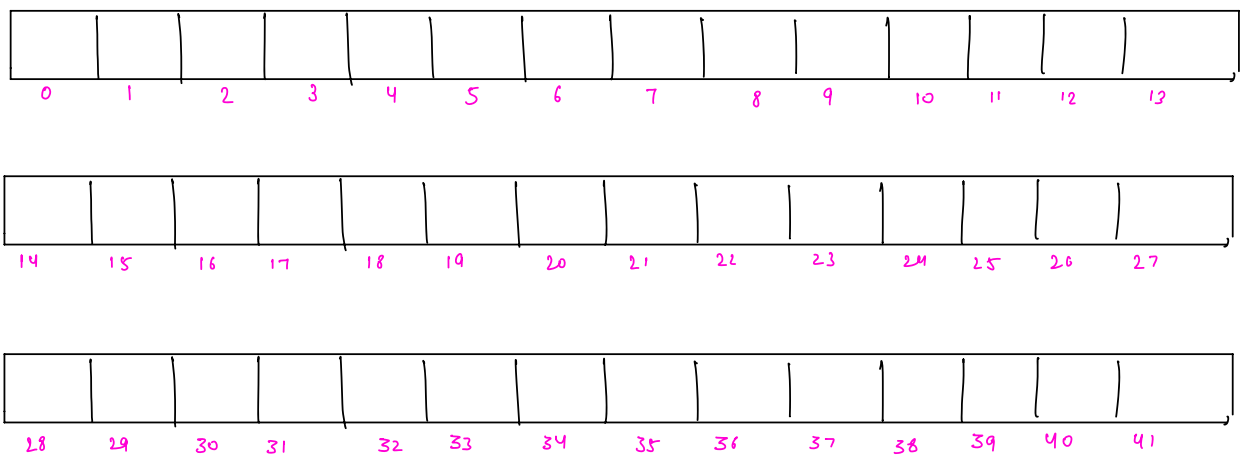
Lecture :- Prime numbers

Agenda

- Revision
- Sieve of eratosthenes
- Count of divisors of all no from 1 to n
- Smallest prime factors
- Count of divisors using smallest prime factor

Class starts at 7:05 AM



Prime no :- factors = 2 [1 and no. itself]

Eg: 2, 3, 5, 7, 11, 13, 17, 19, 23 ---

↳ 1, 7

Qu Given a no. n, check whether it is a prime or not?

```
boolean isPrime(int n) {
    int factors = countFactors(n);
    if( factors == 2) {
        return true;
    }
    return false;
}
```

O(√n) — countFactors(n); → 1st class of intermediate

TC: O(√n)

SC: O(1)

Qu2 Given a no. n, for all no. from 1 to n, check whether it is prime or not?

Eg: n = 10 [2, 3, 5, 7]

ans = 4.

Approach 1:

```
int countPrimes(int n) {
    int cnt = 0;
    for(i = 1; i <= n; i++) {          — O(n)
        if (isPrime(i)) {              — O(√n)
            cnt++;
        }
    }
    return cnt;
}
```

TC: $O(n\sqrt{n})$

SC: $O(1)$

Prerequisite:

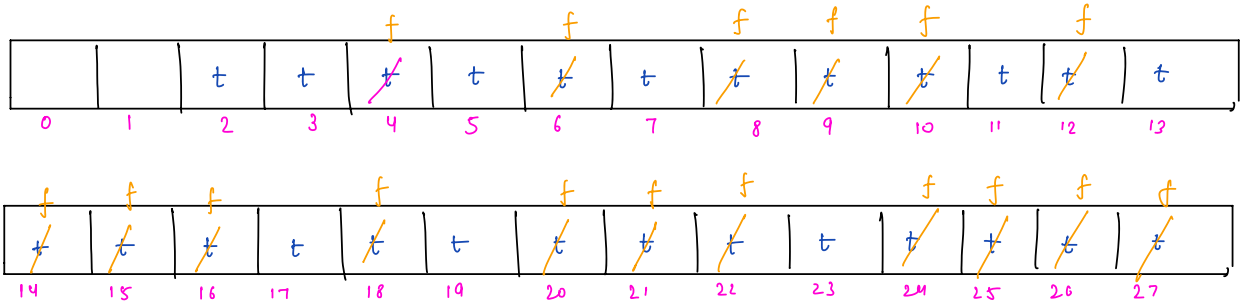
$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} \cdots\cdots \frac{1}{n} = \log n.$

⇓

sum of all reciprocals

$$\int_1^n \frac{1}{n} = \left| \log n \right|_1^n = \log n - \log 1 = \log n.$$

Approach2: n = 27




```
boolean[] sieve (int n) {

    boolean[] sieve = new boolean[n+1];

    for (i=2; i<=n; i++) {

        sieve[i] = true;

    }

    for (i=2; i<=n; i++) {

        if (sieve[i]) {

            for(j= 2*i; j<=n; j=j+i) {

                sieve[j] = false;

            }

        }

    }

    return sieve;

}
```

TC: $O(n \log_2(\log_2 n))$

SC: $O(n)$

TC: $i = 2 : \frac{n}{2}$

$i = 3 : \frac{n}{3}$

$i = 5 : \frac{n}{5}$

$\vdots$

```
for(i=2; i<=n; i++) {
    if (sieve[i]) {
        for(j= 2*i; j<=n; j=j+i) {
            sieve[j] = false;
        }
    }
}
```

TC: $\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} + \frac{n}{11} \cdots\cdots\cdots$

TC: $n\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} \cdots\right) = n\left[\log_2(\log_e n)\right]$

$\Downarrow$

$< \log n$

We know —

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} \cdots\cdots \frac{1}{n} = \log n$$

Small correction

2 : 4, 6, 8, 10, 12, 14 ---- 2^2

3 : 6, 9, 12, 15, 18, 21 ---- 3^2

5 : 10, 15, 20, 25, 30, ---- 35 --- 5^2

7 : 14, 21, 28, 35, 42, 49, 56 ---, ... 119 7^2

11 : 22, 33, 44 ---------- 99, 110, 121 --- 11^2

```
boolean[] sieve (int n) {
    boolean[] sieve = new boolean[n+1];
    for(i=2; i<=n; i++) {
        sieve[i] = true;
    }
    for(i=2; i<=√n; i++) {
        if (sieve[i]) {
            for(j= i*i ; j<=n; j=j+i) {
                sieve[j] = false;
            }
        }
    }
    return sieve;
}
```

TC: $O(n(\log_2 \log n))$ — h/w

SC: $O(n)$

```
int countPrimes(int n) {
    boolean[] sieve = sieve(n);
        int cnt = 0;
        for(i=2; i<=n; i++){
            if(sieve[i]){
                cnt++;
            }
        }
    return cnt;
}
```

TC: $O(n \log_2 \log_2 n)$ ——— SC: $O(n)$

TC: $O(n \log_2 \log_2 n) + O(n) \simeq$

SC: $O(n)$

Qu Given no n, for all no from 1 to n, count factors of all numbers.

Eg $n = 5$

1 : 1
2 : 2 [1, 2]
3 : 2 [1, 3]
4 : 3 [1, 2, 4]
5 : 2 [1, 5]

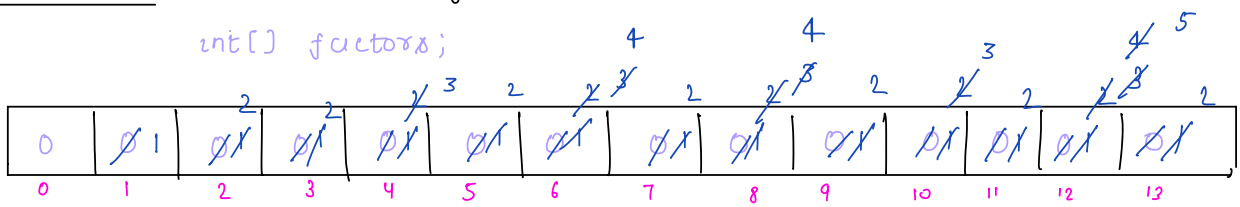
Brute force:

```
void countfactors(int n) {
    for(i=1; i<=n; i++) {
        int factors = countfactors(i);
        print(factors);
    }
}
```

TC: $O(n\sqrt{n})$

SC: $O(1)$

Approach2: TC: O(n log n)



1 : 1 , 2 , 3 , 4 , 5 , 6 , 7 ---- 27

2 : 2 , 4 , 6 , 8 , 10 , 12 .

1 : 1
2 : 2
3 : 2
4 : 3
5 : 2
6 : 4
7 : 2
8 : 3
9 : 2
10 : 3
12 : 5
13 : 2

```
void countFactors(int n){
    int[] factors = new int[n+1];
    for(i=1; i<=n; i++){
        for(j=i; j<=n; j=j+i){
            factors[j]++;
        }
    }

    for(i=1; i<=n; i++){
        int fact = factors[i];
        print(fact);
    }
}
```

TC: O(n log n)

SC: O(n)

i=1: n.

i=2: $\frac{n}{2}$

i=3: $\frac{n}{3}$

i=4: $\frac{n}{4}$

$n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} \text{ ----}$

$n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \text{ ----}\right]$

$\downarrow\downarrow$

n * log n.

Break: 8:30 AM

Qu Smallest prime factor

n = 15, [1 3 5 15]

n = 49 [1 7 49]

Given no. n, for all no from 1 to n. find smallest prime factor.

Brute force:

```
Go from i = 1 to n —                    O(n)
    List<Integer> factors = getfactors(i); — O(√n)
    for(int fact : factors) {           — O(√n)
        if (isPrime(fact)) {
            print(fact);
            break;
        }
    }
```

TC: $O(n\sqrt{n})$

SC: $O(\sqrt{n})$

Approach 2

n = 10 : 2

n = 11 : 11

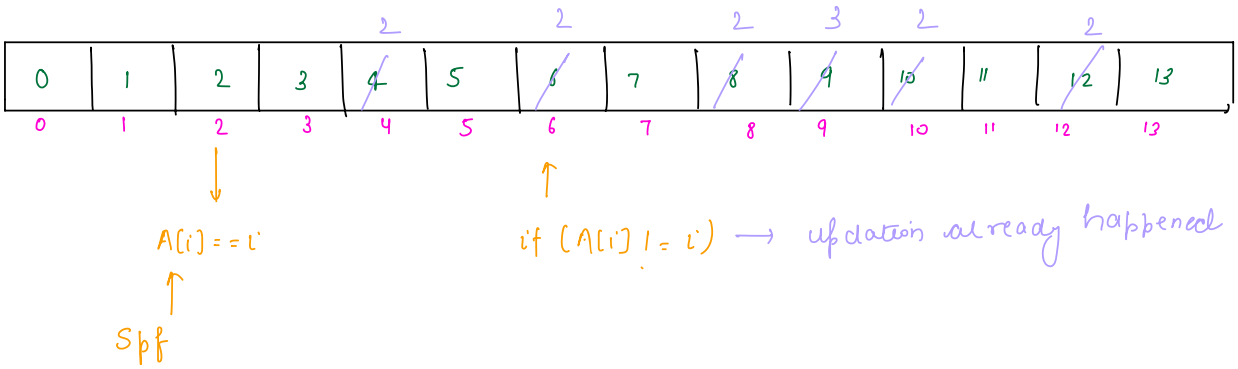
n = 12 : 2

n = 13 : 13

n = 15 : 3

n = 17 : 17

if no and its smallest prime factor are same, prime no [Exception = 1]



```
void smallestPrimeFactors(int n) {
    int[] spfactors = new int[n+1];
    for(i=0; i<=n; i++){
        spfactors[i] = i;
    }
    for(i=2; i<=√n; i++) {
        if ( spfactors[i] == i) {
            for(j = i*i; j<= n; j+=i) {
                if ( spfactors[j] != j) {
                    spfactors[j] = i;
                }
            }
        }
    }
    for(i=2; i<=n; i++){
        print(spfactors[i]);
    }
}
```

TC: $n \log_2 \log_2 n$

SC: $O(n)$

Prime factorization

48:

2	48
2	24
2	12
2	6
3	3
	1

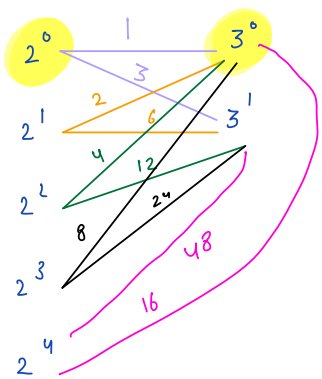
48 :- $2^4 * 3^1$

factors: 10 $[(4+1) * (1+1)]$

45: $3^2 * 5^1$

factors: 6 $[(2+1) * (1+1)]$

48: $2^4 * 3^1$



Generalization:

$n :- p_1^{a1} * p_2^{a2} * p_3^{a3} \cdots p_x^{ax}$

factors: $(a1+1) * (a2+1) * (a3+1) \cdots (ax+1)$

Qu Given no n, for all no from 1 to n, count factors of all numbers.

n = 48 :

$$\frac{48}{\text{spf of } 48} = \frac{48}{2} = 24$$

$$\frac{24}{\text{spf of } 24} = \frac{24}{2} = 12$$

$$\frac{12}{\text{spf of } 12} = \frac{12}{2} = 6$$

$$\frac{6}{\text{spf of } 6} = \frac{6}{2} = 3$$

$$\frac{3}{\text{spf of } 3} = \frac{3}{3} = 1$$

map ⇐ [2 → 1 ~~2~~ ~~3~~ 4 ; 3 → 1]

2 : 4 [2^4]

3 : 1 [3^1]

factors: (4+1) * (1+1) = 10

```
void countfactors(int n){
    int[] spf = smallestprimefactors(n);   // TC: O(n log2 log2 n)  SC: O(n)

    for(i=1; i<=n; i++){                   ------- O(n)

        Map<Integer, Integer> map;

        // prime factorization

        while(i>1){

            int spf = spf[i];

            if(map.containskey(spf)){

                int freq = map.get(spf);

                map.put(spf, freq+1);

            } else{

                map.put(spf,1);

            }

            i = i/spf;

        }

        int factors=1;

        for(int key : map.keyset()){

            int pow = map.get(key);

            factors = factors * (pow+1);

        }

        print(factors);

    }

}
```

i → i/2 → i/4 → i/8 -- → $\log_2 n$.

TC: O(n log n)

SC: O(n)

Thankyou

Doubt:

```
for(i=2; i<=√n; i++) {
    if (sieve[i]) {
        for(j = i*i ; j<=n; j=j+i) {
            sieve[j] = false;
        }
    }
}
```

Aim: Inc your pop by 10%

atleast

$n = 100$

i	j
2	$4, 6, 8 \dots 100 \simeq \frac{n}{2}$
3	$9, 12, 15, 18 \dots \simeq \frac{n}{3} - 1$ (ignored 6)
5	$5, 10, 15, 20, 25, 30 \dots 100 = \frac{n}{5} - 4$
7	$7, 14, 21, 28, 35, 42, 49 \dots 100$
:	
i	$\frac{n}{7} - 6$

$$\frac{n}{2} + \frac{n}{3} - 1 + \frac{n}{5} - 4 + \frac{n}{7} - 6$$

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} - 11$$

$$n\left[\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} \dots\right]$$

TC: $O(n \log_2 \log_2 n)$

$n = 10^{18} \rightarrow ans = 6$

$n = 10^{18}$

$\log_2 10^{18} = \log_2 2^{64} = 64$

$\log_2 64 = \log_2 2^6 = 6$

gcd(120, 240) → ans = 120

$gcd(a, b) = gcd(a, b-a)$ $b > a$

$gcd(120, 240) = gcd(120, 240-120)$ (120)

$= gcd(120, 120)$

$= gcd(120, 0)$

if b == 0, return 120