BASIC MATHS FOR DSA

[I] DIGIT CONCEPT

N = 7789

Perform extraction of digits

7 7 8 9

7789 % 10 = 9 (modulo operator %)
/10 ↳ 778 % 10 = 8
/10 ↳ 77 % 10 = 7
/10 ↳ 7 % 10 = 7
/10 ↳ 0

```
int n;
cin >> n;   OR
while (n != 0) {
    lastdigit = n % 10;
    n = n / 10;
}
```

Pseudo code

[I] Reverse Number

7789 ⟶ 9877

7789 % 10 = 9
7/10 ↳ 778 % 10 = 8

revNo = 0
revNo = (revNo * 10) + lastdigit
(0 × 10) + 9
= 9
(9 × 10) + 8 = 98
(98 × 10) + 7 = 987

Print all divisors

36 → 1, 2, 3, 4, 6, 9, 12, 18, 36
these divide 36.

for loop → Time complexity O(N)

Method 2 – Mathematical observation

36

1	x	36	n/1
2	x	18	(36/2) n/2
3	x	12	(n/3)
4	x	9	n/4
6	x	6	→ √n
9	x	4	
12	x	3	
18	x	2	
36	x	1	

using function would increase time
$i * i \leq n$ ✓
$6 \times 6 \leq 36$
$7 \times 7 \leq 36$ ✗

Pseudo code

```
for (int i=1; i<=sqrt(n); i++) {
    if (n%i==0) {
        print(i)
        if ((n/i) != i) {
            print(n/i);
        }
    }
}
```

prints :- 1 36 2 18 3 12 4 9 6

Prime Numbers

loop till sqrt(n) to check
if n % i == 0

```
int cnt=0
for (i=1; i*i<=n; i++) {
    if (n%i==0) {
        cnt++;
        if ((n/i) != i) {
            cnt++;
        }
    }
}
if cnt == 2 → prime ✓
else → not prime ✗
```

GCD / HCF

- $n1 = 9$ $n2 = 12$

gcd of 20 & 40 is 20

(1) (3) 9 (1) 2, 6, 12, (3), 4

largest no. which divides 9 & 12 → 3 = GCD

Brute force Method

- ~~iterate~~ iterate from 1 to 12 & the no.s that divide both 9 & 12.

pseudo code

```
for (int i=1 ; i <= min(n1,n2) ; i++) {
    if (n1 % i == 0 && n2 % i == 0) {
        gcd = i ; // last no. that gets stored is the gcd.
    }
}
```

∴ Time complexity $O(\min(n_1, n_2))$

Method 2 $n_1 = 20, n_2 = 10$ 20

```
for (int i = min(n1, n2) ; i >= 1 ; i--) {
    if (n1 % i == 0 && n2 % i == 0) {
        gcd = i ;   (20)
        break ;  → breaks from the outer loop
    }
}
```

Time complexity better ✓ $O(\min(n_1, n_2))$ for most cases.

Euclidean Algorithm

$$\boxed{gcd(n_1, n_2) = gcd(n_1 - n_2, n_2) \text{ where } n_1 > n_2}$$

eg. gcd(20, 15) = gcd(5, 15)

5 5

gcd(15, 5) = gcd(10, 5) = gcd(5, 5)

gcd(0, 5) GCD

when one num becomes zero, the other num is the GCD

$\therefore$ gcd(a,b) → gcd(a-b, b) → gcd(... → → 0

a = 52, b = 10

gcd(52, 10) → gcd(42, 10) → gcd(22, 10) → gcd(12, 10) →

gcd(2, 10) → gcd(8, 2)

→ → gcd(2, 2) → gcd(0, 2)

GCD = 2

gcd takes much time (52 % 10, 10) (2, 10)

gcd(a, b) == gcd(a % b, b)

where a > b

till one of them is zero, the other is gcd

Euclidean Algorithm

TC = O(log$_{\phi}$ min(a, b))

Why log?

Note: wherever there is division happening the T.C in terms will be ~~them~~ of log( )

Note: why ϕ?

a % b } since this is
b % a } varying we take TC = log$_{\phi}$( )

if this was 10 it would be TC = log$_{10}$

```
a   b
while (a>0 && b>0) {
    if (a>b)  a = a % b;
else if (b>a)  b = b % a;
}
if (a==0) cout << b;
else if (b==0) cout << a;
```