DAY-12

Find-GCD [Great Common Divisor]

Naive to Euclid

→ gcd (m,n)

→ Largest k such that k devides m and k diviides n

→ e.g gcd (8,12) = 4

→ gcd (18,25) = 1

→ 1 divides every number.

→ at least one common divisor for every m,n

Computing gcd (m,n)

→ list out factors of m

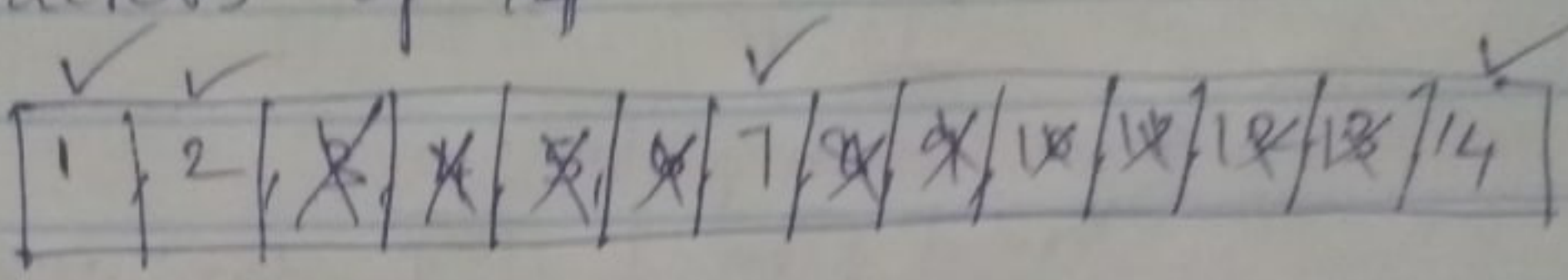
→ List out factors of n

→ Report the largest number that appears in both.

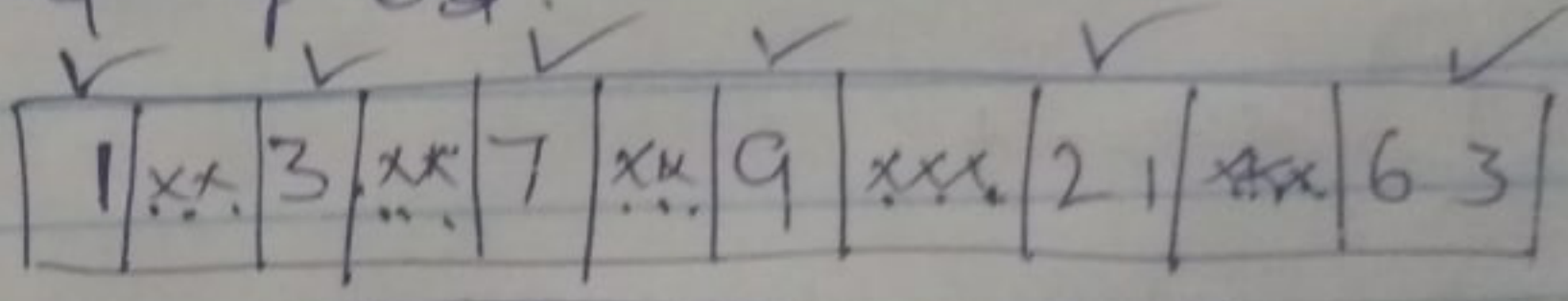
* Note :- May not be good for all the queries.

→ Factors of m must be between
1 and m.
→ Test each number in this range.
→ If it dividing m without
remainder, add it to list of factors.
→ Example: gcd (14, 63)

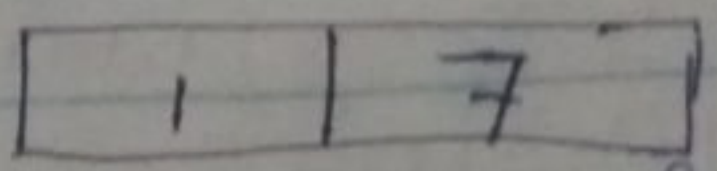
→ Factors of 14



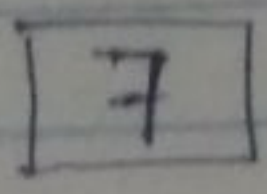
→ Factors of 63:



→ Construct list of common factors
→ For each factor of 14,
check if it is a factor of 63.



→ Return the largest factor in the list:



→ this is the gcd.

An algorithm for gcd (m,n)

→ Use fm, fn for list of factors of m,n

→ For each i from 1 to m,
add i to fm if i divides m

→ For each j from 1 to n,
add j to fn if j devides n

→ Use cf for list of common factors

→ For each f in fm,
add f to cf if f also appears in fn

⇒ Return largest (rightmost if ascending order) value in cf

(1).

```
def gcd(m,n):
    fm = []
    for i in range (1, m+1):
        if(m%.i)==0:
            fm.append(i)
    fn = []
    for j in range(1,n+1):
        if(n%.j)==0:
            fn.append(j)
```

```
cf = [ ]
for f in fm :
        if  f in fn :
                cf. append(f)
return cf[-1]
```

Improving Naive

→ With singlescan from 1 to max(m,n)

→ No need to compute two lists and then comparing

→ if i dividing both(m,n) then add it to cf.

→ any common factor must be less than min(m,n)

→ will go with ⇒ for each i in 1 to min(m,n), instead of 1 to max(m,n).

(2)

```
def gcd (m,n):
cf=[]
for i in range (1, min(m,n)+1):
        if (m %. i)==0 and (n %. i)==0 :
                cf . append (i)
return cf [-1]
```

Without Lists:

```
def gcd(m,n):
    for i in range (1,min(m,n)+1):
        if (m%.i)==0 and (n%.i)==0:
            mrcf = i   #mrcf - most recent
                       # commonfactor
    return mrcf.
```

Scam Backwards

→ When-scam backwards, may get the result faster than before.

```
def gcd (m,n):
    i = min(m,n).
    while i>0:
        if (m%.i)==0 and (n%.i)==0:
            return(i)
        else:
            i = i-1
```

Euclid's Algo

→ suppose d divides both m and n; and m>n

→ Then m=ad, n=bd

→ So m-n = ad-bd = (a-b)d

→ d devideg m-n as well.

→ so gcd(m,n) = gcd(n,m-n)

to code:
→ consider gcd(m,n) with m > n
→ If n devides m, return n
→ otherwise, compute gcd(n, m-n)
and return that value.

```
def gcd(m,n):
    if m < n:
        m,n = n,m
    if m%n == 0
        return(n)
    else:
        diff = m-n
        return(gcd(max(n,diff),min(n,diff)))
```

Still, It Can Be Improved;
I left here you to
practice.