Prime Numbers : Only 2 factors

↳ 1 & itself : 5, 9, 11, 1 — Eg: not prime ; 9: 3 factors

count factors
- == 2 : is prime
- != 2 : not prime

[1, → N] <, >, <=, >=, !=, ==

```
bool isPrime(int N){
    int c=0;
    i=1; i<=n; i=i+1){
        if(N % i == 0){
            c=c+1
        }
    }
    if(c==2)
        return True
    else return False
}
```

TC: O(N)

SC: O(1)

N (24)

i	N/i
1	24
2	12
3	8
4	6

$i <= N/i$

$i^2 <= N$

$\boxed{i <= \sqrt{N}}$

$i : [1, \sqrt{N}]$

N 100

i	N/i
1	100
2	50
4	25
5	20
10	10

```
c=0;
i=1; i<=√N; i=i+1){
    if(N % i == 0){ // i, N/i
        if(i == N/i){c=c+1}
        else { c=c+2}
    }
}
if(c==2) return True
else return False
```

TC: O(√N)

SC: O(1)

1Q] Given N, get all primes from [1 - N]

N: 10 → [1 10] → {2 3 5 7}

N: 20 → [1 20] → {2 3 5 7 11 13 17 19}

```
— getallPrime (int N){
    // Idea 1: For all numbers
       from [1 - N], check if
       it's prime or not

    i = 1; i <= N; i = i+1){
        if (isPrime(i)){
            print(i)
        }
    }
TC: O(N√N)
SC: O(1)
}
```

// Given N=50 : [1 50] get primes

~~1~~ F	2 T	3 T	4 F	5 T	6 F	7 T	8 F	9 F	10 F
11 T	12 F	13 T	14 F	15 F	16 F	17 T	18 F	19 T	20 F
21 F	22 F	23 T	24 F	25 F	26 F	27 F	28 F	29 T	30 F
31 T	32 F	33 F	34 F	35 F	36 F	37 T	38 F	39 F	40 F
41 T	42 F	43 T	44 F	45 F	46 F	47 T	48 F	49 F	50 F

2 – 4

3 – 9

5 → 5 * 2 = 10

5 * 3 = 15

5 * 4 = 20

$\boxed{5 * 5 = 25}$

TODO:

given N, how may primes are there are from [1 - N] ?

→ {check}

Pseudocode:

↦ google TODO → { Seive of Eratosthenes }

```
get all prime(N){
    bool p[N+1] = {T}   // iterate & initialize
    p[0] = p[1] = F
    i = 2; i <= N; i++) {
        if( p[i] == T ){   // if i is prime, all multiples of i not
                           // j = 2i → 3i → 4i → 5i → 6i
            j = 2i; j <= N; j = j+i ){
                p[j] = F
            }
        }
    }
    // all T are our primes
    ↳ iterate & get   TC: O(N)
}
```

i = 2   6   9

largest prime <= N

TC: $N/2 + N/3 + N/5 + N/7 + N/11 + \cdots\ N/p$

$\rightarrow N\left[\underbrace{1/2 + 1/3 + 1/5 + 1/7 + 1/11 \cdots 1/p}_{}\right]$

↳ sum of reciprocals of prime = ? (log (log N))

TC: N * (log (log N))

Idea: $\left\{ \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \cdots \frac{1}{N} \right\} \to \log N$

$$= \int_1^N \frac{1}{N} \longrightarrow \log N$$

$N = 2^{64} \approx 10^{18}$

$$\log_2 (2^{64}) = 64$$

$$\log_2 (\log_2 (2^{64})) = \log_2 64 = 6$$

$\boxed{TC: \ N \log(\log N)}$ $\boxed{SC: O(N)}$

// $2 \to 4$

$3 \to 9$ $\{3 \times 2\}$

marked

$5 \to 25$ $\{5 \times 2, 5 \times 3, 5 \times 4\}$

$7 \to 49$ $\{7 \times 2, 7 \times 3, 7 \times 4, 7 \times 5, 7 \times 6\}$

$i \to (i^* i)$

$p \times 2 \to$

$p \times 3 \to$

$p \times 4 \to$

$\vdots$

$p * p$

— get all prime (N) {

bool p[N+1] = {T} // iterate & initialize

p[0] = p[1] = F

i = 2; i <= √N; i = i+1

→ i : [1, N]

i : √N : N : iteration

prime

if(p[i] == T) {

j = i * i; j <= N; j = j+i) {

p[j] = F

}

}

}

j = i * i; j <= N

j = i * i

i : √N + 1 : j > N; j <= N

prime

no iteration

j = (√N + 1)(√N + 1)

j = (N + 2√N + 1)

i = √N + 2 : no iteration

i = √N + 3 : no iteration

// all T are our prime

iterate & get TC : O(N)

}

TC: √N log (log (N)) → wrong

TC: O(N log(log N)) → {TODO} → next Class Doubt Session

Constant Table

10:38 pm

// Smallest prime factor

10 → 2 5

49 → 7

21 → 3

11 → 11

35 → 5

1, 11

smallest prime factor

// Given N for all Numbers from [1 - N], store it's spf

N : 10 :

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

2nd part: N = 50

if (i != spf[i])

1 1	2 2	3 3	4 2	5 5	6 2	7	8 2	9 3	10 2
11 11	12 2	13	14 2	15 3	16 2	17	18 2	19	20 2
21 3	22 2	23	24 2	25 5	26 2	27 3	28 2	29	30 2
31	32 2	33 3	34 2	35 5	36 2	37	38 2	39 3	40 2
41	42 2	43	44 2	45 3	46 2	47	48 2	49	50 2

// Edge: If something is already don't mark it again?

Step1:

```
int spf[N+1] , // initialization => spf[i] = i

i = 2; i <= √N; i = i+1) {
    if( i == spf[i]) {    -> prime, multiples
        j = i*i ; j <= N; j = j+i) {
            if( spf[j] == j) {    -> not marked till now
                spf[j] = i
            }
        }
    }
}
```

TC: O(N log(log N))   SC: O(N)

Prime factorization: (Representing a number → {multiples of power of unique prime number})

$$\begin{array}{r|l} 2 & 48 \\ 2 & 24 \\ 2 & 12 \\ 2 & 6 \\ 3 & 3 \\ & 1 \end{array}$$

$= 2^4 * 3^1 = \{4+1\} * \{1+1\} = 10$

$$\begin{array}{r|l} 2 & 300 \\ 2 & 150 \\ 3 & 75 \\ 5 & 25 \\ 5 & 5 \\ & 1 \end{array}$$

$= 2^2 * 3^1 * 5^2 = \{2+1\}\{1+1\}\{2+1\} = 18$

// get all factors of N?

given N, Assume below is prime factor

$$p_1^{a_1} * p_2^{a_2} * p_3^{a_3} * \cdot\, p_y^{a_y}$$

$p_1 \neq p_2 \neq p_3 \neq p_y$

TODO

no. of divisors: → ?

$(a_1+1)(a_2+1)(a_3+1) \ldots (a_y+1)$

Sum of all divisors:

TODO:

Product of all divisors:

TODO:

4(Q) Given N for all Numbers from [1 - N] get no: of factors for all number from [1 - N]

N:10 :

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

// N : 500 :

↳ for every number from [1 500] get prime factorization

↳ spf[501] = create this.

read // say n = 360 , list<int> l

2 | 360
2 | 180
2 | 90
3 | 45
3 | 15
5 | 5
1

l.add(2)
l.add(2)
l.add(2)
l.add(3)
l.add(3)
l.add(5)

l : | 2 | 2 | 2 | 3 | 3 | 5 |

$2^3 * 3^2 * 5^1$

$(3+1)(2+1)(1+1)$

→ 24

(break)

Idea:

→ For every number 1 – N, get prime factorization.

→ For all numbers from [1 N] , create spf[ ] ✓
↳ O(N log log(N))

→ i = 1; i <= N; i = i+1) { ✓ → O(N)

list<int> p;

n = i;

while(n > 1)

p.add(spf[n])

n = n/spf[n]

→ $\log_2 N$

string smaller prime factor ✓

N log N

// We have prime factorization in p

TODO

Ex: l = 2 2 2 3 3 5 5

(indices 0 1 2 3 4 5 6)

ans = 1   c = 0   +1 +2 +3 (c=1) +2 +1 (2)

ans = ans * (c+1)

ans = 4

c = 1

ans = ans * (c+1)

ans = 12

c = 1

out of loop

ans = ans * (c+1)

= 12 * (2+1)

12 * 3

= 36

ans[i] = primeFactor(p)

}

overall TC : $(N \log\log N + N \log N) \Rightarrow O(N \log N)$

SC : $O(N)$

prim seive

spf[ ]

prime fact

facts[ ]

TODO

l contains prime factorization

```
int primefact( list<int> l)
    ans = 1; c = 0;
    p = l[0]
    i = 0; i < N; i = i+1) {
        if ( p == l[i]){
            c = c+1
        }
        else{
            // a new prime
            ans = ans * (c+1)
            c = 1;
            p = l[i]
        }
    }
    ans = ans * (c+1)
}
```

TODO

gcd of entire array: TC: O(N)

```
ans = ar[0];
i = 0; i < N; i++){
    ans = gcd(ans, ar[i])
}
```

ans — = ar[i] = : (1) iteration — gcd is same

↳ 1, gcd(1, ar[i]) = 1 iteration

↳ ans should decrease 1

// ans = 64

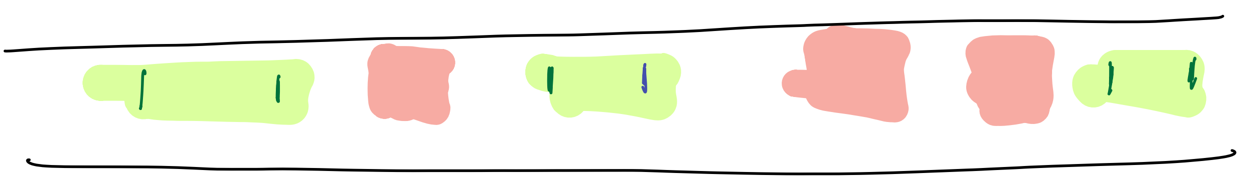
Ex: → 64 64 40 16 24 14 19 21 25

ans: 64 → 64 → 64 → 8 → 8 → 8 → 2 → 1 → 1

gcd(64, 40)
= gcd(40, 24)
= gcd(24, 16)
= gcd(16, 8)
= gcd(8, 0)

gcd(14, 8)
= gcd(8, 6)
= gcd(6, 2)
= gcd(2, 0)
ans = 2

gcd(19, 2)
gcd(2, 1)
gcd(1, 0)
ans = 1



$+ \quad + \quad = O(N) \Rightarrow O(N + \log_2 ar[0])$

$ar[0] \quad \longleftrightarrow \quad + \quad \longleftrightarrow \quad + \quad \longleftrightarrow \quad 1$

ans $\longrightarrow$ ans

$\Rightarrow O(N)$ $\qquad 2^{64}$

$N \approx 10^4$ $\qquad 64$

$ar[0] = 2^{30}$

$N = 2$

$O(\log_2(ar[0]))$

$TC \Rightarrow O(N + \log_2(ar[0]))$

↳ In general for large $N = O(N)$

Doubts:

$(A^{B!}) \% p$ } p is prime, $A < p$, $\gcd(A, P) = 1$

↳ $(A^{P-1}) \% p = 1$

↳ Fermat's Theorem

$\Rightarrow (A^{P-1}) \% p = 1$

$(A^{P-1})^2 \% p = 1$

$(A^{P-1})^3 \% p = 1$

// $B! = $ dividend

$p-1 = $ divisor

$$\underset{A}{B!} = \underset{A}{(q)(p-1) + r}$$

$$A^{m+n} = A^m * A^n$$

$r = B! \% (p-1)$

$$= \left(A^{(q)(p-1)+r}\right) \% p$$

$$= \left(A^{(q)(p-1)} * A^r\right) \% p$$

$$= \left(\underbrace{(A^{p-1})^q}_{1} * A^r\right) \% p$$

$$= (A^r) \% p$$