GCD (Greatest Common Divisor)

GCD(a,b) = x if $a \% x == 0$ and $b \% x == 0$

(12, 10)

X	12	10
1	1	1
2	2	2
3	3	X
4	4	X
5	X	5

GCD = 2

(20, 10)

X	20	10
1	1	1
2	2	2
5	5	5
10	10	10

GCD = 10

Find gcd (24, 34)

$$
\begin{array}{r}
1 \\
24 \overline{)\,34} \\
24 \\
\hline
10 \overline{)\,24} \\
20 \\
\hline
4 \overline{)\,10} \\
8 \\
\hline
2 \overline{)\,4} \\
4 \\
\hline
0 \overline{)\,2}
\end{array}
\qquad
\begin{array}{r}
2 \\
\\
\\
2 \\
\\
\\
2 \\
\\
\\
\end{array}
$$

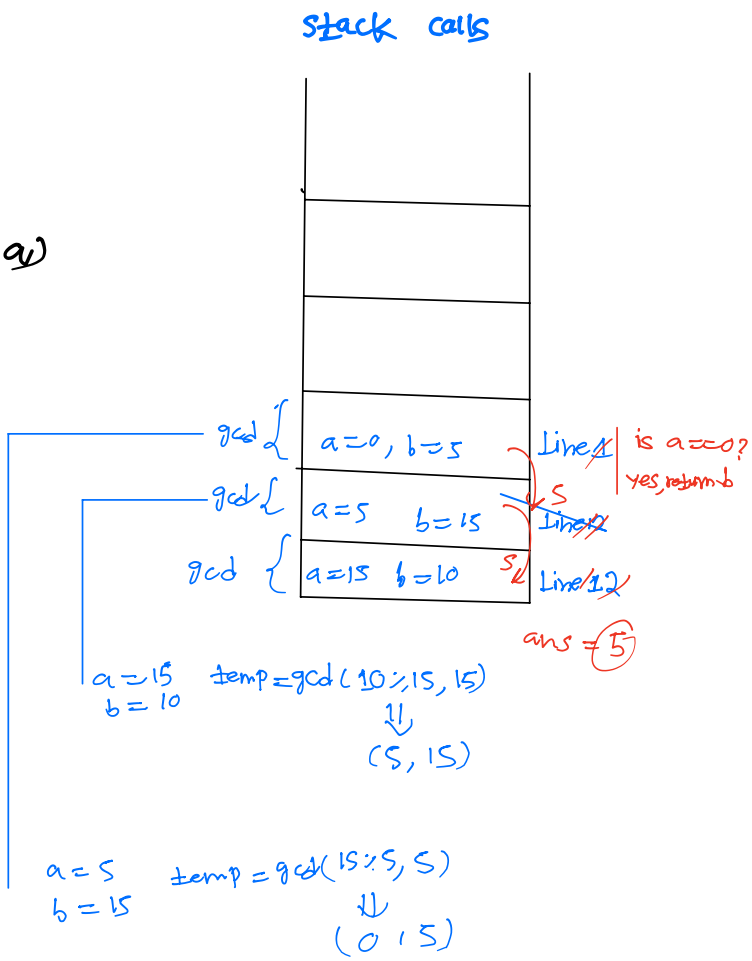
Euclidian Algorithm

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

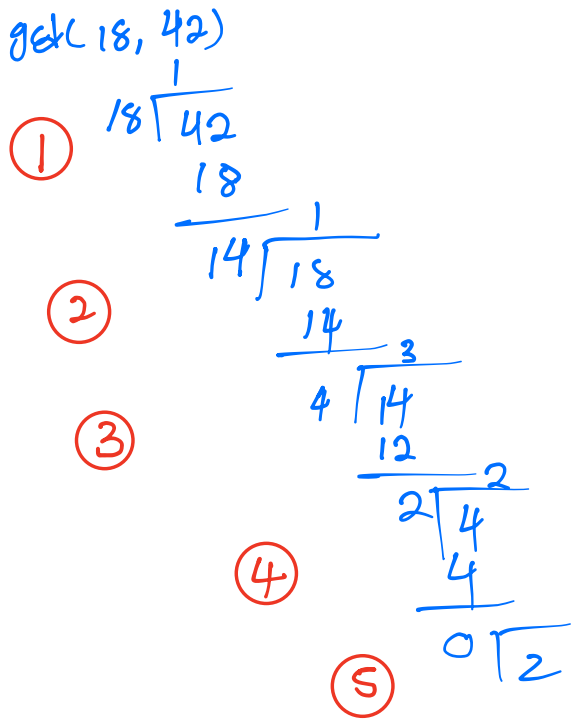
- we have gcd(0,b) = b
- continue this until you get any of the above zero;

```
def gcd(a,b):
    if a==0:
        return b
    temp = gcd(b%a, a)
    return temp
```

find gcd(15, 10)



Analysis: if the number is getting divided by '2' every time, it takes longer until it reaches zero



⇒ max(18, 42)

$5 < \log_2(42) < 6$

⇒ Time complexity $= O(\log_2 N)$

Space complexity $= O(1)$

Number of steps to reach to the point where gcd(a, b) becomes gcd(0, b) $= \log_2 N$

where N is max(a, b)

Q:- Delete to maximize: Given array of N elements, delete one element that results the remaining elements to have maximum GCD.

```
def gcd(a, b):
    if b==0:
        return b
    return gcd(b, a%a)
```

arr = [2 4 6 8]

X [4 6 8] = 2

[2 6 8] X = 2

[2 4 8] = 2

X [6 4 8] = 2

```
ans = arr[0]
for i in range(1,n(arr)):
    ans = gcd(ans, arr[i])
```

How to neglect an element?

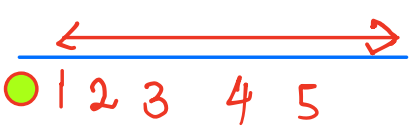
First neglect 0 ele;

second neglect 1, but 0 should be

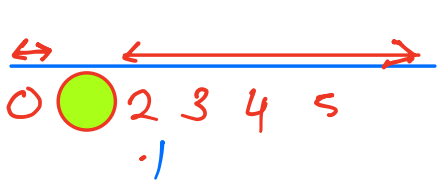
third neglect 2, but 0, 1 should be there

fourth neglect 3rd, but remaining element should

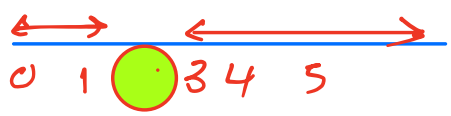
should be there



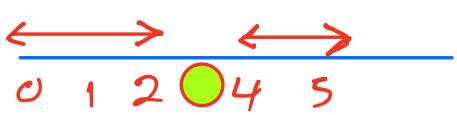
(1, n)



(0, 1), (2, n)

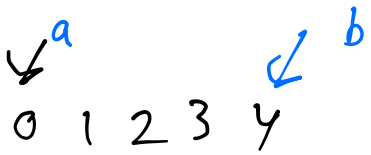


(0, 2), (3, n)



(0, 3) and (4, n)

we can have two variables
(0, i+1), (i+1, n)
before, after



a = 0, 0+i
b = i+1, n

to take arrays, but not including i

temp_arr = arr[:i] + arr[i+1:]

GO for all elements

```
for i in range(n):
    temp_arr = arr[:i] + arr[i+1:]
    ans = arr[0]
    max_gcd = 0
    // find the GCD for each temp_arr;
    for j in range(1, len(temp_arr)):
        ans = self.gcd(ans, temp_arr[j])
        if ans > max_gcd:
            max_gcd = ans
    return max_gcd, temp_arr:
```