Lecture - 9

Part-I

Functions

1. GCD → efficient method
   - ✓ Multiple Inputs
   - ✓ LCM
   - ✓ Bonus: GCD of N Numbers

2. Binary → Decimal

   Decimal → Binary

Static → every fn till now.

OOPS → ??

Part-II

- Arrays (Data Structure)
- Create
- Input
- output
- Traversal

(Later)

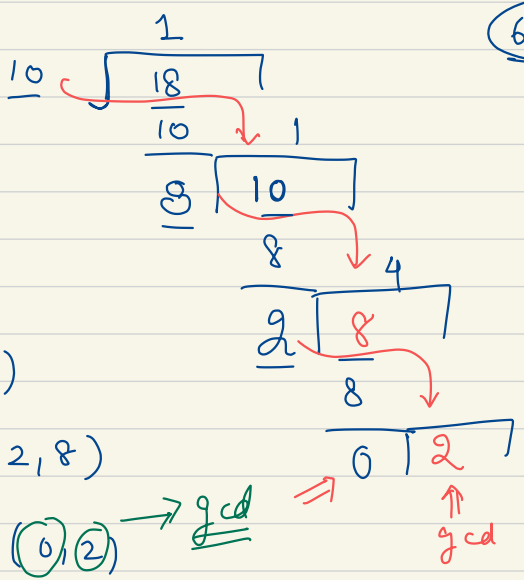
Language Specific Module

- Java (collections,
- JS
- Python
- C#

Greatest Common Divisor

Idea → School level Technique (Long Division)

6th class



gcd(10,18)

= gcd(8,10)

= gcd(2,8)

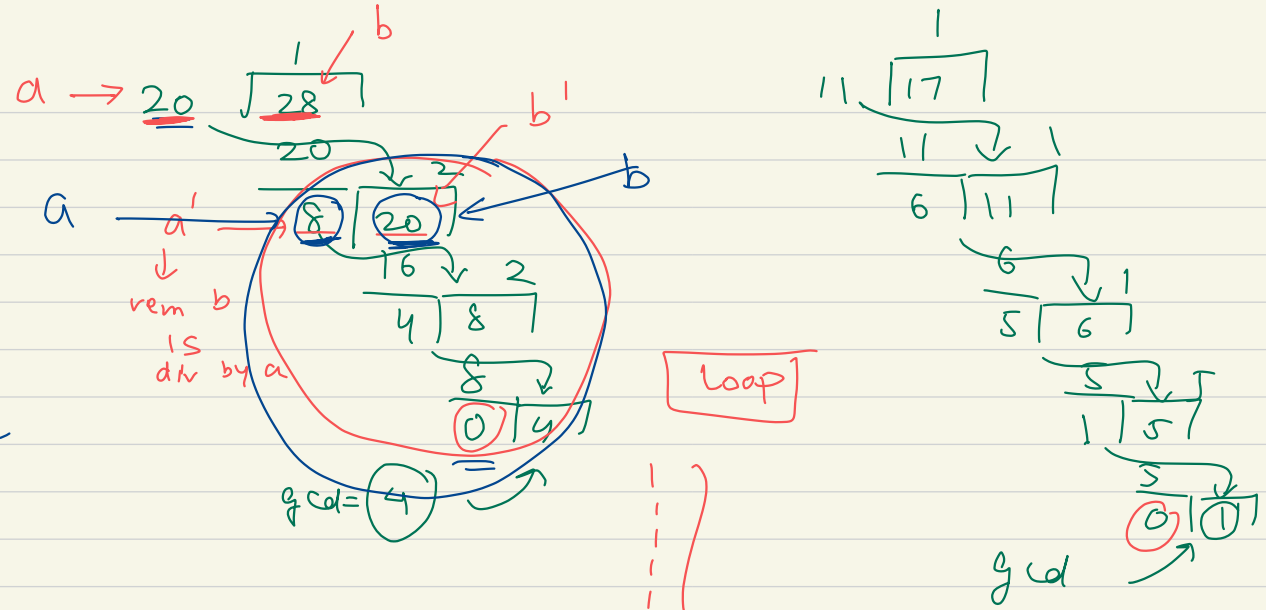
= gcd(0,2) → gcd

Inefficient

	18	10	gcd
i=1	✓	✓	1
i=2	✓	✓	2
i=3	–	–	
i=4	–	–	
i=5	–	–	
i=6	✓	×	
7	–	–	
8	–	–	
9	–	–	
10	–	–	2

equal amt of work.

Stop at first no which divides both



gcd(a, b)

= gcd(a', b')

|| ||

b%a, a

while (a != 0) {

↳ a' = b%a;

b' = a;

a = a' ← update

b = b'

}

print(b) → gcd

gcd(20, 28) = gcd(8, 20)

Long Div
Euclid's Algo for GCD

Step-1 $a! = 0$
$20! = 0$

$a' = 28 \% 20 = 8$
$b' = a = 20$

$a = 20$, $b = 28$

$a = a'$ $\quad b = b'$
$\Rightarrow a = 8$ $\quad b = 20$

Step-2 $8! = 0$

$a' = 4$
$b' = 8$

$a = 4$ $\quad b' = 8$

4 steps

Step-3 $4! = 0$

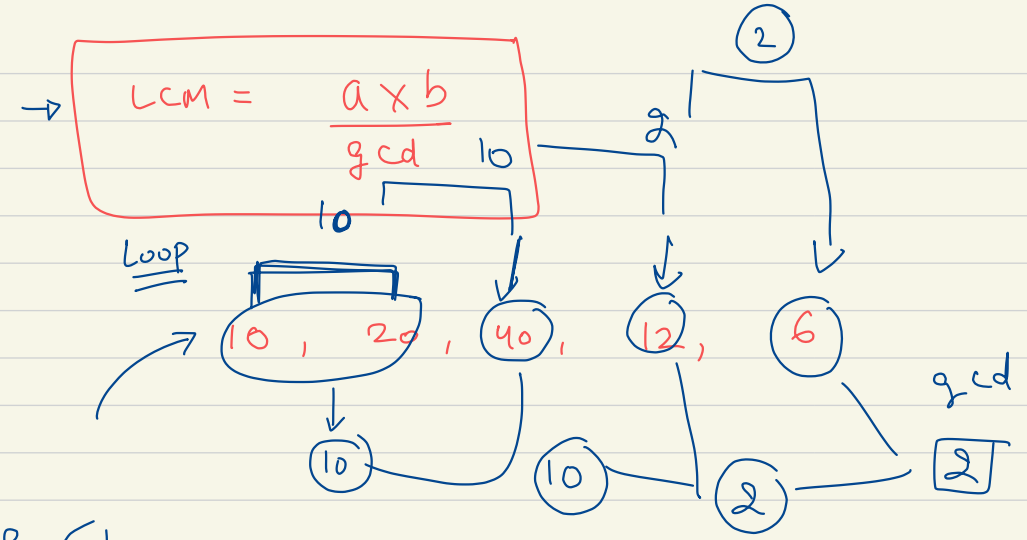
$a' = 0$
$b' = 4$

Step-4 $0! = 0$

Stop

$a = a'$
$= 0$

$b = b'$
$= 4$



how to find it

2

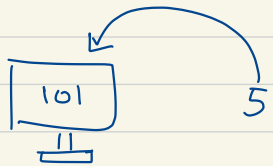
Multiple No's

Loop

ans ~~gcd~~ = gcd (prev_gcd, next_no)

Time $\propto$ Instructions $\propto$ Loop $\uparrow$

Conversions



Computer Memory

000 10101

10 unique digit

Decimal NO 0, 1, 2, 3, 4, ---- 9

10^2 10^1 10^0

1 7 3

Powers of 10

$= 1 \times 10^2 + 7 \times 10 + 3 \times 1$

$= 100 + 70 + 3$

$= 173$

2 uniq digit

Binary NO 0, 1

2^2 2^1 2^0

0 0 1 0 1

16 8 4 2 1

$= 0 + 0 + 4 + 0 + 1$

$= 5$

to decimal

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

2^4 2^3 2^2 2^1 2^0

1 0 1 0 1

$= 16 + 0 + 4 + 0 + 1$

$= 21$

Powers of

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$a^0 = 1$

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 = 1$

1 1 1 0 0 1

32 16 8 4 2 1

$32 + 16 + 8 + 0 + 0 + 1$

$= 57$ ✓

Algorithm:

N = 1 1 1 0 0 1 (32, 16, 8, 4, 2, 1; p = 1)

N % 10 = 1

ans = ans + 1 × 1

p = p × 2 = 2

N = N / 10 = 10

N % 10 = 0

ans = ans + 0 × 2

p = p × 2 = 4

N = N / 10 = 1

N % 10 = 1

ans = ans + 1 × 4

p = p × 2 = 8

N = N / 10 = 0

ans = 0

\+ 1

\+ 0

\+ 0

\+ 8

\+ 16

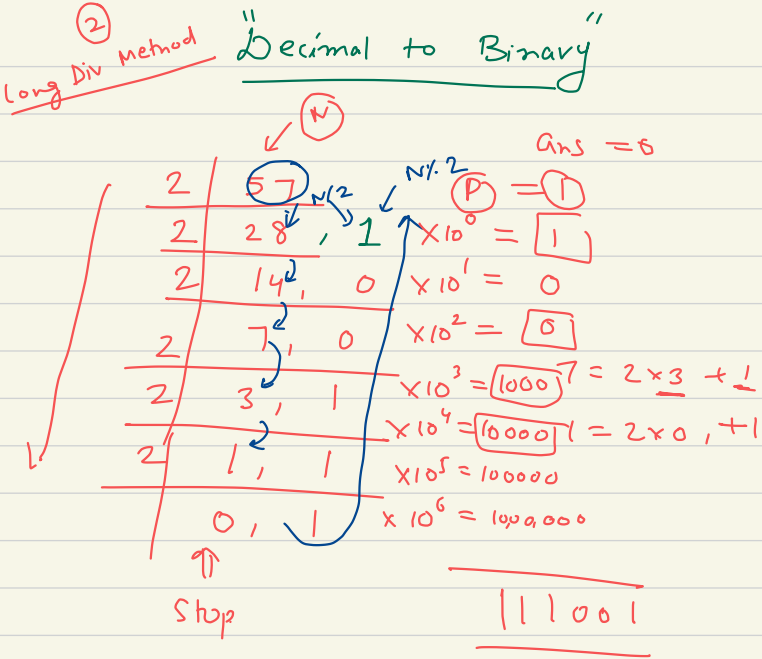
\+ 32

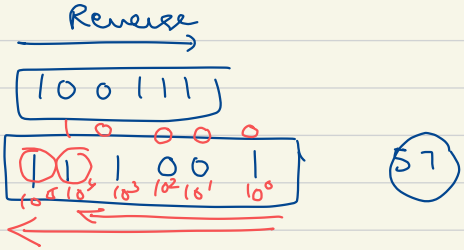
57

```
while (N != 0) {
    ans = (N % 10) * p + ans.
    p = p * 2
    N = N / 10
}
print(ans)
```

(2) Long Div Method

"Decimal to Binary"



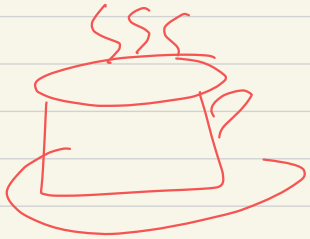



```
while ( N != 0){
    print ( N%2)
    N = N/2;
}
```

3

P = 10

10 = P x 10 = 100 = 10^2 x 10 = 10^3



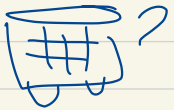
10 40 :)

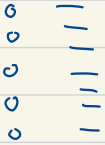
ARRAYS

$n = 10$
$T = 5$
$A, B = 28, 20$

Lot of Data can come as input, you might need to store it.

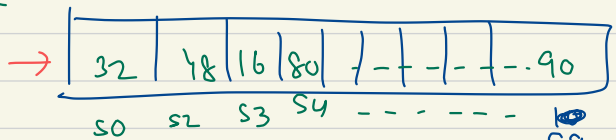
(a) (Rice , Pulses , Fruit ---) ⇐ Storage





ToDo List

Storage



1000 students

- Rank lists

S9 — 90
⋮
S4 — 80
S2 — 48
S0 — 32
S3 — 16

ARRAY

100 items

100 variables its not practical

	Cost
item1 =	102
item2 =	2050
item3 =	168
⋮	⋮
item100 =	____

int S1 = 32
int S2 = 48
⋮

impractical ☹

Container that allows us to store a collection of items under a single variable name.

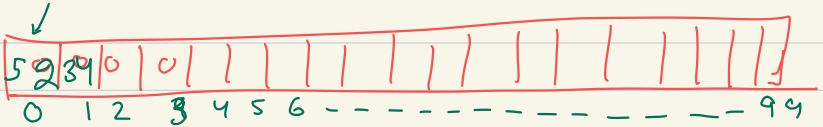
Fixed Size ARRAY

`int n = sc.nextInt();` ← Runtime $n = 100$

`int [] marks = new int [n];` ← Run time Memory Allocation

`int []` — Array Datatype

`[n]` — Possible n user input



⇒ Inside memory (Heap)

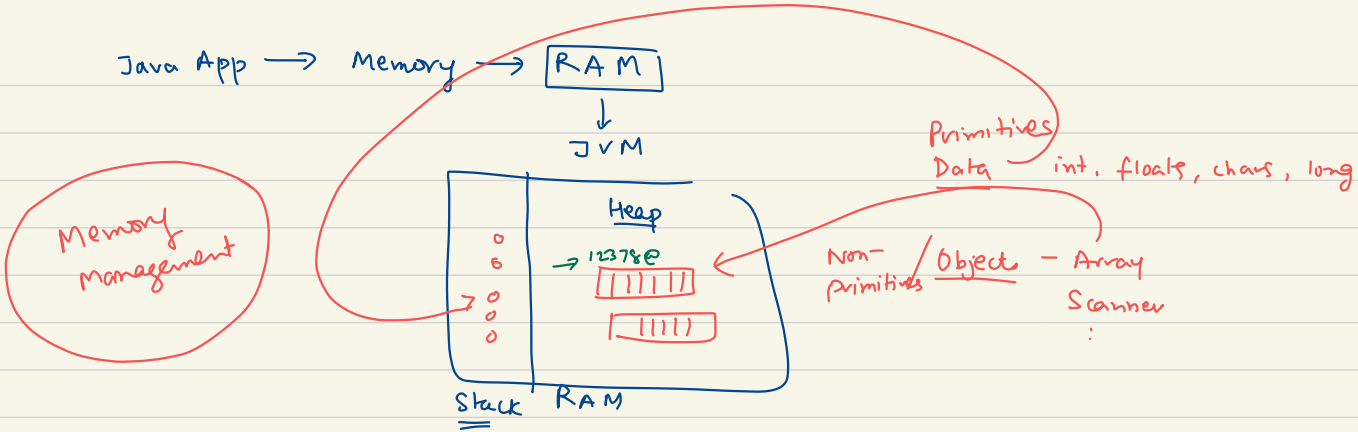
Index → 0 to n-1

marks

$n = 100$

`print(marks[3]);` → 34

```
marks[0] = 52;
marks[1] = sc.nextInt();
:
marks[i].
```



Basic op

1. Create (2 ways)
2. Read
3. Print
4. Edit / Update

Object

Dog → height
→ color
→ weight
} Knows

→ bark()
→ run()
→ eat()
} actions

dog.height

Array → length
} Knows

} behaviour

marks.length