"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 $\Rightarrow$ $O(\min(A,B))$ $\rightarrow$ linear in terms of Input N

i = 5

⋮

i = 12

x x y y y y y

x ✓

Brute Force $\rightarrow$

```
i = min(A,B)
while(i > 0){
    if i divides A,B
        [ print i
          break ]
}
```

Linear $\rightarrow$ 2

GCD – The School Method (Euclid's Algorithm)

gcd (13, 15) $\rightarrow$ $\boxed{1}$

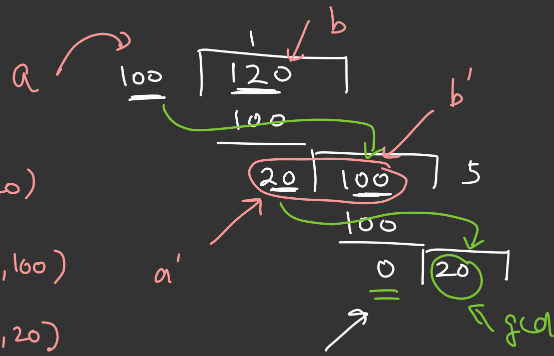
[worst case remains same]

gcd (28, 60)

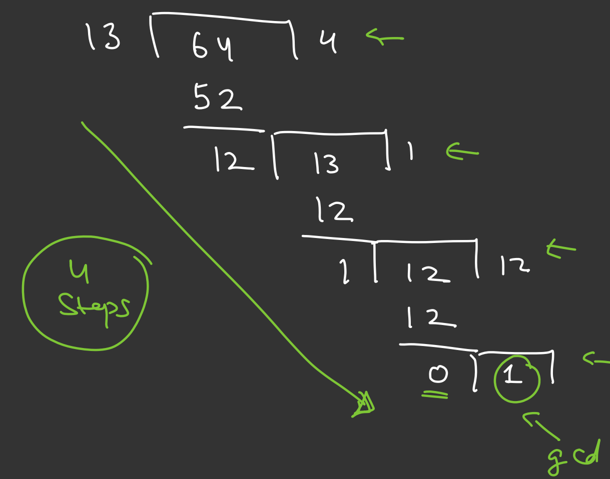
$$
\begin{array}{r|l|l}
28 & 60 & 2 \\
 & 56 & \\
\hline
4 & 28 & 7 \\
 & 28 & \\
\hline
 & 0 & 4 \\
\end{array}
$$

4 $\rightarrow$ gcd

$$\gcd(100, 120)$$
$$= \gcd(20, 100)$$
$$= \gcd(0, 20)$$



$\gcd(13, 64) = 1$



Recursive Case

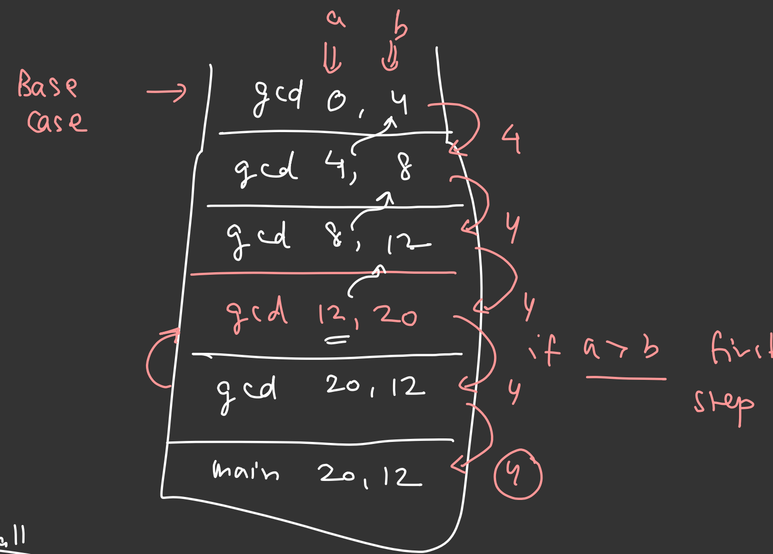
$$\gcd(a, b) = \gcd(a', b')$$
$$= \gcd(b \% a, a)$$

// Euclid's Algorithm

Base Case

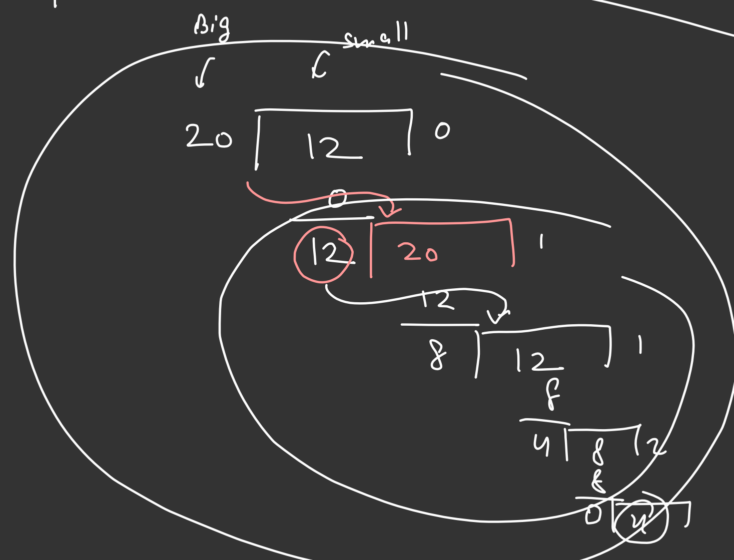
```
if (a == 0)
    gcd = b
    return gcd
```

```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



$\boxed{LCM}$ → Least Common Mutiple

Algo-1

lcm (12, 15) = 66

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

O(LCM(A, B))

↑

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

Algo-2

12, 15

30

45

60

$\frac{60}{15} = 4$ steps

15 % 12 NO

30 % 12 NO

45 % 12 NO

60 % 12 $\boxed{\text{Yes}}$

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)

$$\boxed{\underbrace{GCD}_{} * LCM = a * b}$$

$a = 12$
$b = 20$

Euclids method

Recursively

Iteratively

$$LCM = \frac{12 \times 20}{gcd}$$

$$= \frac{\overset{3}{\cancel{12}} \times \cancel{20}}{\cancel{4}} = \boxed{60}$$

Same as gcd O(Log N)

$a = 25 \quad b = 5.$

$$LCM = \frac{25 \times \cancel{5}}{\cancel{5}} = 25$$

$25 \times 1 = 25$
$5 \times 5 = 25$ $\Big] \boxed{LCM}$

Iterative Code for GCD →

Constant Space $O(1)$

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

$a = 12$ $b = 20$

$a' = 8$ $b' = 12$

$a = 4$ $b = 8$

$a = 0$ $b = 4$

Stop

Read a, b,

while(a != 0) {

$a' = b \% a,$ 8

$b' = a;$ → old val 12

$a = a'$ 8

$b = b'$ 12

}

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

fast

Space → Rec = $O(\log(\min(a,b)))$

↳ Itr = $O(1)$

Q) Given N Numbers find their gcd.

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

Output → $\boxed{2}$

10

10

2 → $\boxed{2}$

Pairwise fashion

$\boxed{\text{2 Mins}}$

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

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

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

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 = {12, 13} = 1

gcd = {10, 12, 13} = 1

Yes

find gcd of array == 1

{15, 35} → 1

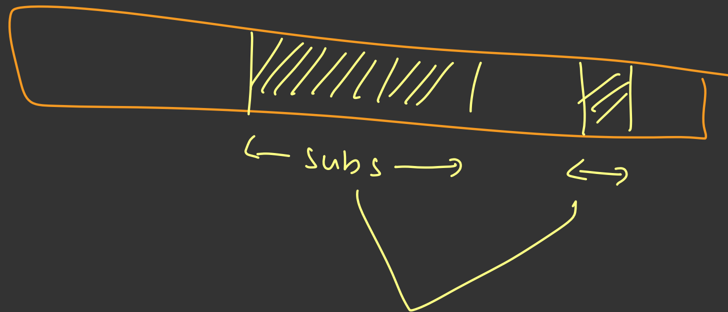
{15, 35, 25, 50} → NO

{9, 8, 6} → Yes

[9, 8] → 1

Co-prime

Two no's 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 $\boxed{\text{Exist}}$

Examples

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

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

Divisors   2, 4   5   3   3   3

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.

$$24, \underbrace{16, 18, 30, 15}_{\boxed{1}}$$

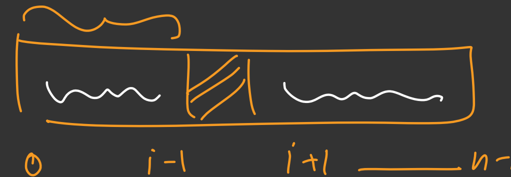
24 $\cancel{16}$ 18 30 15 , 3

24, 16, 18, 30, $\cancel{15}$ 1

4

[Brute Force]

Compute gcd at every i except a[i]



$gcd_i = gcd(gcd1, gcd2)$

for(i){
gcd of remaining array
}

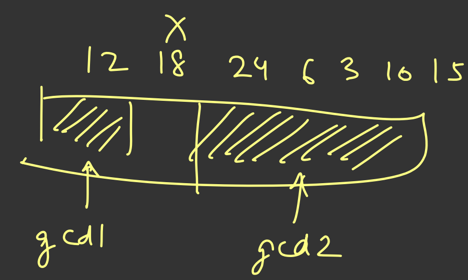
$O(N \cdot N \log No)$

$= O(N^2 \log(No))$

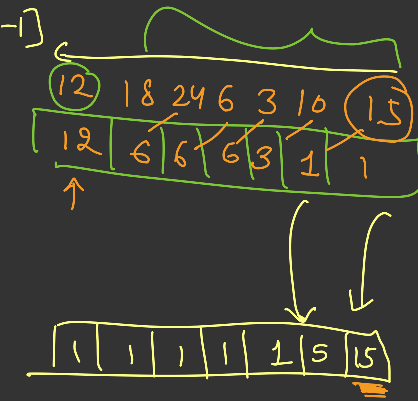
Think...

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

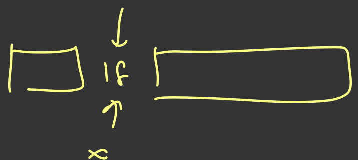
Suffix gcd [i] = gcd of All elements [i+1 — N-1]

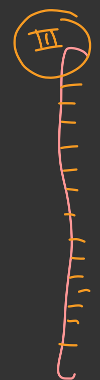


prefix gcd Array



suffix gcd





Algo

O(log N

spl case

ans = 0

for (i=0 ____ i<=n-1){

Left gcd = pf gcd [i-1], O(1) if i==0 leftgcd = 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, current Ans).

}

① pfgcd = []

pfgcd[0] = arr[0]

for(i=1 — N-1){

pfgcd[i] = gcd(pfgcd[i-1], arr[i]);

}

gcd → method

N log(N0)

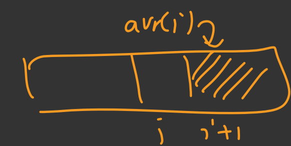
② sfgcd = [], sfgcd[n-1] = arr[N-1]

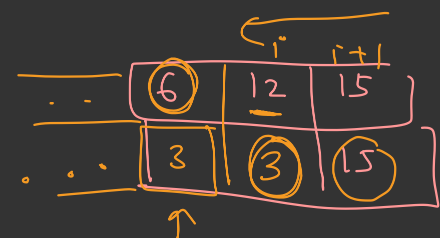
for(i = N-2; i>=0; i--){

sfgcd[i] = gcd(sfgcd[i+1], arr[i])

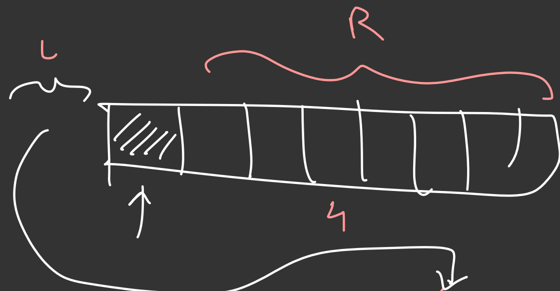
}

N log(N0)

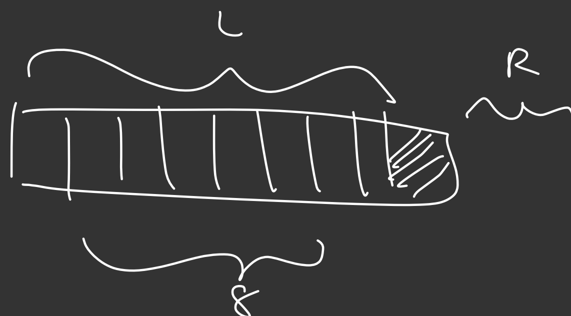




sfgcd(i)



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



$gcd = gcd(8, 0)$

$= 8$

Example

12

12, 24, ~~16~~, 18, 30, 15

3

x

12

3

gcd (12,3) = $\boxed{3}$

Max Sub Sum

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

$N = 10^5$

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);
```

3

All -ve No's is a spl case.

ms = 0

$\boxed{-ve} + \boxed{+ve}$ ↓

-2 1 3 6 8 $\boxed{-4}$ 12 -10 5

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

ms = ∅, 1, 4, 10, 18, 18, 26, 26, 26