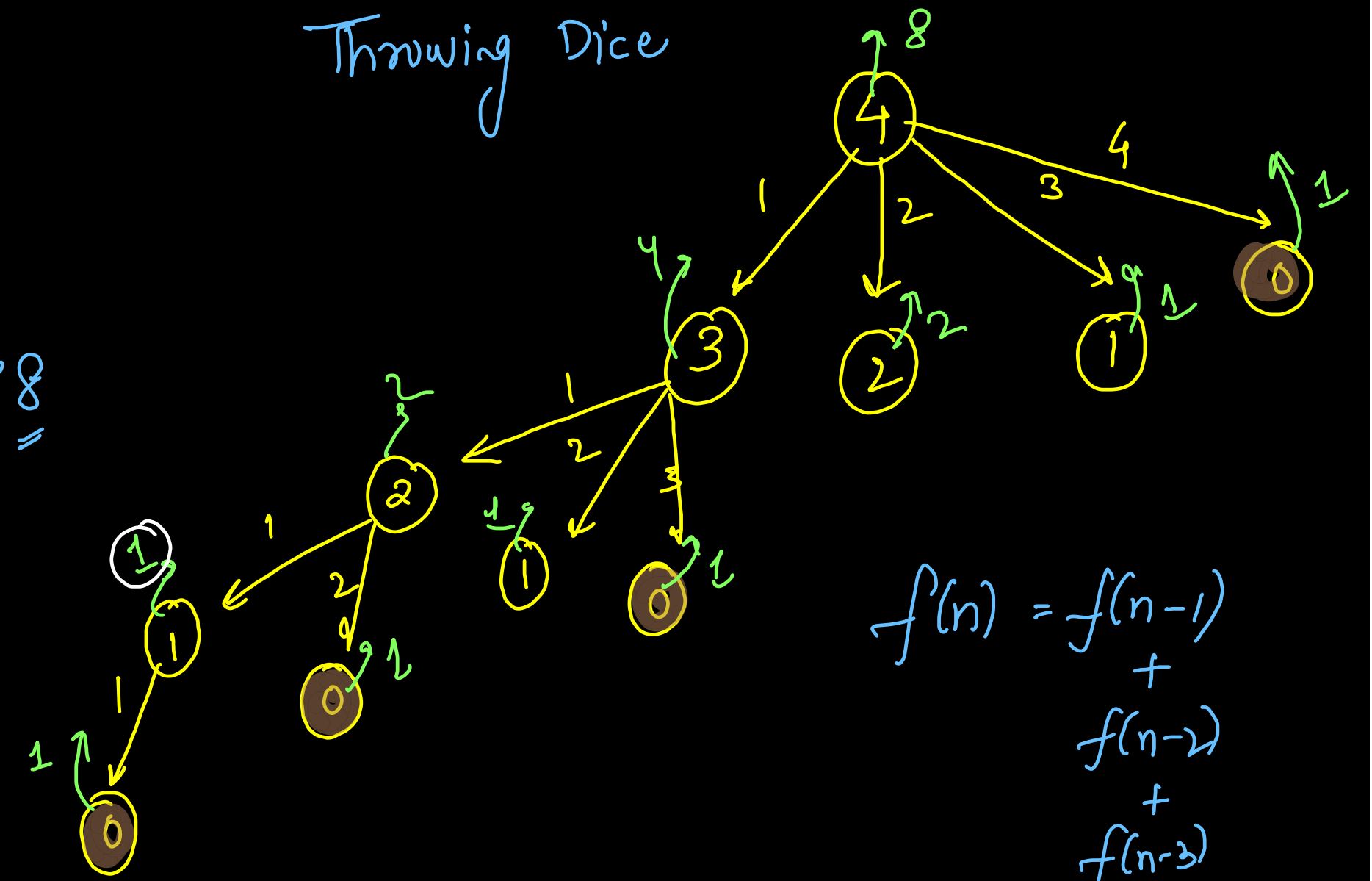
2+2

3+1

1+3

4

8



$$f(n) = f(n-1) + f(n-2) + f(n-3) + \cdots f(n-6)$$

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{n-1} \quad 6 \times 6$$

$$*$$

$$F_1 = \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ \vdots \\ f(6) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 8 \\ ? \\ ? \end{bmatrix} \quad 6 \times 1$$

$$F_n = \begin{bmatrix} f(n) \\ f(n+1) \\ \vdots \\ f(n+5) \end{bmatrix}$$

Facts about Fibonacci

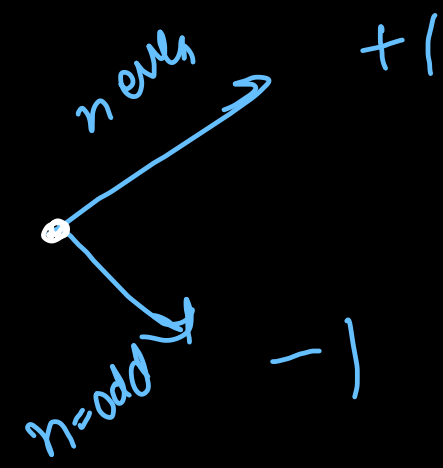
① Fibonacci Last Digits

↳ pattern will repeat after every 60 fibonacci no's

```
static int twoptr(int n){
    if(n <= 1) return n;
    int a = 0, b = 1, c = 1;

    for(int i = 2; i < n; i++){
        a = b; b = c; c = (a + b) % 10;
    }
    return c;
}

static int fib(int N){
    N = N % 60;
    return twoptr(N) % 10;
}
```

② Cassini's identity

$$f(n-1) * f(n+1) - f(n) * f(n) = (-1)^n$$




```
static int fibExpression(int N){
    if(N % 2 == 0) return 1;
    return -1;
}
```

2. **Factors of Fibonacci number :** On careful observation, we can observe the following thing

- Every 3-rd Fibonacci number is a multiple of 2
- Every 4-th Fibonacci number is a multiple of 3
- Every 5-th Fibonacci number is a multiple of 5
- Every 6-th Fibonacci number is a multiple of 8

④ $gcd(n^m fib, m^n fib) = fib(gcd(n,m))$

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

static int fibGcd(int M, int N){
    int g = gcd(M, N);

    int a = 1, b = 1, c = 1;
    for(int i = 2; i < g; i++){
        a = b; b = c; c = (a + b) % 100;
    }
    return c;
}
```

⑤ Fibsum

$N_{max} \leq 10^{15}$

$$F_n = F_{n-1} + F_{n-2} \Rightarrow$$

$$S(n) = f(n+2) - 1$$

```
static long fibSum(long n){
    if(n == 0) return 0;
    if(n == 1) return 1;
    if(n == 2) return 1;

    long a = 0, b = 1, c = 1;

    for(long i = 2; i < n + 2; i++){
        a = b; b = c; c = (a + b) % 1000000007;
    }

    return c - 1;
}
```

$$F_{n-1} = F_n - F_{n-2}$$

$$F_{n-2} = F_{n-1} - F_{n-3}$$

$$F_{n-3} = F_{n-2} - F_{n-4}$$

$$F_{n-4} = F_{n-3} - F_{n-5}$$

$$\vdots$$

$$F_1 = F_2 - F_0$$

$$S(n) - F(n) = F(n+1) - 1$$