Prime Numbers

Today's Content:

- Intro to Prime Numbers
- Get all primes from 1 to N
- Print smallest prime factor from 2 to N.
- Prime factorization.
- Number of factors.
- Open doors

Prime numbers: With only 2 factors, $N > 0$

1 → 1 ✗

2 → 1, 2 ✓

5 → 1, 5 ✓

7 → 1, 7 ✓

11 → 1, 11 ✓

Question: Check Prime.

```
func checkPrime (int N){
    count=0;
    for(i=1; i*i<=N; i++){
        if(N%i==0){
            if(i*i==N) count++;
            else count+=2;
        }
    if(count==2) return true;
    else return false;
}
```

T.C = $O(\sqrt{N})$

Question: Given a number N, print all prime numbers from 1 to N.

N = 10 O/P = 2, 3, 5, 7

N = 20 O/P = 2, 3, 5, 7, 11, 13, 17, 19

Brute Force Approach:

- Iterate from 2 to N, check if i is prime

$T.C = O(N\sqrt{N})$ $S.C = O(1)$

Observation for Non-Primes:

2 → Prime
- 4 → No
- 6 → No
- 8 → No

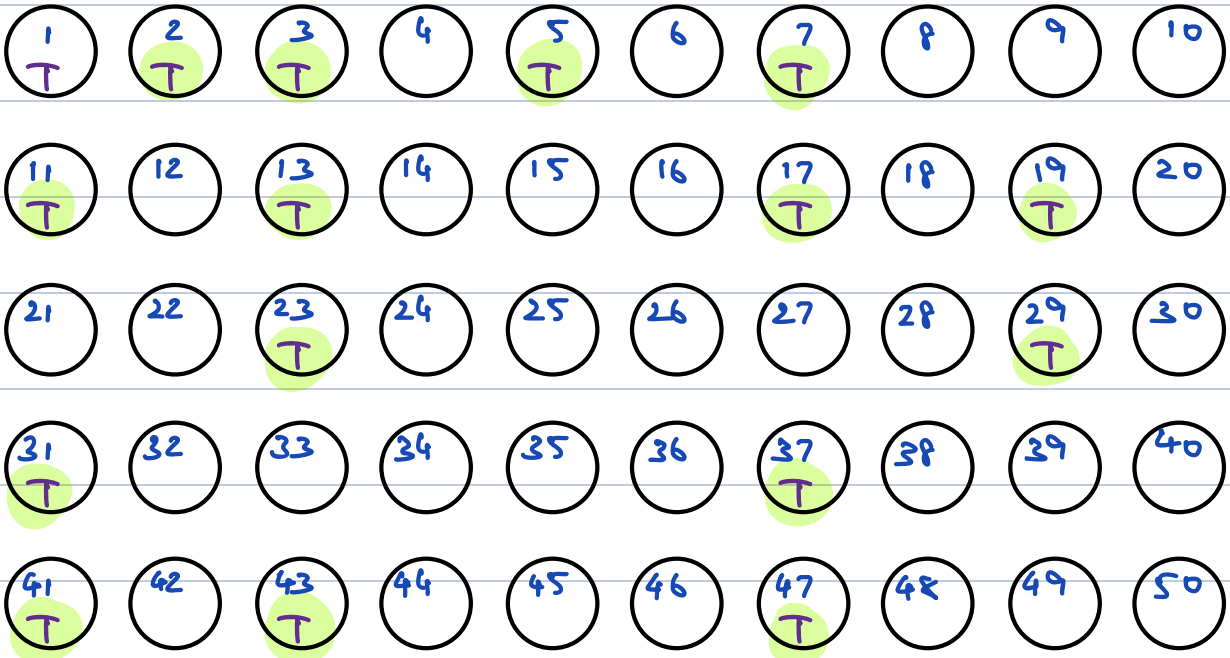
3
- 6
- 9
- 12

5
- 10
- 15
- 25

Multiples of Prime numbers are not primes.

Optimised Approach (Sieve Algorithm)

N = 50 ans = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]




```
func getAllPrimes(int N){
    bool primes[N+1] = {true};
    ans = {};
    for(i=2; i<=N; i++){ → O(N)
        if(primes[i] == true){
            for(j=2; i*j<=N; j++) → O(N)
                primes[i*j] = false;
            ans.add(i);
        }
} return ans; }
```

2 → 4, 6, 8, 10 . . .

3 → 6, 9, 12, 15 . . . .

5 → 10, 15, 20, 25 . . . .

7 → 14, 21, 28, 35, 42, 49

11 → 22, 33, 44,

```
func getAllPrimes (int N){
    bool primes[N+1] = {true};
    ans = {};
    for (i=2; i<=√N; i++){   → O(N log log N)
        if (primes[i] == true){
            for (j=i; i*j <= N; j++)
            |   primes[i*j] = false;
        }
    }
    for (i=2; i<=N; i++){   → O(N)
    |   if(primes[i] == true) ans.add(i);
    }
    return ans;
}
```

i	for N = 50	
2	4, 8, 10 ...	→ N/2
3	9, 12, 15 ...	→ N/3
5	25, 30, 35 ...	→ N/5
7		N/√N

$$\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \cdots \frac{N}{\sqrt{N}}$$

$$N\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \cdots \frac{1}{\sqrt{N}}\right)$$

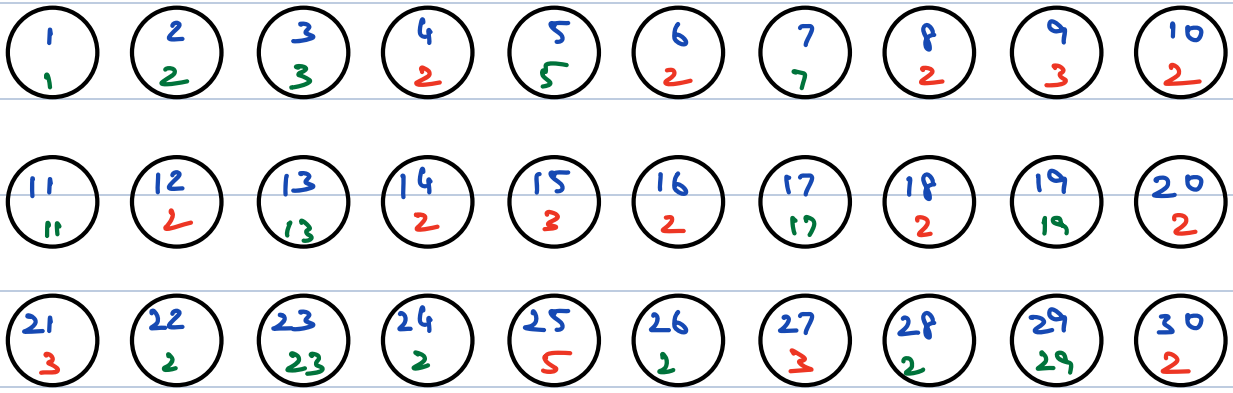
$$= N \log(\log N)$$

$$T.C = O(N \log(\log N))$$

Question: Smallest Prime Factor

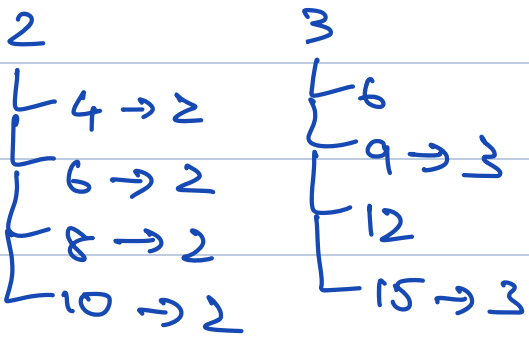
→ Factor which is prime

Given N, return the smallest prime factors for all numbers from 2 to N.



if (spf[i] == i): set multiples of i as spf[i*j] = i

if (spf[i*j] > i) → set.




```
func SPF(int N) {
int spf[N+1] = {i}; // TODO
    for (i=2; i<=√N; i++){
        if (spf[i] == i)  {
            for (j = i ; i*j <= N ; j++){
                if(spf[i*j] > i)  spf[i*j] = i;
            }
        }
}
// pop first two elements and return spf.
```

T.C = O(N log log N) S.C = O(1)

Question: Prime Factorization

Given a number N, return the prime factors of N of the form $N = P_1^a * P_2^b * P_3^c$.
Return the prime factorization in an hash map.

N = 48

2	48
2	24
2	12
2	6
3	3
	1

O/P

2 → ~~1~~ ~~2~~ ~~3~~ 4

3 → 1

Brute Force

- Go over all primes till $\sqrt{N}$
- Keep checking if $N \% i == 0$

```
func primeFactorization(int N) {
    spf = SPF(N); // O(N log log N)
    hm = {};
    while (N > 1) { // log(N)
        if (hm.get(spf[N])) hm[spf[N]]++;
        else hm[spf[N]] = 1;
        N = N / spf[N];
    }
    return hm; }
```

T.C = O(N log log N)

S.C = O(N)

Question: Given a number N, get the number of factors from 1 till N.

N = 10 →	1	2	3	4	5	6	7	8	9	10
O/P →	1	2	2	3	2	4	2	4	3	4

Brute Force Approach:

- Go from 1 to N, count factors.

$T.C = O(N\sqrt{N})$ $S.C = O(1)$

Optimised

$N = P_1^a * P_2^b * P_3^c$

# of factors $= (a+1) * (b+1) * (c+1)$

$N = 45 \rightarrow 3^2 * 5^1$

↳ 1, 3, 5, 9, 15, 45 → 6

$N = 25 \rightarrow 5^2$

↳ 1, 5, 25

$3^2 * 5^1$

3^0 5^0

3^1 5^1

3^2

5^2

5^1

5^0

5^1

5^2

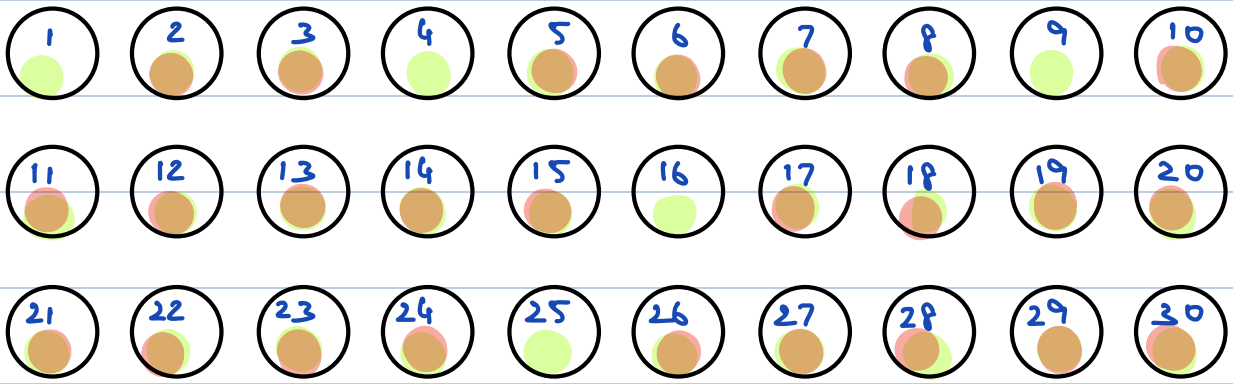
```
func countFactors(int N){
    hm = primeFactorisation(N); // O(N log log N)
    ans = 1;                    // O(N) - Space
    for(i : hm.keys()){         // O(log N)
        ans *= (hm[i] + 1);
    }
    return ans;
}
```

$$T.C = N \log\log N + \log N + \log N$$
($\log N$: primefactors)
$$= O(N \log\log N)$$

$$S.C = N + \log N$$
(N: SPF, $\log N$: primefactors)
$$= O(N)$$

Question: Number of Open Doors

Given a number N, there are N closed doors.
A person moves to and fro and alters the states of the doors (Open → Closed, Closed → Open)
In first go, he alters with 1, 2, 3, 4 . . . N
In second go, he alters with 2, 4, 6, 8 . . . N.
In third go, he alters with 3, 6, 9, 12 . . . . N.
Continues till the Nth go.
Find and return the number of open doors at the end.



Open Clos Op Cl Op Cl — Final State

Door 12 → 1, 2, 3, 4, 6, 12 → 6 → Closed

Door 15 → 1, 3, 5, 15 → 4 → Closed

Door 25 → 1, 5, 25 → 3 → Open

Approach 1:

- Count all factors till N.
- If count is even → Closed else → Open.
- Count only open doors.

$T.C = O(N \log \log N)$

↳ $N \log \log N + \log N + \log N + N$ → Open/Close

SPF ↵ Prime Factors ↳ Count Factors

$S.C = O(N)$ → $N + \log N + N$

SPF ↵ Prime Factors ↳ Count Factors

Optimised

- Count perfect squares from 1 to N.

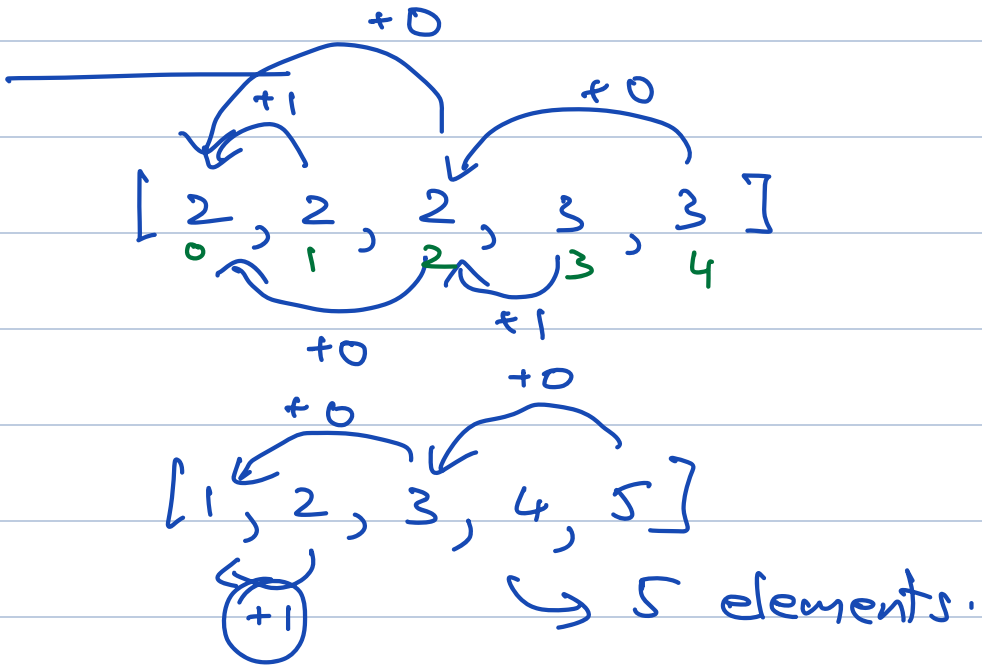
```
func openClosed(int N){
    count = 0;
    for(i=1; i*i<=N; i++){
        count++
    }
    return count
}
```

$T.C = O(\sqrt{N})$

$S.C = O(1)$

Contest → 5th July 2024

- Maths
- Recursion
- Backtracking
- OOPs



# of even indexes > # of odd indexes :

return # of odd

else:

return # of even.