"GCD"

- → Core concepts
- → Problems

Greatest Common Divisor

↳ largest No that divides both a , b

a = 10 b = 12

gcd (10,12) = 2

gcd (20,12) = 4

gcd (56, 70) = 7

Max value of GCD

↓

min (8 , 12)

Brute Force

gcd = 1

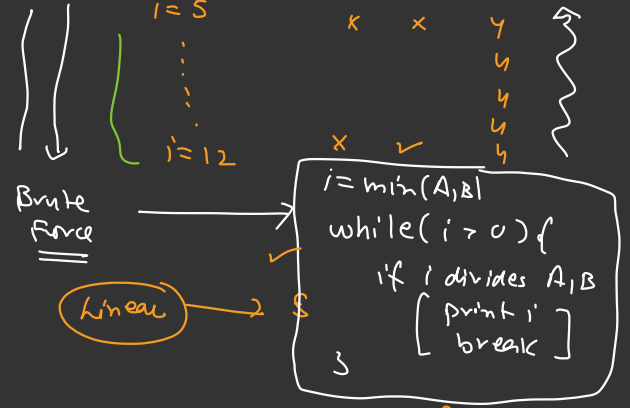
```
for( i=1;  i<= min(A,B) ; i++){
    if(A%i==0  &&  B%i==0){
        gcd = i
```

	a	b	gcd
	20	12	
i=1	✓	✓	1
i=2	✓	✓	2
i=3	x	✓	2
i=4	✓	✓	4

Break

```
    }
}
print(gcd)
```

Time ⇒ $O(\min(A,B))$ → linear in terms of Input N

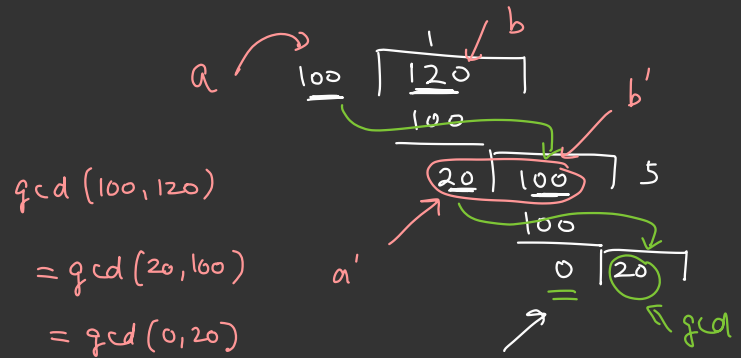


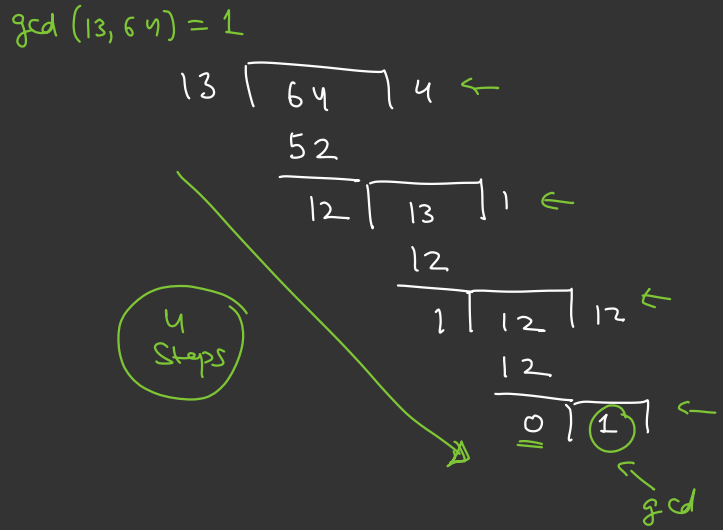
GCD – The School Method (Euclid's Algorithm)

gcd (28, 60)

```
28 | 60 | 2
   56
 4 | 28 | 7
   28
 0 | 4  |
 =   ↑
```

→ gcd





Recursive Case

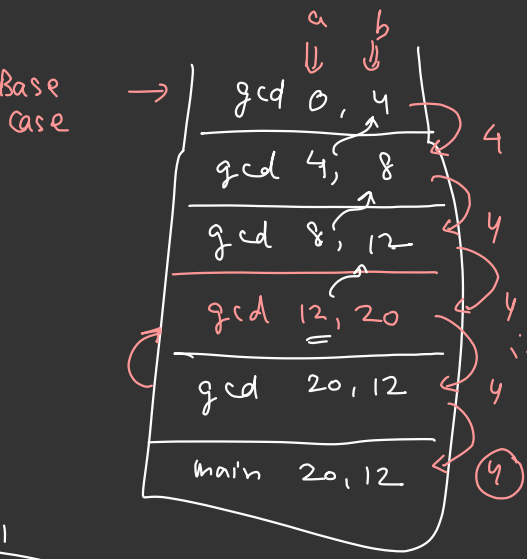
$$gcd(a, b) = gcd(a', b')$$
$$= gcd(b \% a, a)$$

// Euclid's Algorithm

Base Case

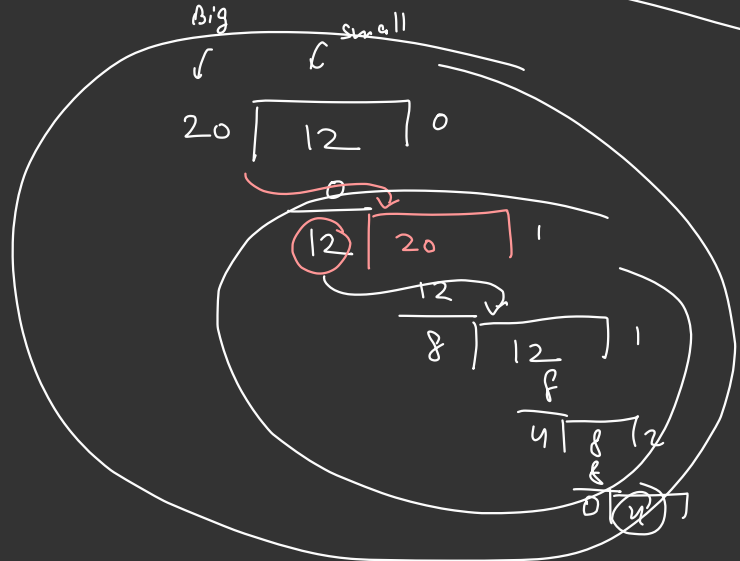
```
if (a == 0)
    gcd = b
    return gcd
```

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



a = 20
b = 12
one extra step

a = 12
b = 20



LCM → Least Common Mutiple

Algo-1

lcm (12, 15) = 66

```
i = 1
while (true) {
    if (i%A == 0 && i%B == 0) {
        break;
    }
    i++;
}
print(i)
```

i=1 12, 15
=2
=3
=4
⋮
=58 12, 15
=59 - -
=60. 12, 15

O(LCM(A, B))

Algo-2

12, 15

30

45

60

60 = 4 steps
15

15 % 12 NO

30 % 12 NO

45 % 12 NO

60 % 12 Yes

LCM → 60

$O\left(\frac{LCM(A,B)}{Max(A,B)}\right)$

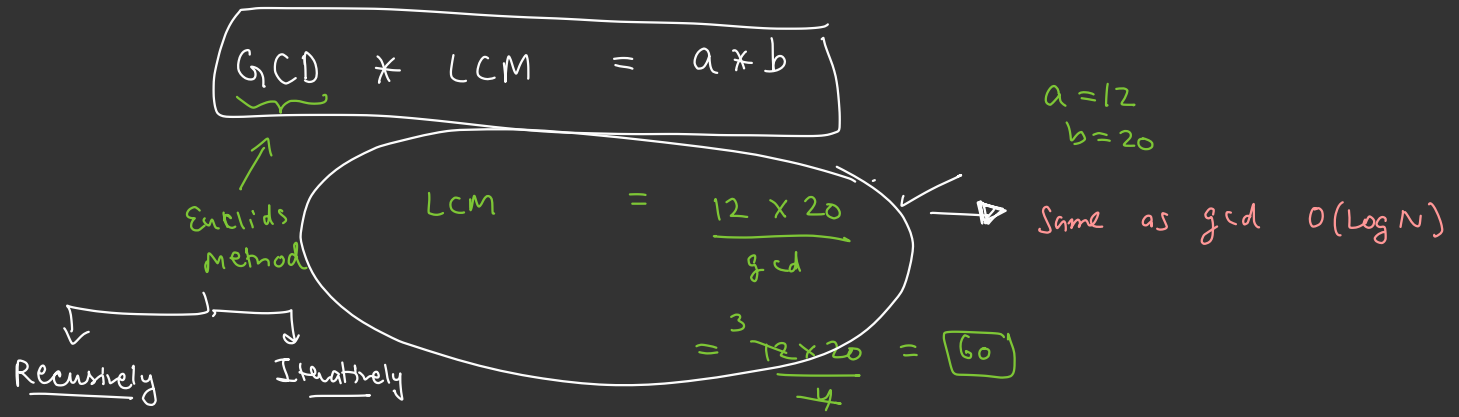
generate multiples of 1 No and check dis by other NO

large = max(A,B) , other = min(A,B)

```
i = large
while ( true ) {
    if ( i % other ) {
        print(i) , break,
    }
    i = i + large ;
}
```

Algo-3

GCD Property (Result)



a = 25 b = 5

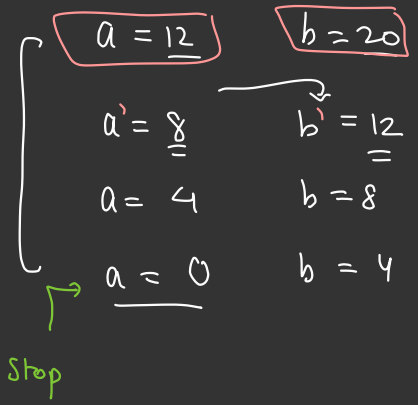
LCM = 25 x 5 / 5 = 25

25 x 1 = 25
5 x 5 = 25
LCM

Iterative Code for GCD →

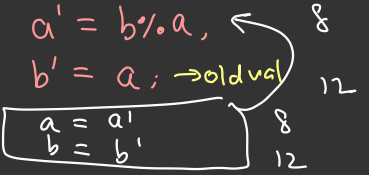
Constant Space O(1)

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



Read a, b,

while(a != 0) {



}

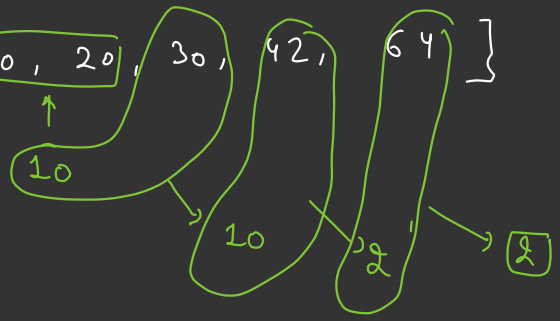
Time Complexity of Euclids → $O(\log(\min(a,b)))$ ⇝ $O(\log N)$ fast

Space → Rec = $O(\log(\min(a,b)))$

↳ Itr = $O(1)$

(2) Given N numbers find their gCd.

N = [10, 20, 30, 42, 64]



Output → 2

Pairwise fashion

2 Mins

```
→ 'n = readNo()
→ gcd = readNo()
  for(i=1; i<n; i++){
→   no = readNo();
→   gcd = getGcd(gcd, no);
→ }
  return gcd;
```

⑤ 10, 20, 30, 64, 42

gcd = 10

gcd(20, 10) = 10

gcd(30, 10) = 10

gcd(64, 10) = 2

gcd(2, 42) = 2

②

$O(n \log(\text{number}))$

10.25 PM

PROBLEMS

Q1 given an array check if there is subsequence with gcd = 1

↓
zero or more

{ 10, 20, 12, 40, 13 }

Examples

gcd = gcd{12, 13} = 1

gcd = gcd{10, 12, 13} = 1

{15, 35} → 1

↓
Yes

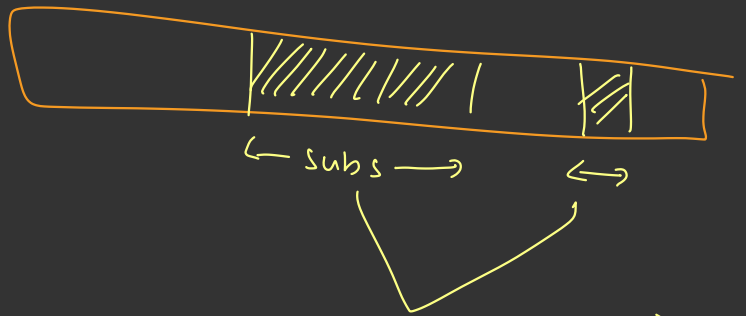
↑
find gcd of array == 1

{ 15, 35, 25, 50 } → NO

{ 9, 8, 6 } → Yes

[9, 8] → 1

Co-prime
Two nos whose gcd 1, they are called co-prime



gcd (subsequence) = 1

gcd (array) = 1

if array gcd is 1, then such a subsequence will always Exist

Examples

{ 6, 12, 3, 18 } gcd = 3 No

{ 16, 10, 6, 15, 27 } gcd = 1 Yes

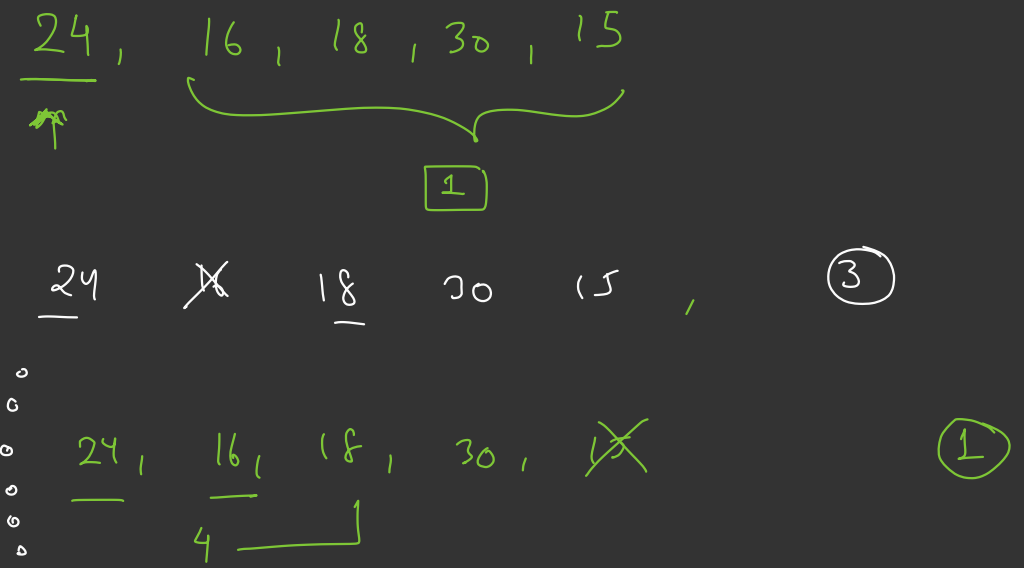
Divisors: 2, 4 | 5 | 3 | 5 | 3

{27, 6}

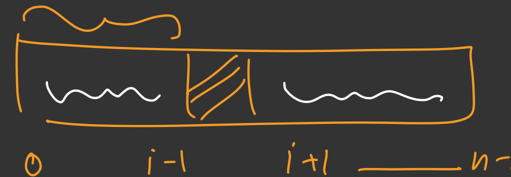
{27, 6, 15}

{ _ _ _ _ _ }

Q Given an array, we can delete 1 element from the array, and take gcd of remaining elements. Find the max gcd we can generate by removing 1 element from the array. Duplicates can be present.



[Brute Force] Compute gcd at every i except a[i]



$gcd_i = gcd(gcd1, gcd2)$

for(—— i ——){
  gcd of remaining array
}

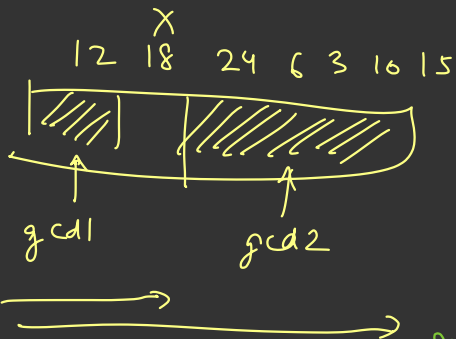
$O(N \cdot N \log N_0)$

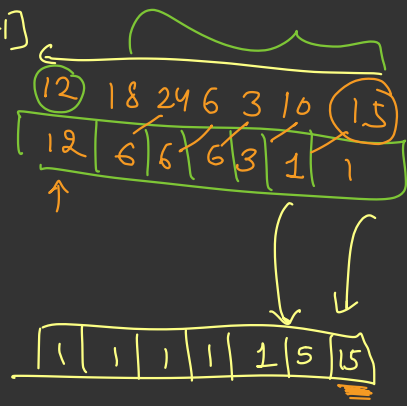
$= O(N^2 \log(N_0))$

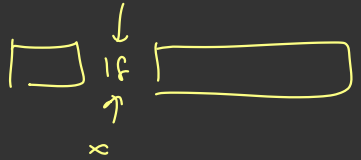
Think...

prefix Gcd [i] = gcd of All elements [0–i]

Suffix gcd [i] = gcd of All elements [i+1 — N-1]







Algo

O(log N

}

ans = 0

Spl case

for (i=0 —— i<=n-1){

Left gcd = pf gcd [i-1], O(1) if i==0 Left gcd = 0,

Right gcd = sf gcd [i+1]; O(1) if i==n-1 Right gcd = 0

CurrentAns = gcd(leftGcd, right gcd), O(log N)

ans = max (ans, CurrentAns).

(I)

```
pfgcd = []
pfgcd[0] = arr[0]
for(i=1 ---- N-1){
    pfgcd[i] = gcd( pfgcd[i-1], arr[i] );
}
```

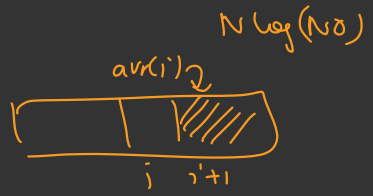
gcd → method

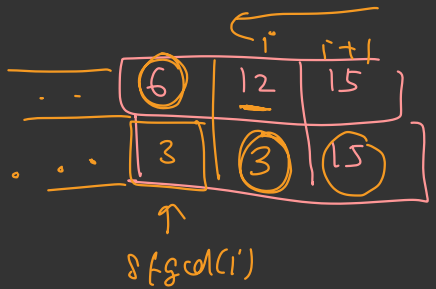
N log(No)

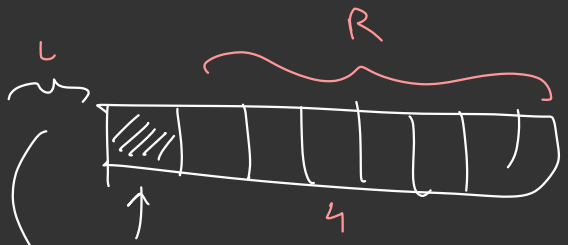
(II)

```
sfgcd = [] , sfgcd[n-1] = arr[N-1]
for( i = N-2;  i>=0 ; i--){
    sfgcd[i] = gcd(sfgcd[i+1], arr[i])
}
```

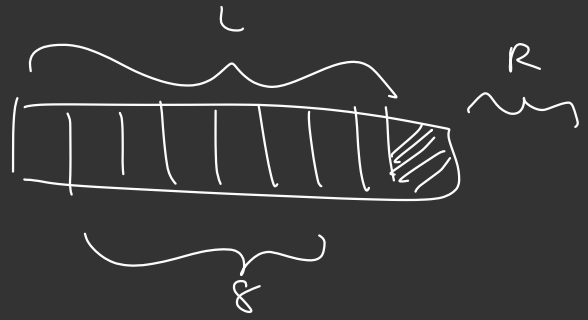
N log(No)





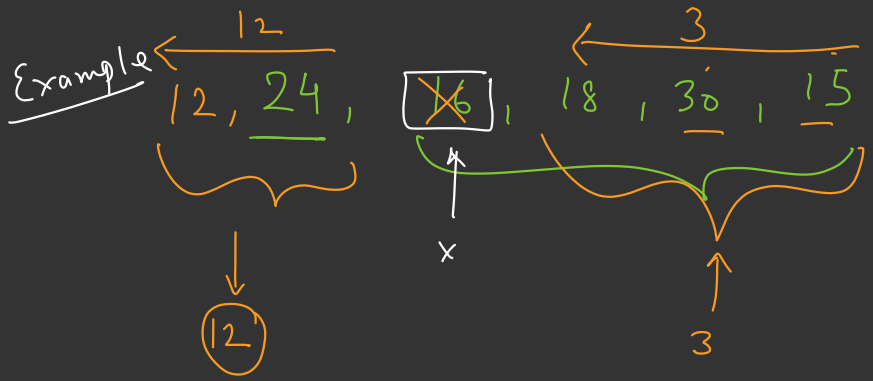


$gcd = gcd(0, 4) = 4$



$gcd = gcd(8, 0)$

$= 8$



gcd (12, 3) = 3

Max Sub Sum

$-1000 \le a[i] \le 1000$

$N = 10^6$

Input

5 3 2 6 8 -4 ...

cs = 5 + 3 + 2 + -6 + ...

```
for (i=0; i<n; i++) {
    cs += a[i];
```

CS, MS

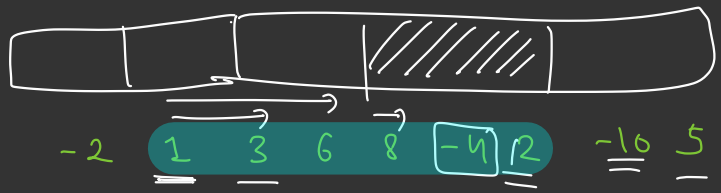
O(N) Time

O(1) Space

```
if(cs<0) { cs = 0;}
ms = max(ms, cs);
}
```

All -ve No's is a spl case.

ms = 0



-ve + +ve ↓

cs = ~~-2~~ 0, 1, 4, 10, 18, 14, 26, 16, 21

ms = ∅, 1, 4, 10, 18, 18, 26, 26, 26