
Rec
Hard
II

→ Number of Digit One

i/p → n

i/p -> n = 13

0, 1, 2, 3, 4, 5, 6

7, 8, 9, 10, 11, 12, 13

o/p -> 6

i/p -> n = 0

0

o/p -> 0

n = 20

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

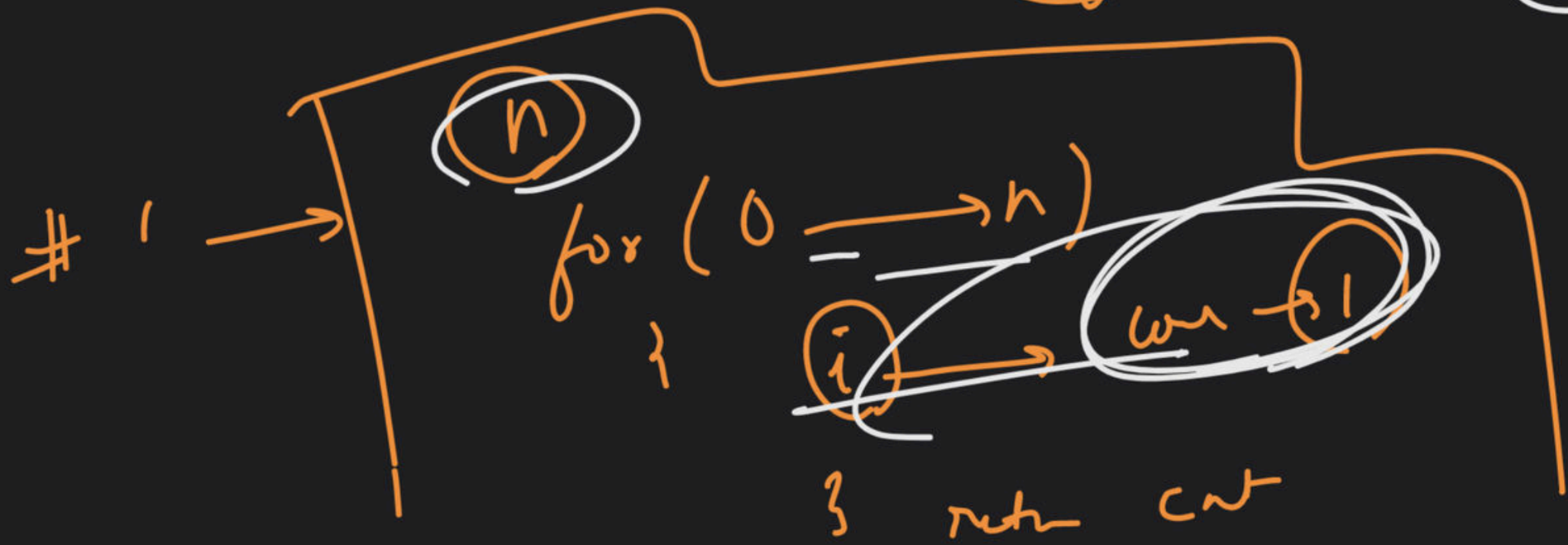
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

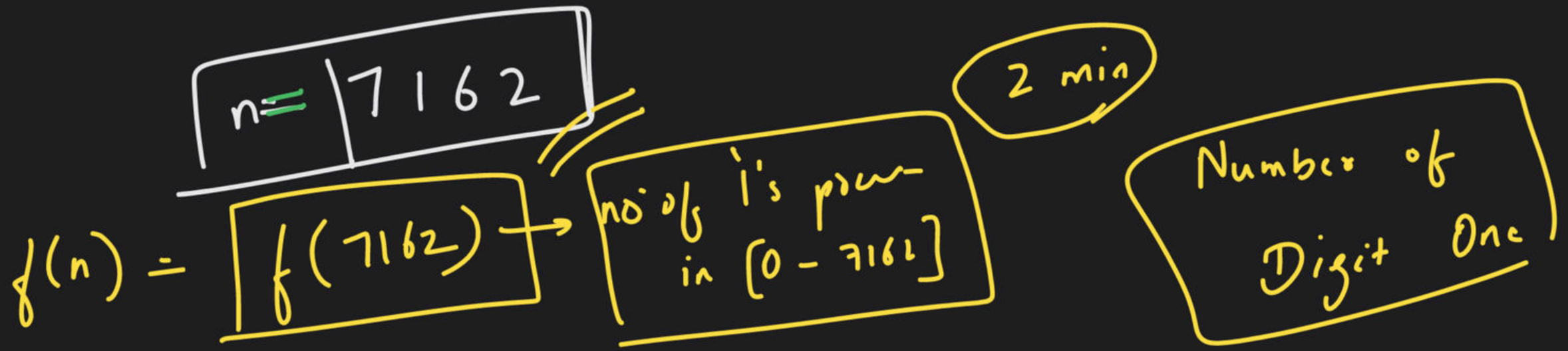
n = -12

Logic

n = 10

1 Cr




n = 7162
f(n) = f(7162)
no of 1's present in [0 - 7162]
2 min
Number of Digit One

n == 0

↳ return 0

7192

while (n > 0)

{ 7192 / 10 → 2

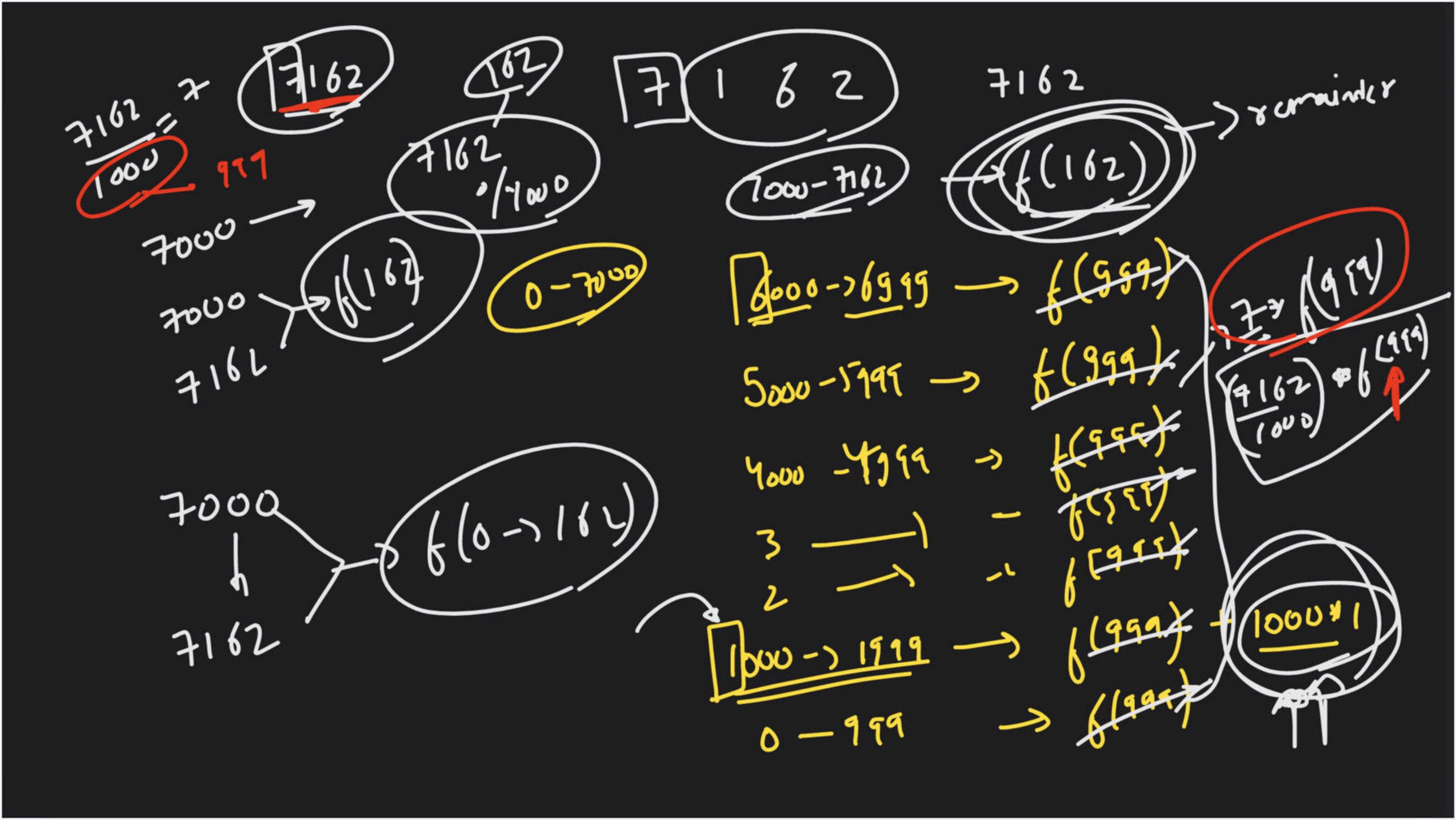
7192 / 10 → 719

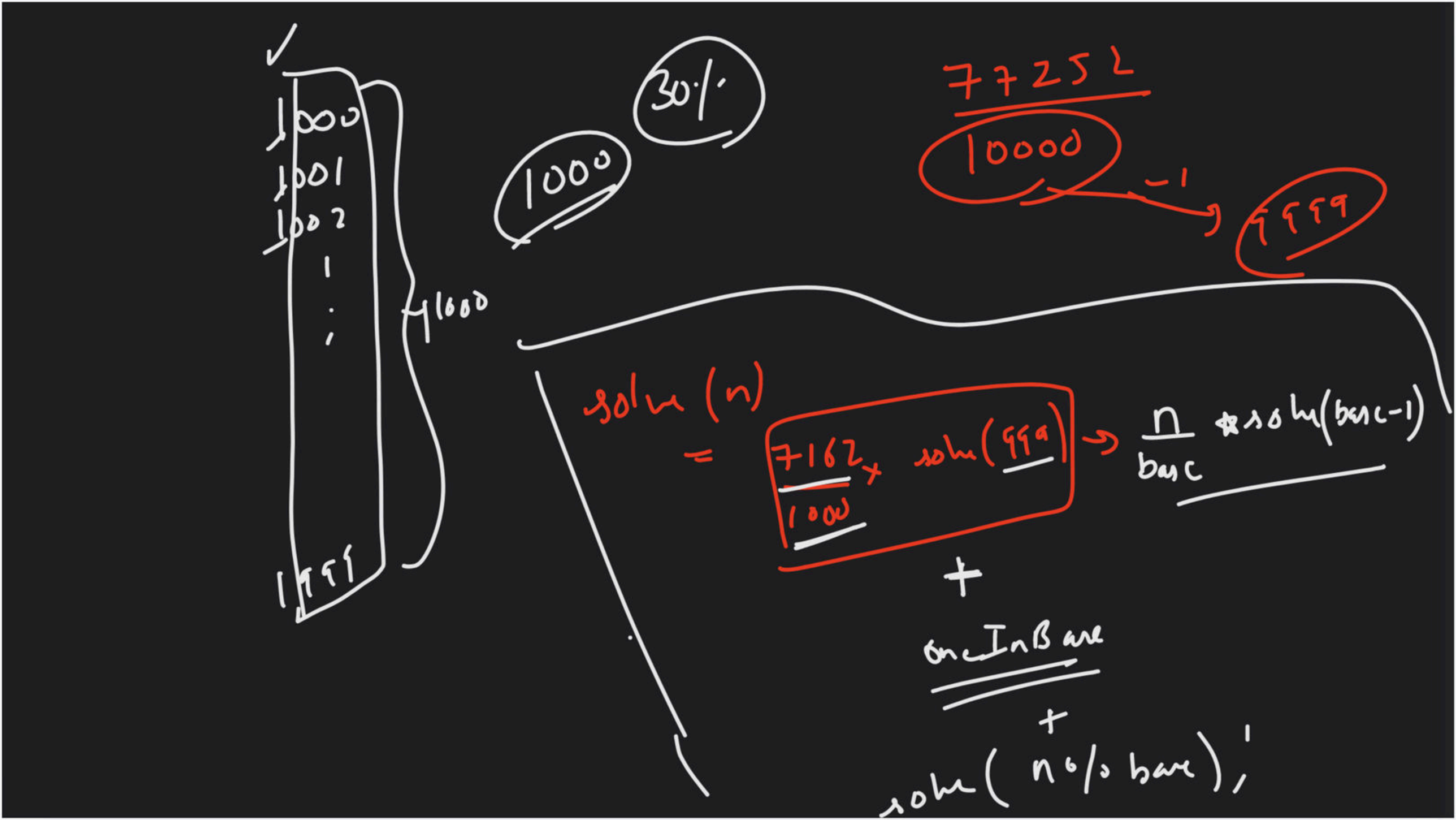
.

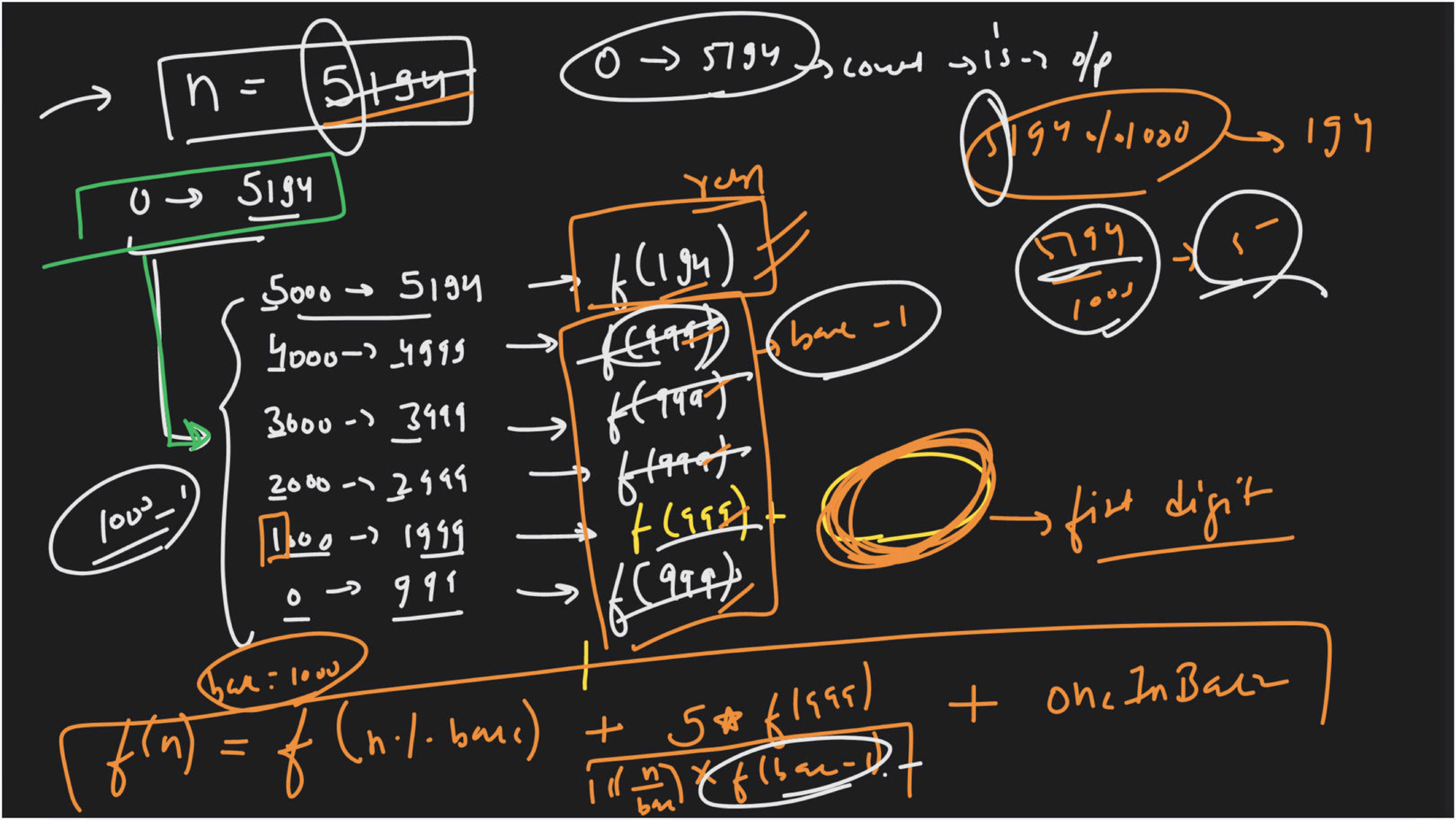
.

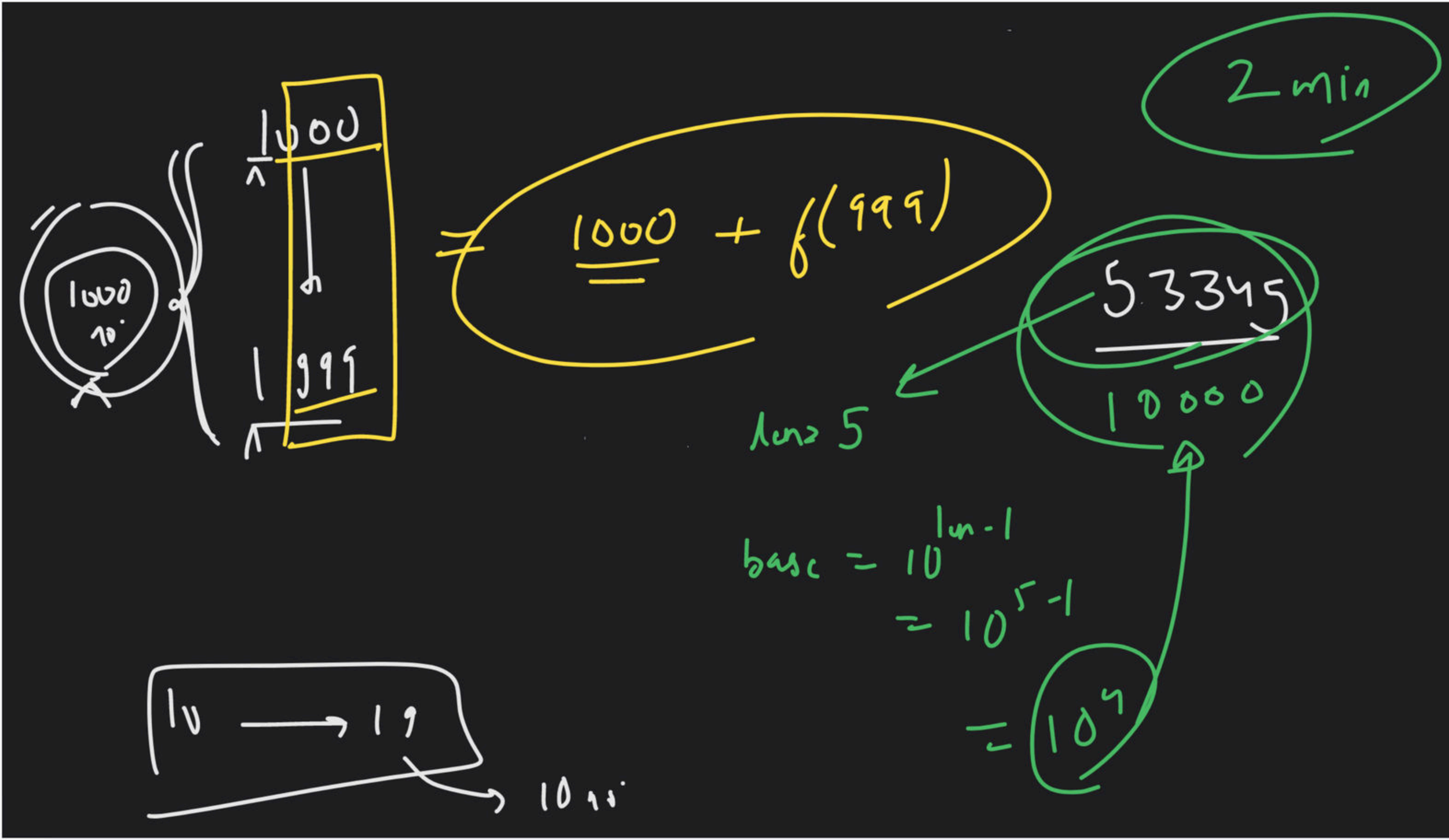
n - -

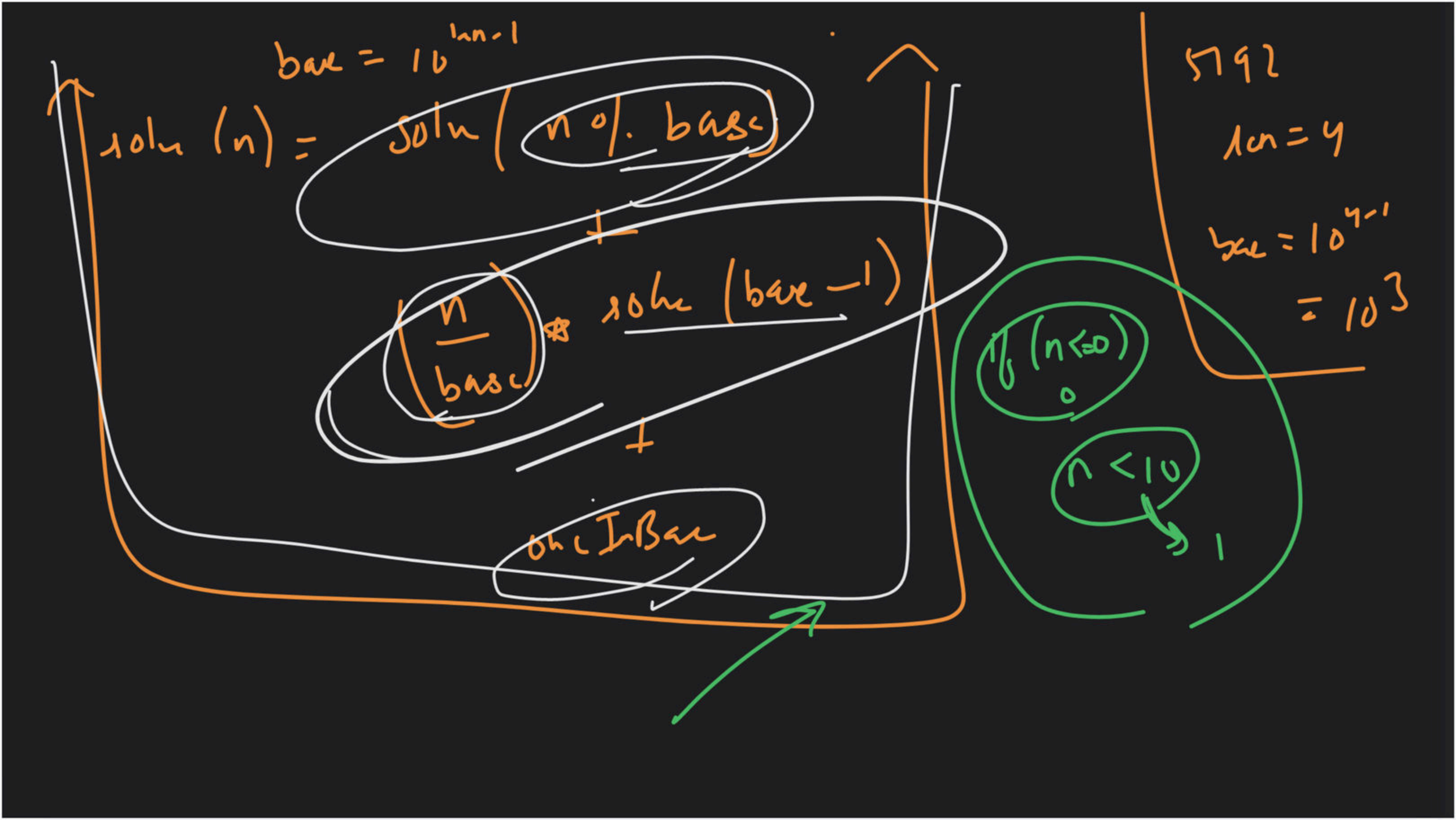
}


7162 / 1000 = 7
7162
162
7162 % 1000
7000
7000
7162
f(162)
0 - 7000
7000
7162
f(0 -> 162)
7 1 6 2
7000 - 7162
7162
f(162)
remainder
6000 -> 6999 -> f(999)
5000 - 5999 -> f(999)
4000 - 4999 -> f(999)
3 -> f(999)
2 -> f(999)
1000 -> 1999 -> f(999)
0 - 999 -> f(999)
7 * f(999)
(7162 / 1000) * f(999)
1000 * 1


1000
1001
1002
1
:
1999
1000
30%
1000
77252
10000
-1
9999
solve(n)
=
7162
1000
x solve(9999)
n
basic
*solve(basic-1)
+
oneInBase
+
solve(n%base);


n = 5194
0 → 5194 → count 1's → o/p
5194 % 1000 → 194
5194 / 1000 → 5
0 → 5194
5000 → 5194 → f(194)
4000 → 4999 → f(999)
3000 → 3999 → f(999)
2000 → 2999 → f(999)
1000 → 1999 → f(999) +
0 → 999 → f(999)
base − 1
1000 − 1
first digit
base = 1000
f(n) = f(n % base) + 5 * f(999) + onesInBase
f(base − 1)


2 min
1000
1000 to.
1 999
1000 + f(999)
53345
10000
len 5
base = 10^(len-1)
= 10^5-1
= 10^4
10 to.


base = 10^(len-1)
toInt (n) = solve (n % base)
+
(n / base) * toInt (base - 1)
+
ohne InBase
5792
len = 4
base = 10^(4-1)
= 103
if (n<=0)
0
n < 10
1

n = 13 ⟶ ans → 6

n = 13

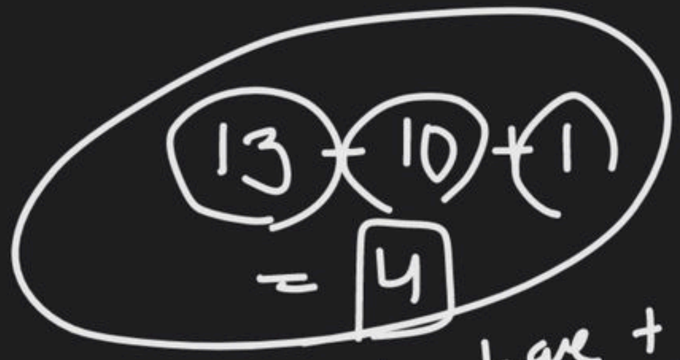
base = $10^{2-1} = 10^1 = 10$

first digit = 13/10 = 1

rem = 13 % 10 = 3

```
int OneInBase = 0
```

oneInbase = 4



n - base + 1

13 - 10 + 1

10 → 13

13 - 10

= 3 + 1 → 4

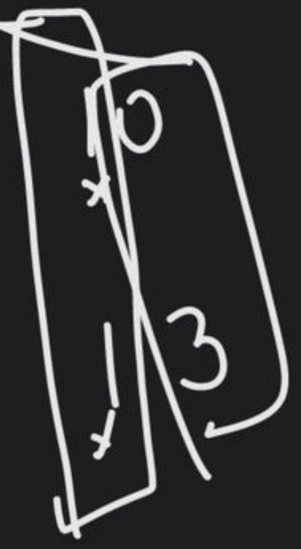
ans = solve(2) + 4 + 1 * f(9)

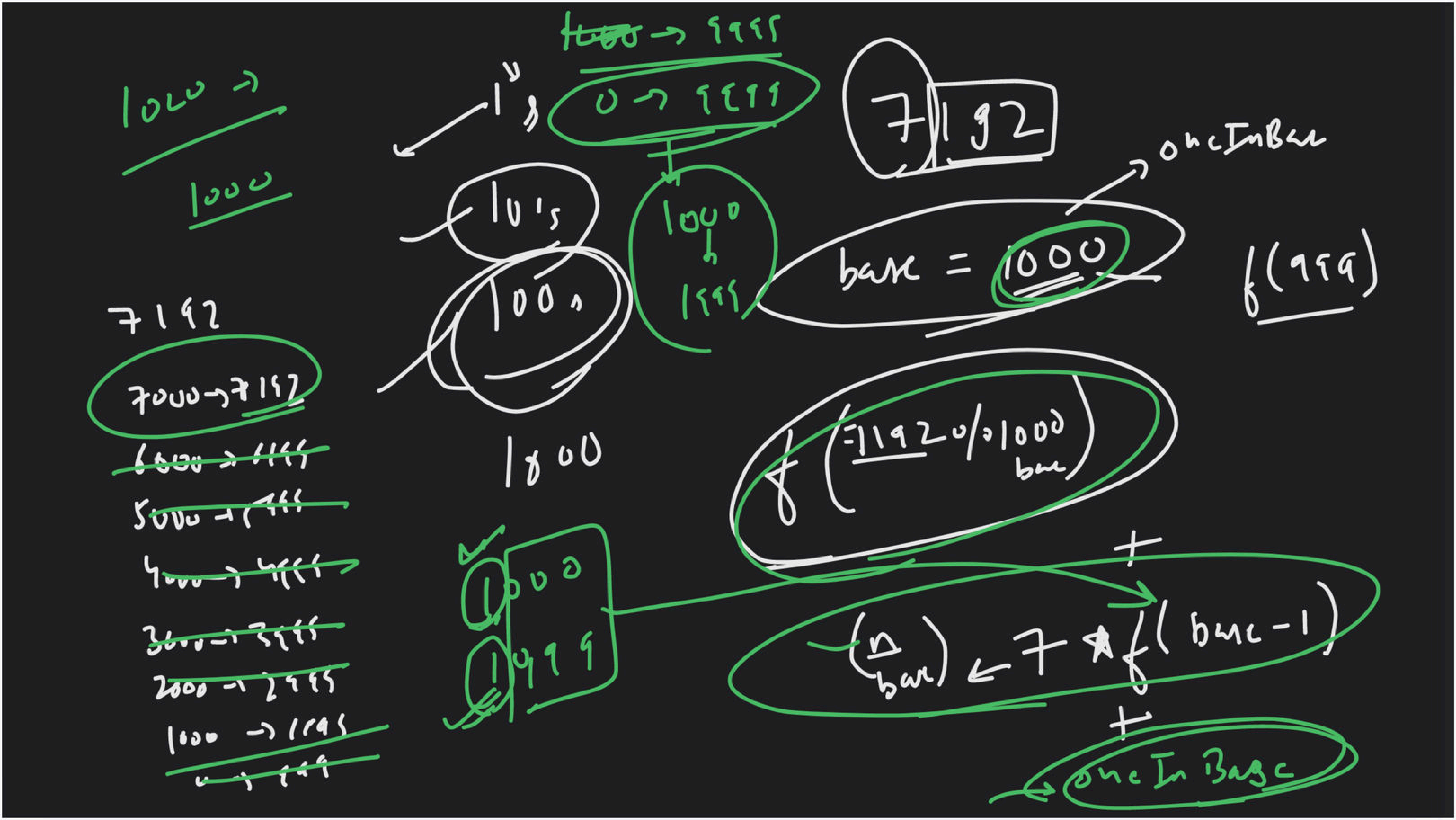
sum

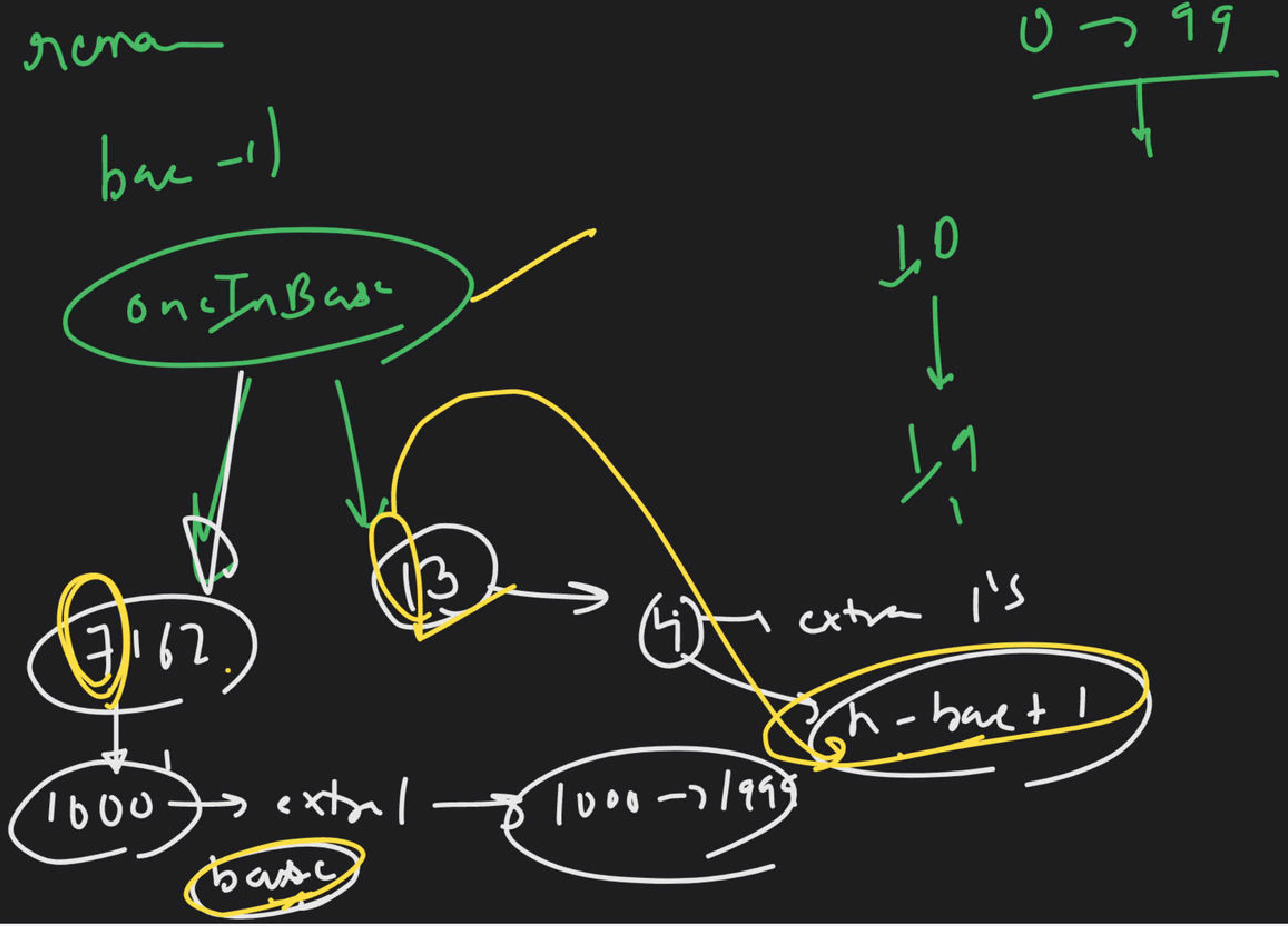
one Index

= 1 + 4 + (+1)

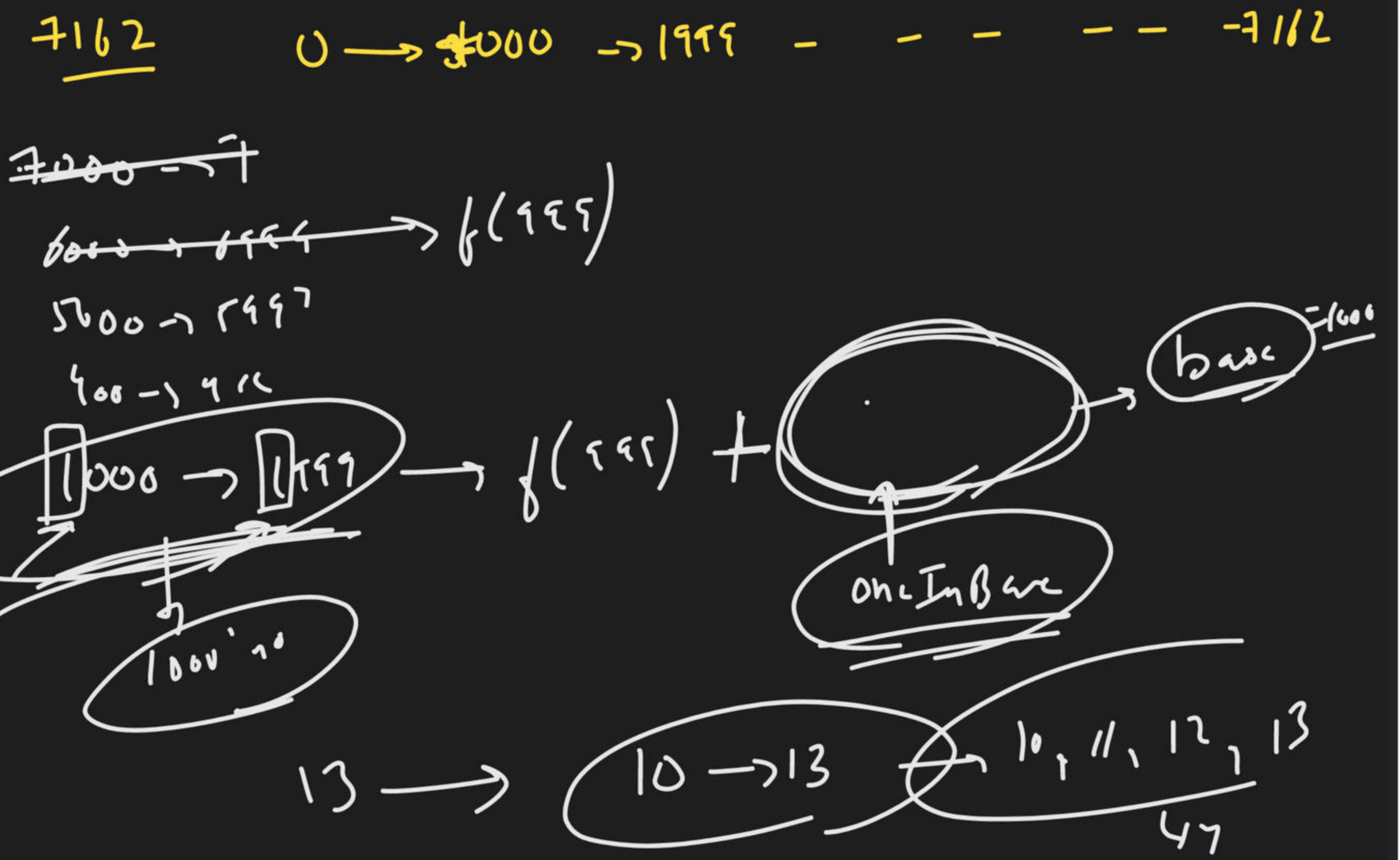
= 1 + 4 + 1 = 1

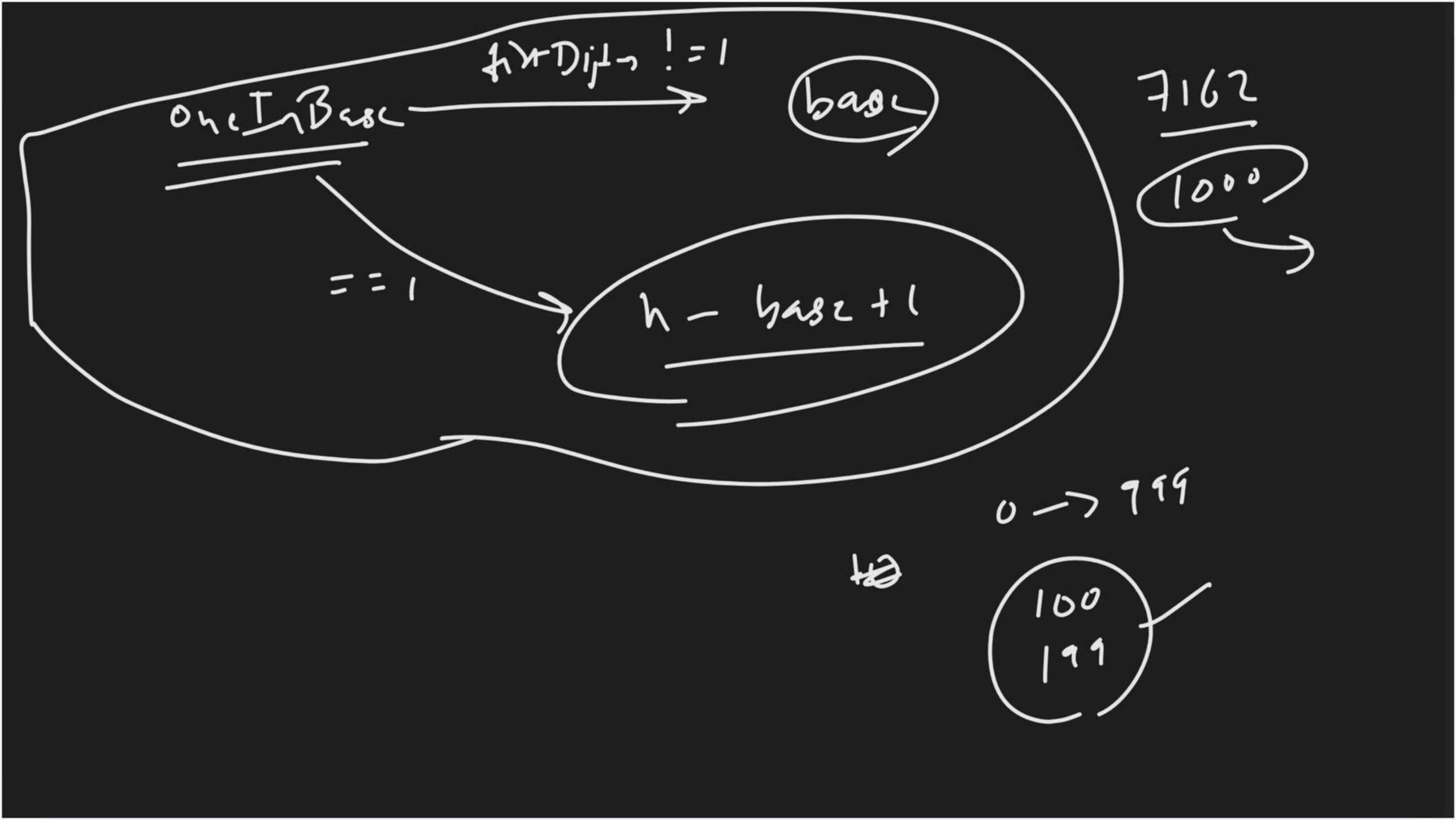



0 → 9999
1000 → 1999
1000
1800
7192
base = 1000
onesInBase
f(999)
7192
7000 → 7192
6000 → 6999
5000 → 5999
4000 → 4999
3000 → 3999
2000 → 2999
1000 → 1999
0 → 999
f(7192 % 1000)
+
7 * f(base - 1)
+
onesInBase


0 → 99
10
19
onesInBase
7162
13
4 → extra 1's
n - base + 1
1000 → extra 1 → 1000 → 1999
base

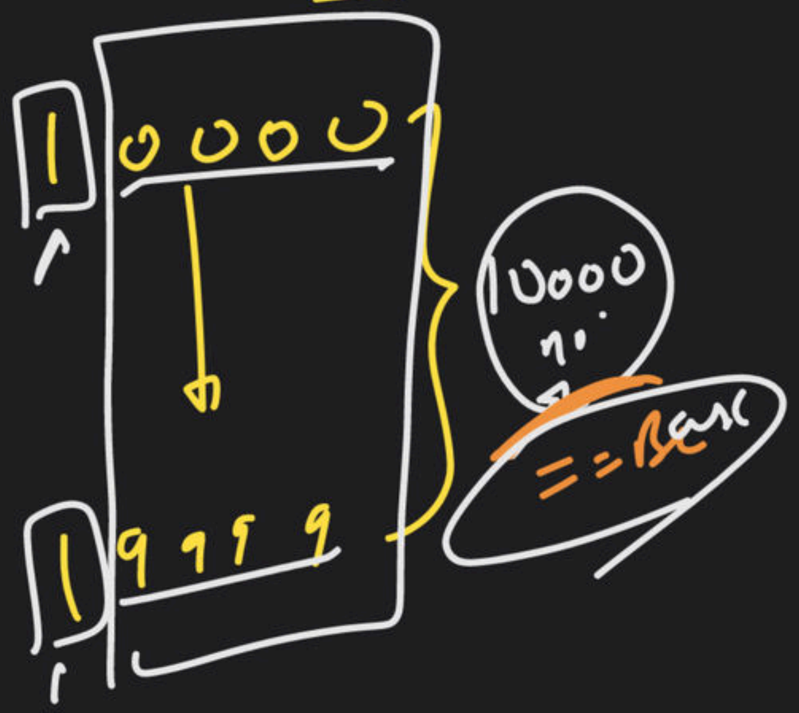
7162




oneInBase
for Digit != 1
base
7162
1000
== 1
n - base + 1
0 -> 999
100
199

n = 74899 → len = 5

basc → 10000 → 10^{len-1} → 10^{1} = 10000



74899 →

70000 → 74899 → f(74899)

60000 → 69999 → f(99999)

50000 → 59999 → f(99999)

40000 → 49999 → f(99999)

30000 → 39999 → f(39999)

20000 → 29999 → f(99999)

10000 → 19999 → 10000 + f(99999)

0 → 9999 → f(99999)

run

onclick

$f(74899) = f(4899) + 7 * f(9999) + 10000$

```
if (findDigit == -1)
    oneInBase = n - base + 1;
else
    oneInBase = base;
```

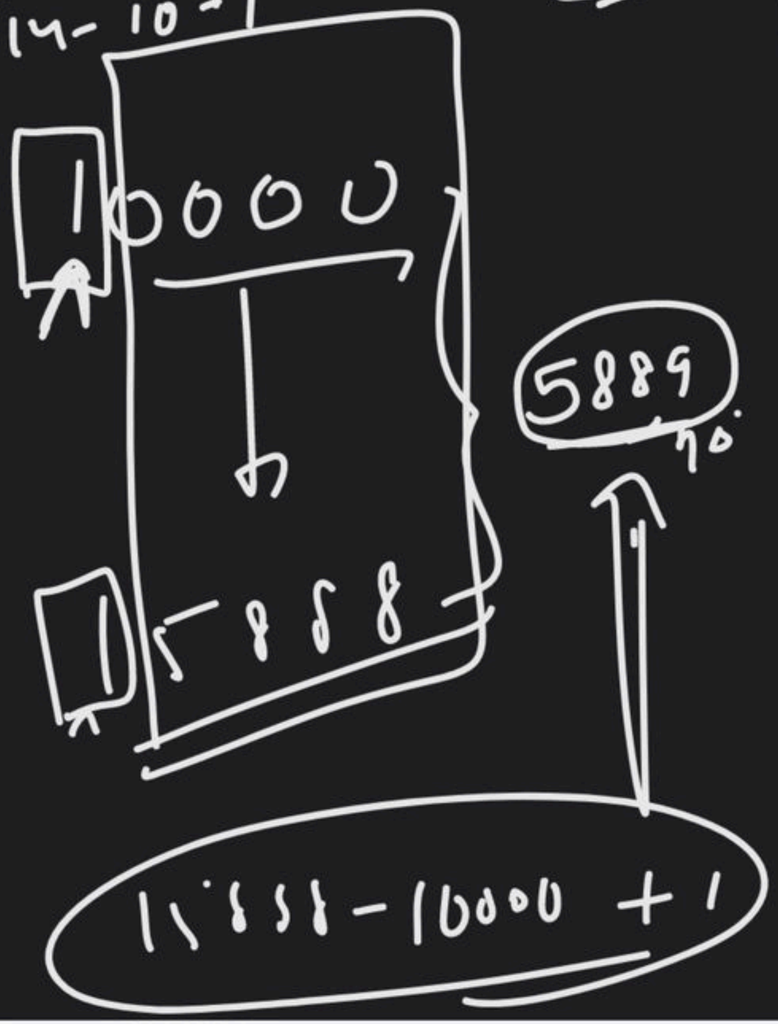
→ Most Important

$n = 15888$

base 2 $10^{5-1} = 10^4 = 10000$

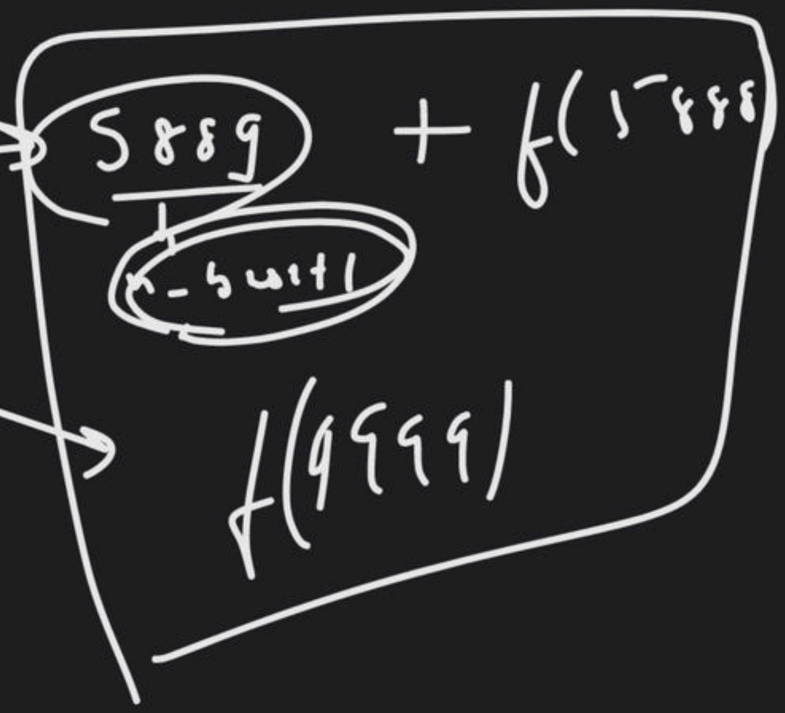
10 → 14

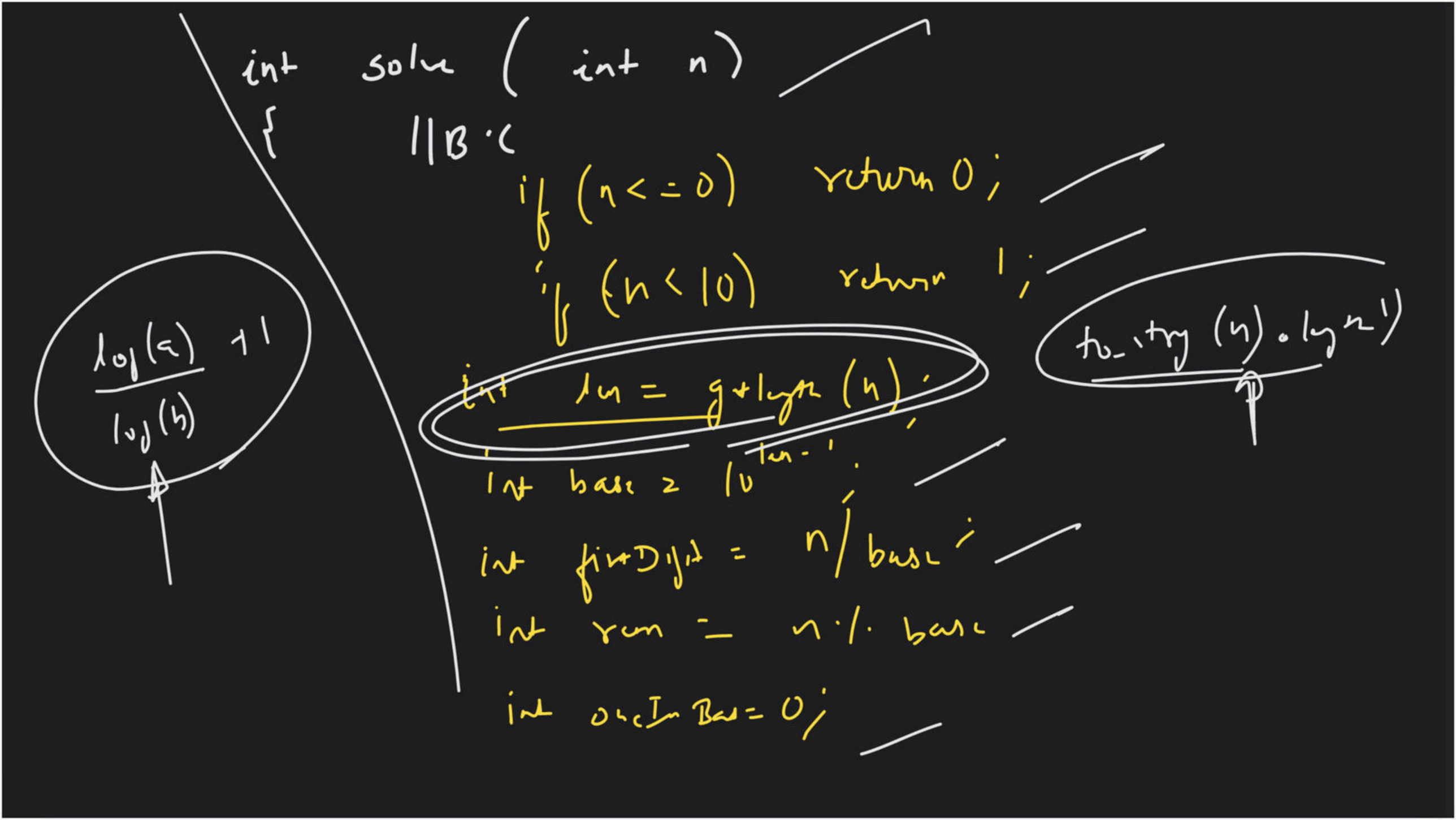
14 - 10 + 1



10000 → 15888

0 → 9999




int solve (int n)
{
//B.C
if (n<=0) return 0;
if (n<10) return 1;
int len = getLength (n);
int base = 10^(len-1);
int firstDigit = n/base;
int rem = n%base
int oneLessBase = 0;
log(a) / log(b) + 1
to_string (n).length()

if (firstDigit == 1)

oneInBase = n - base + 1;

else

oneInbase = base;

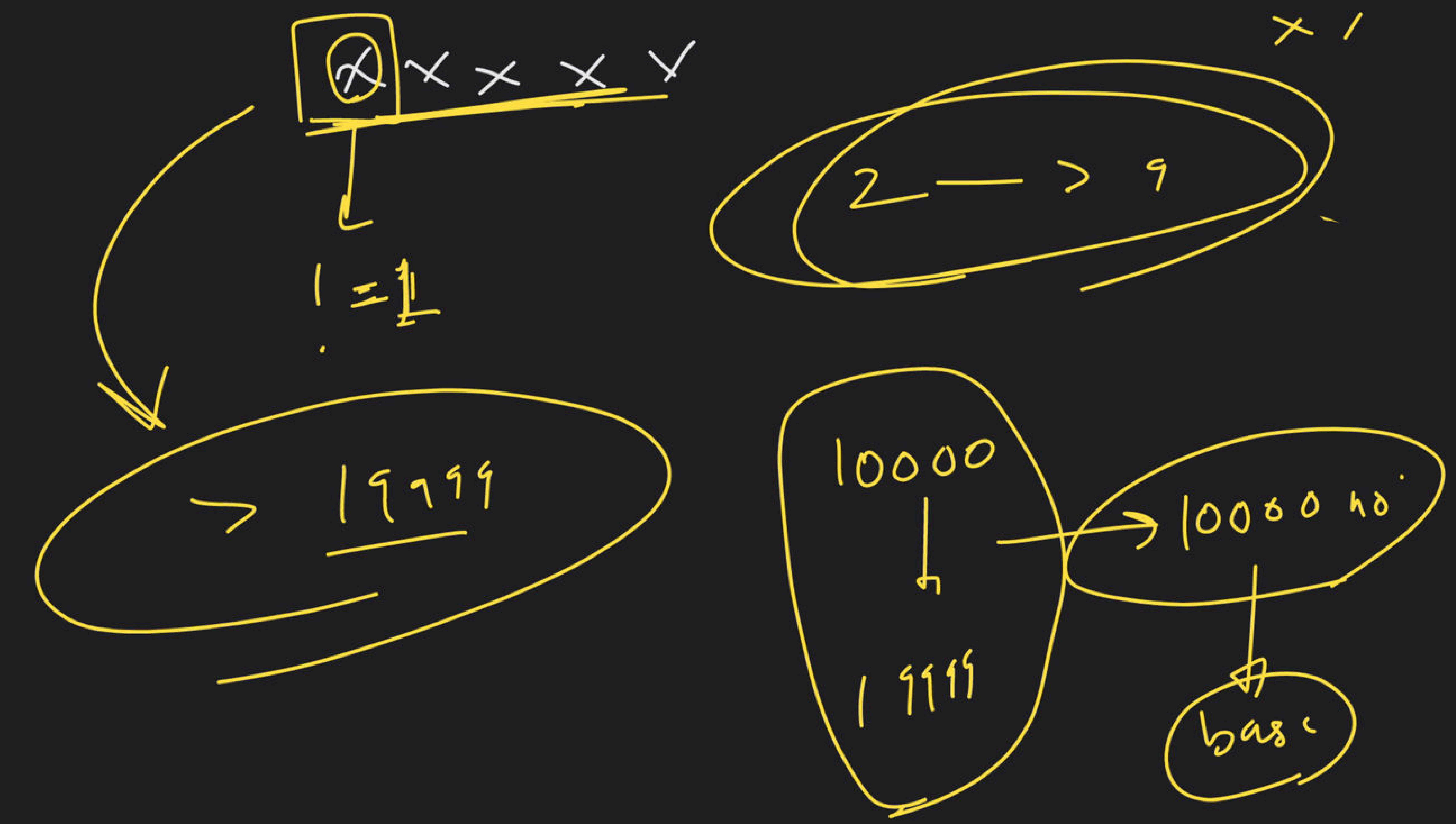
return solve(rem) + oneInBase + ~~firstDigit~~ * solve(base-1);

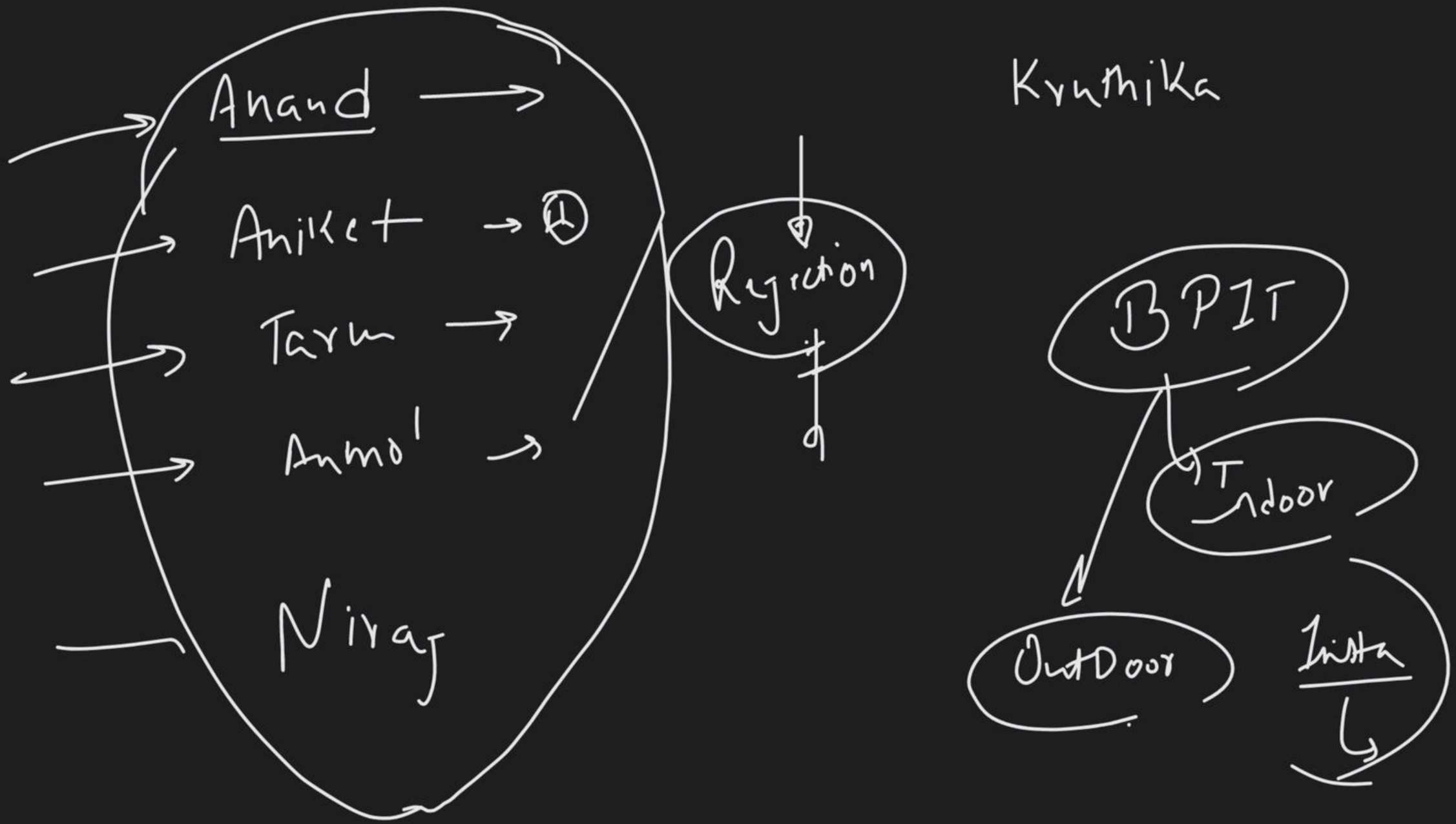
(1-1 min)

45 min

10 min

stuck

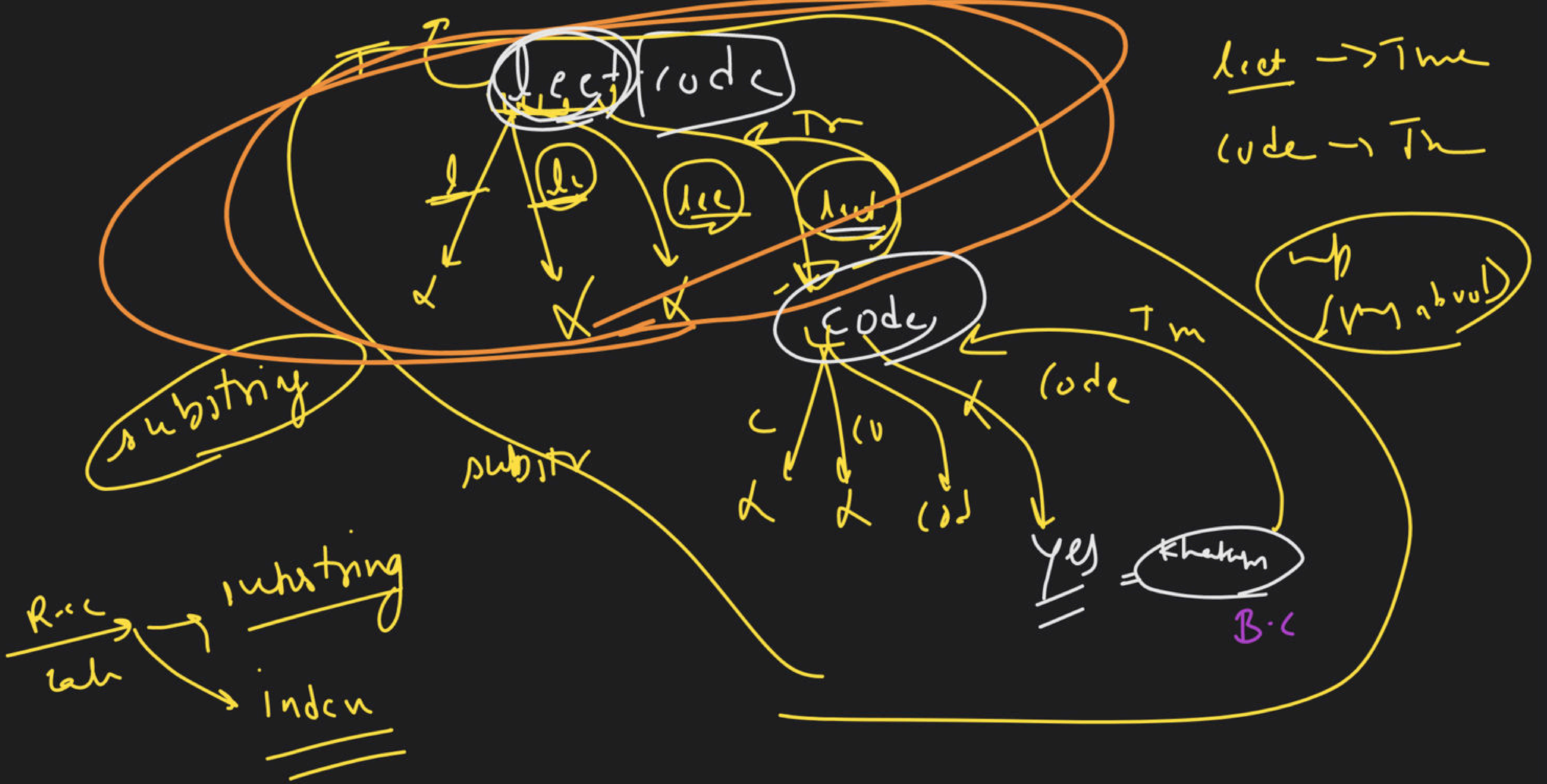

× 1
2 —> 9
> 19999
10000
19999
10000 no.
base

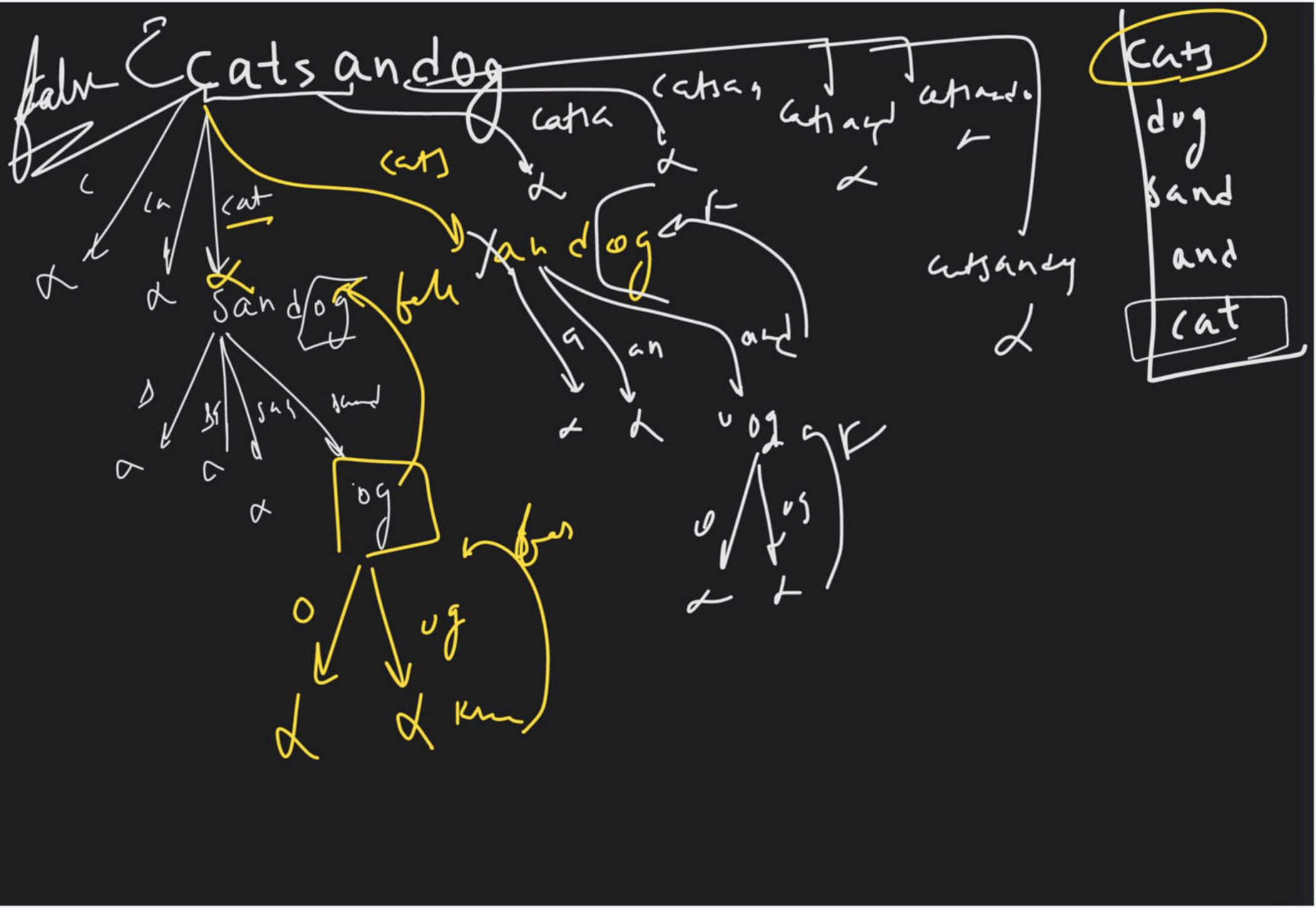

Anand
Aniket
Tarun
Anmol
Niraj
Rejection
Kruthika
BPIT
Indoor
OutDoor
Insta

Word Break

s → leetcode

d → [leet , code]


leet code
code
Yes
substring
index
B.C


cats
dog
and
cat

leet code

H/W

exponential → n^n or 2^n → ? → T.C

```
bool solu (start, str, map)
{
    int n =
    BC  if (start == n)
            return true;

    for (start < n)
    {
        //substring
        if (substring present in map && Rec call for remaining string)
            return true;
    }
    return F;
```

T.C → $O(n^2)$?

Integer to English Words

2 min

Break

n -> 1233

n = 12345

hundred
thousand
million
Billion

ten thousand → "Ten Thousand"

if (n == 0)
return "Zero";

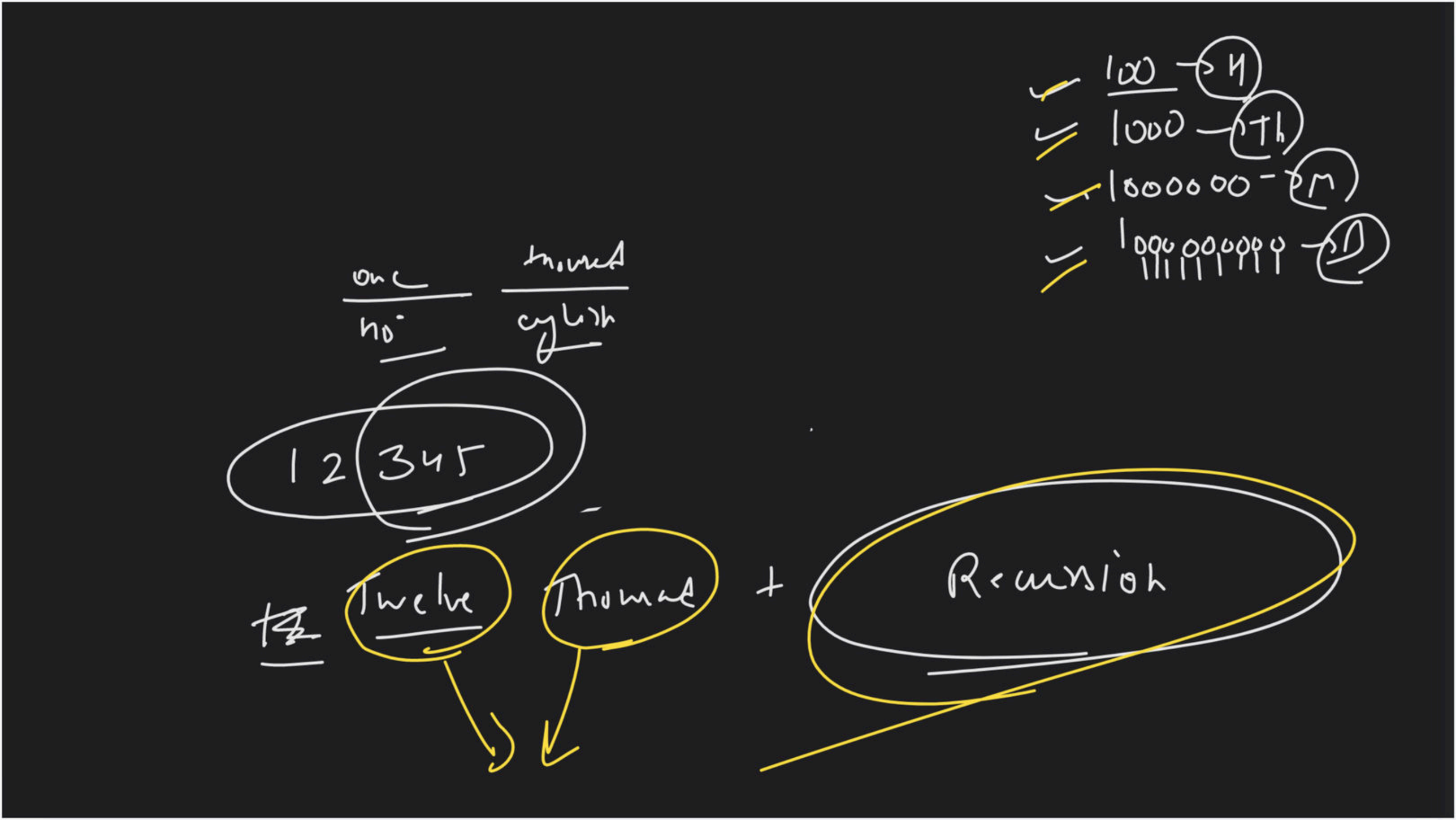
arr1 = { "", "Ten", "Twen", "Th", ..., "Ninety" }

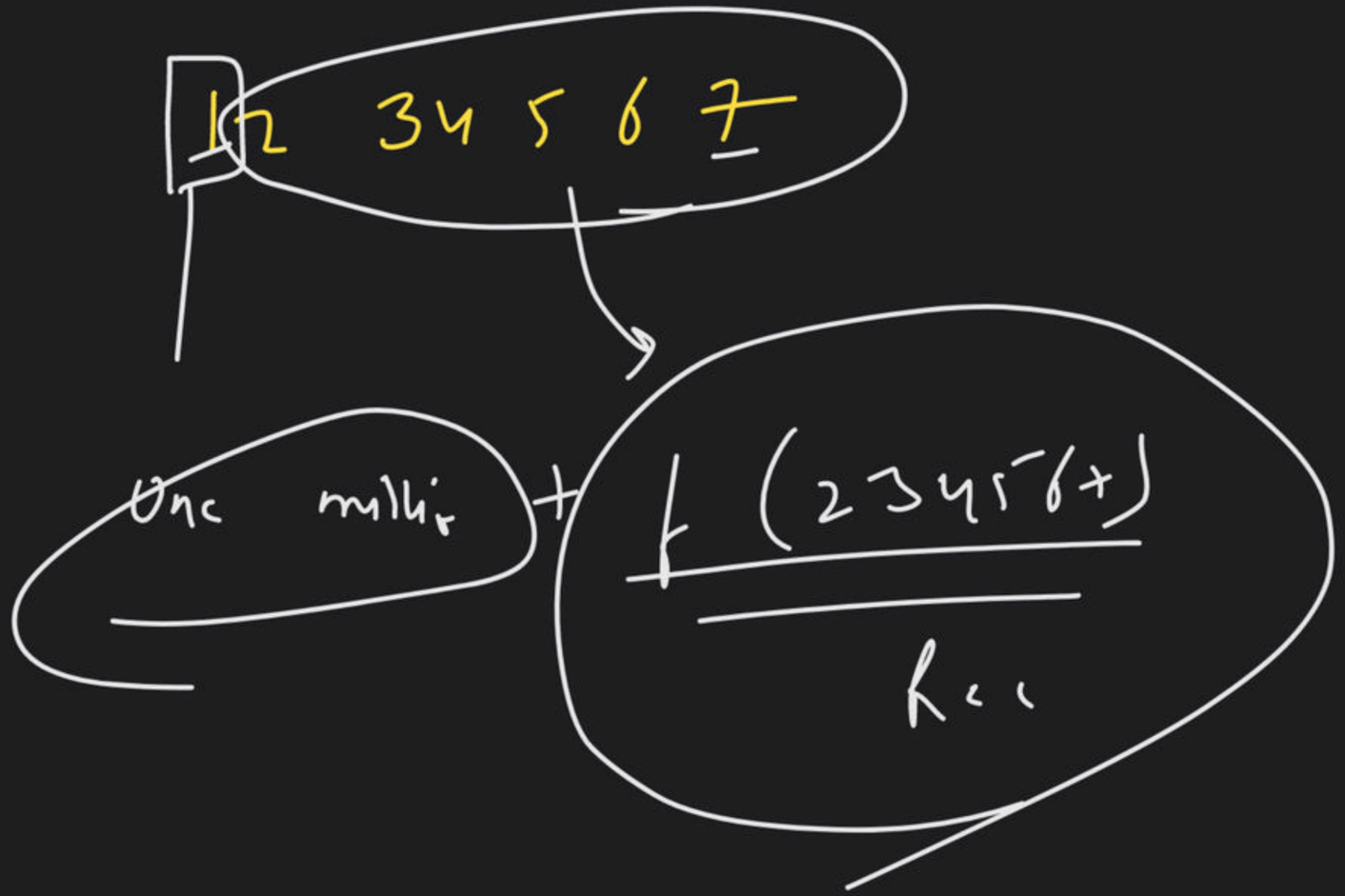
arr2 = { "", "One", ..., "Nineteen" }

$n <= 2^{31} - 1$

$2^{10} \rightarrow 1024$

10^9 → Billion


100 - H
1000 - Th
1000000 - M
1000000000 - B
one
two
no.
cylon
12345
Twelve
Thousand
+
Recursion



1 → One

2 → Two

1000000000 -> Billion

1 → One

2 → Two

3 → Three

lg → Nineteen

20 -

30 -

40 -

50 -

60 -

70

80

90 -

100 -

1000 -

1000000

10000000

array

Vc →

vector

int, string

map

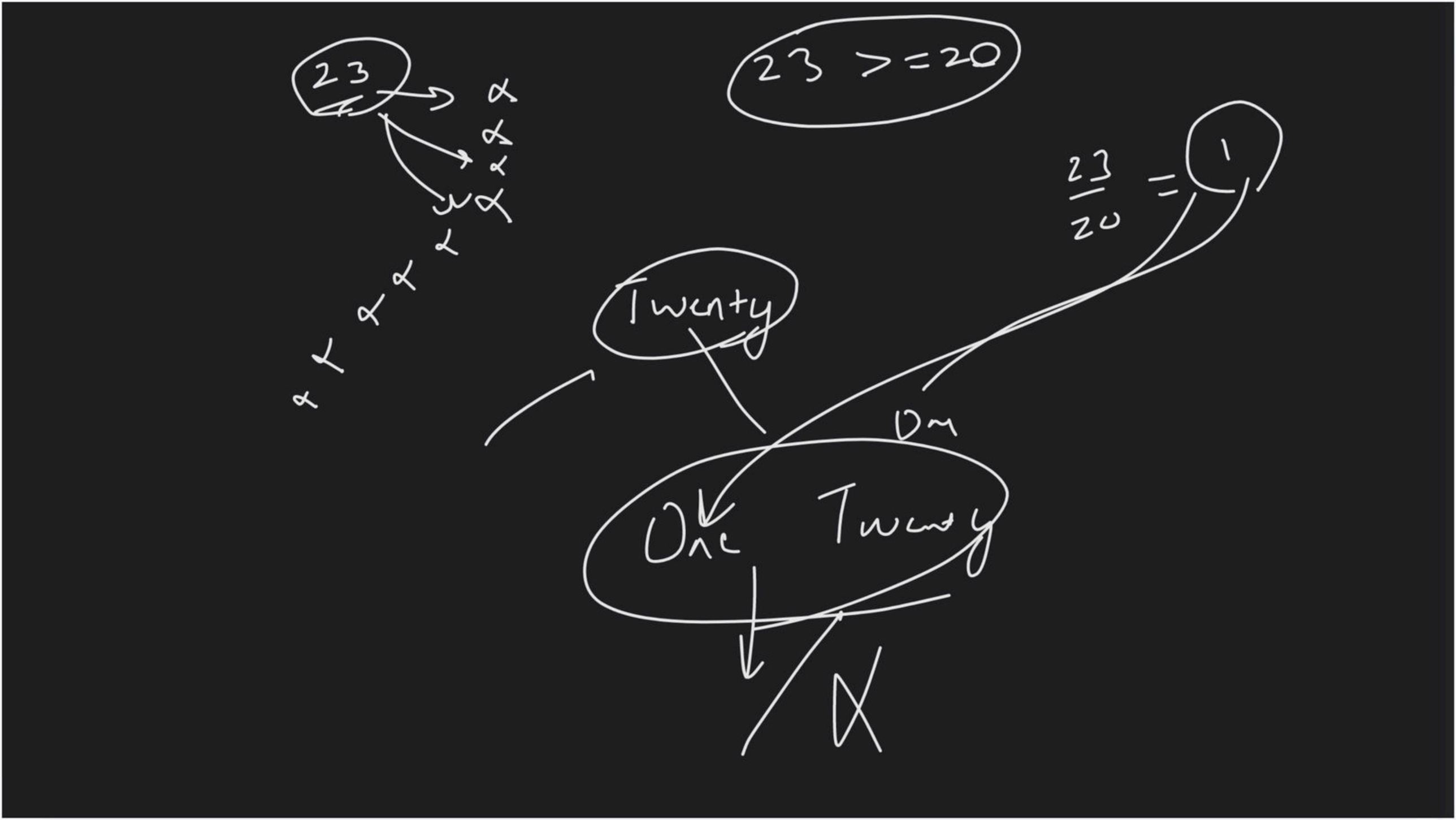
One Hundred

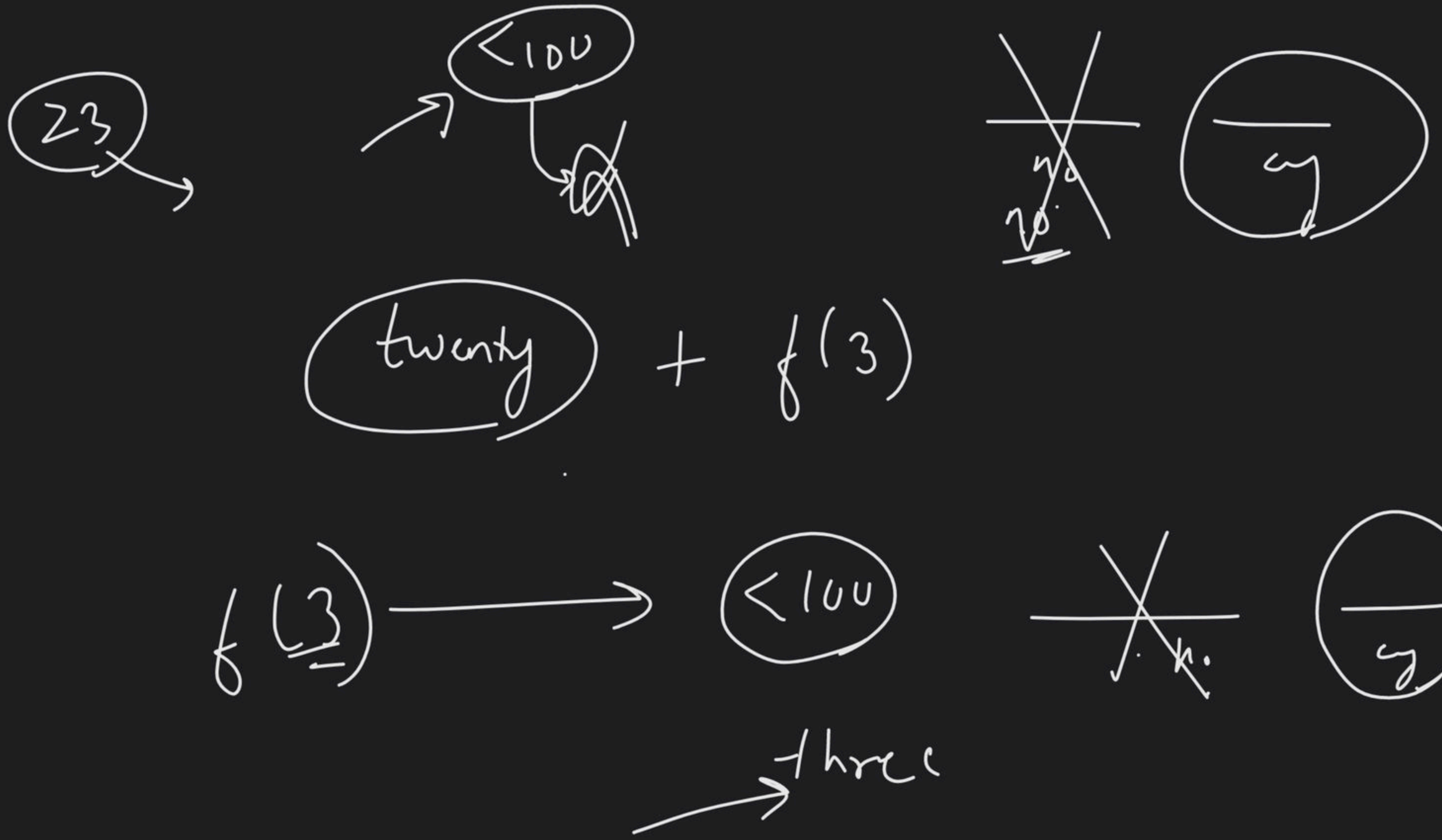
↑ ↑

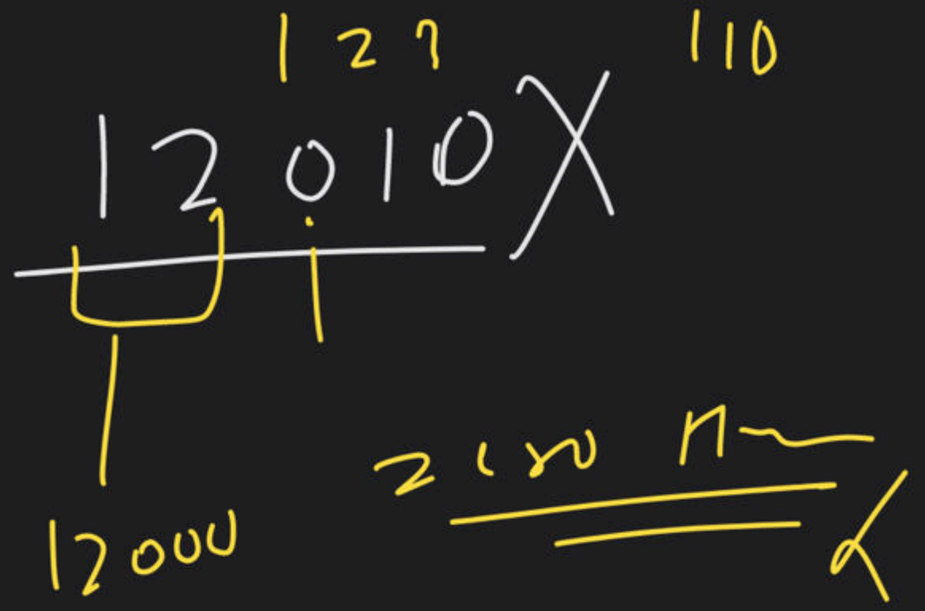
$\frac{123}{100} = 1$ → One

+ f(23)

One Hundred + f(23)


23
23 >=20
23/20 = 1
Twenty
Dm
One Twenty


23
<100
twenty
+ f(3)
f(23)
<100
three


127
110
12010
12000
2100

n = 12345

→ BC

→ MC

→ Th ...

$\frac{12345}{1000} = 12$

Twelve thousand + f(345)

n = 345 → Pc

m

n

M

Three hundred + f(45)

f(45)

45 < 100

45 >= 40

forty + f(5)

f(r) → <100

fifty

① 1234567 → Dx
→ M

1234567 / 1000000 → 1

One million + f(234567)

234567 → Dx
→ Mx
→ Th

234567 → 234
1000
Thousand + f(567)

soln (234)

solu (234) → 100

$\frac{234}{100} = 2$

Two Hundred + f(34)

34 < 100 →

Thirty 34 >= 30 + f(4)

f(4) 4 < 100 →

four

One million Two hundred Thirty four thousand + f(3)

567 → 100 → five hundred + f(67) 567/100 =

67 < 100 → ~~X~~ sixty + f(3)

7 < 100 → ~~X~~ seven

12345

one

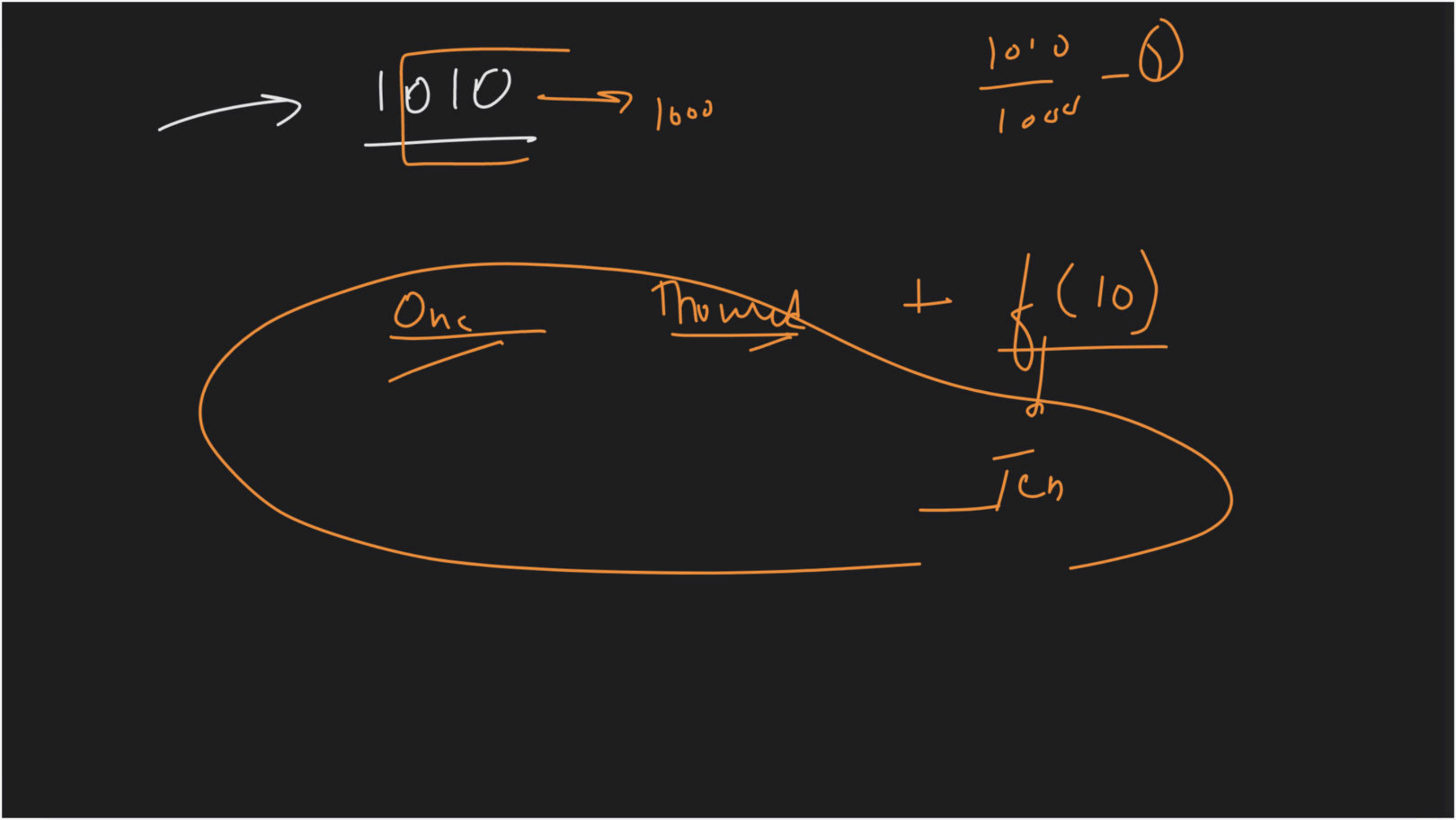
hundred

<100

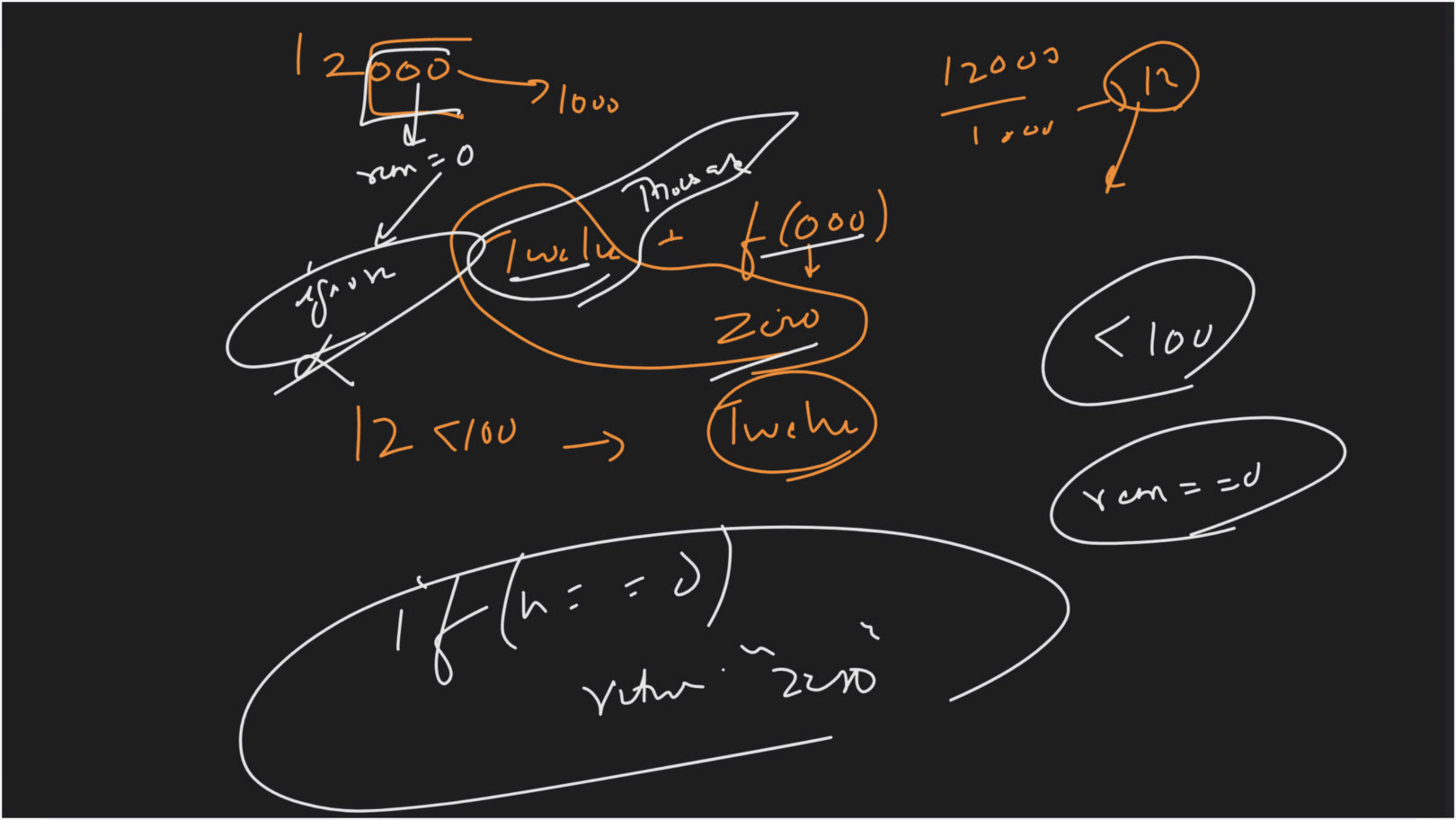
62

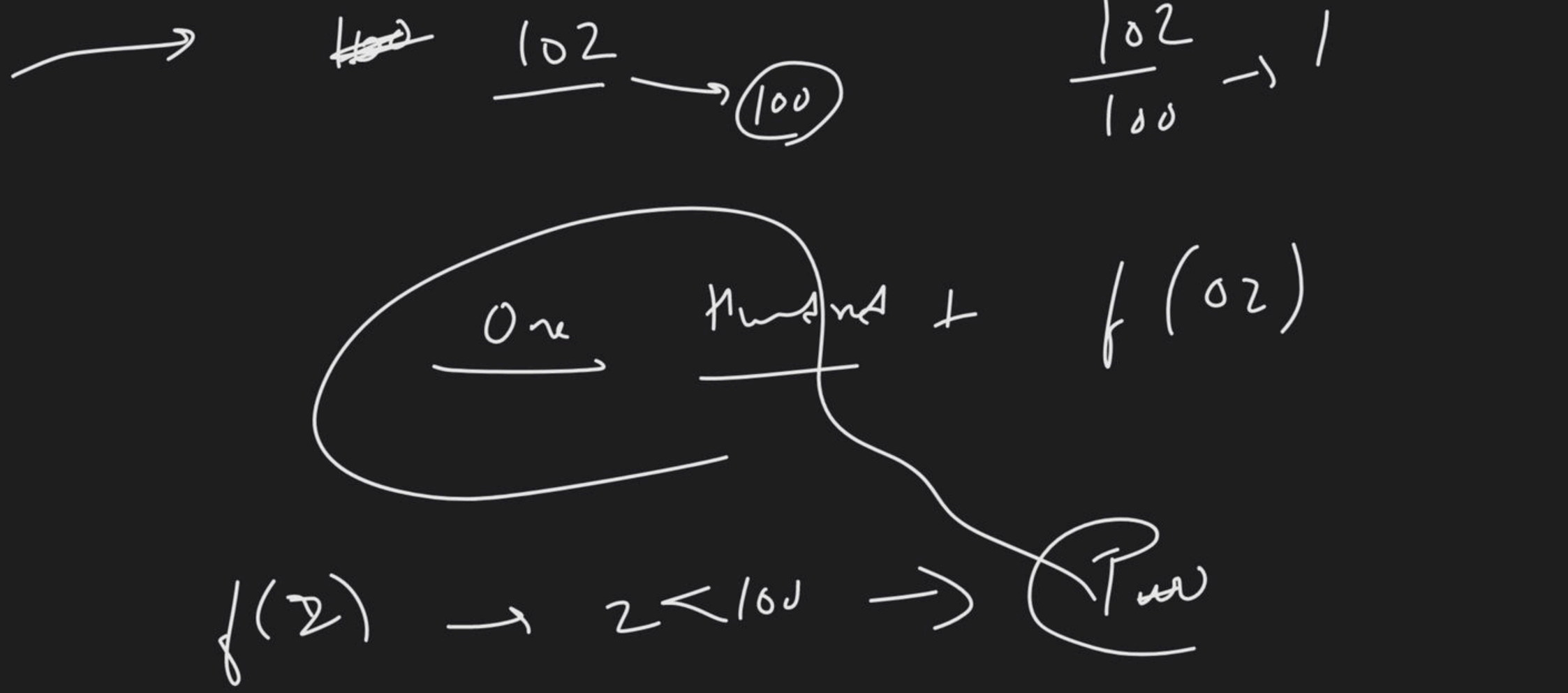
sixty

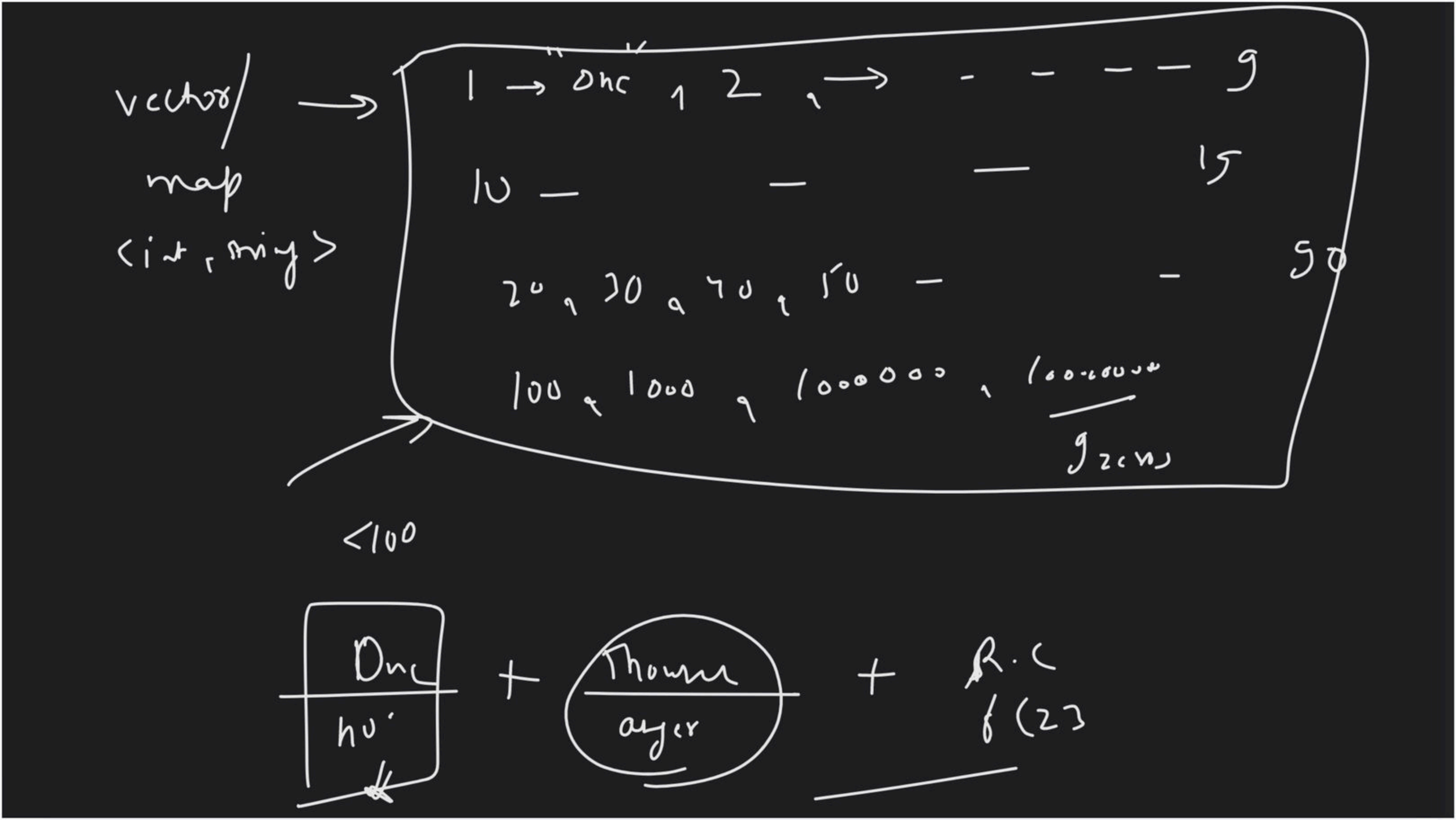
fifty (7)


1010
1000
1010 - 8
1000
One
Thousand
+
f(10)
f(n)

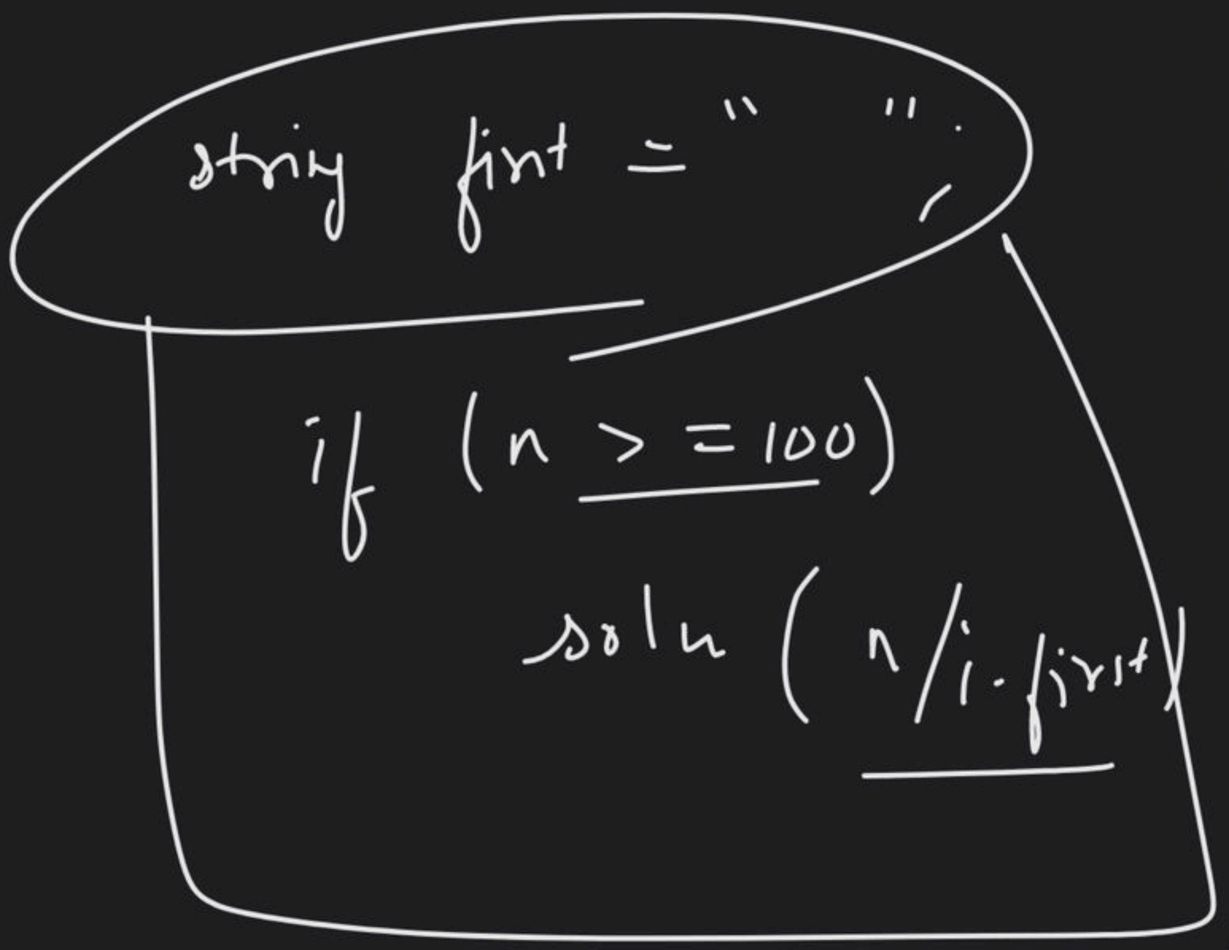

12000

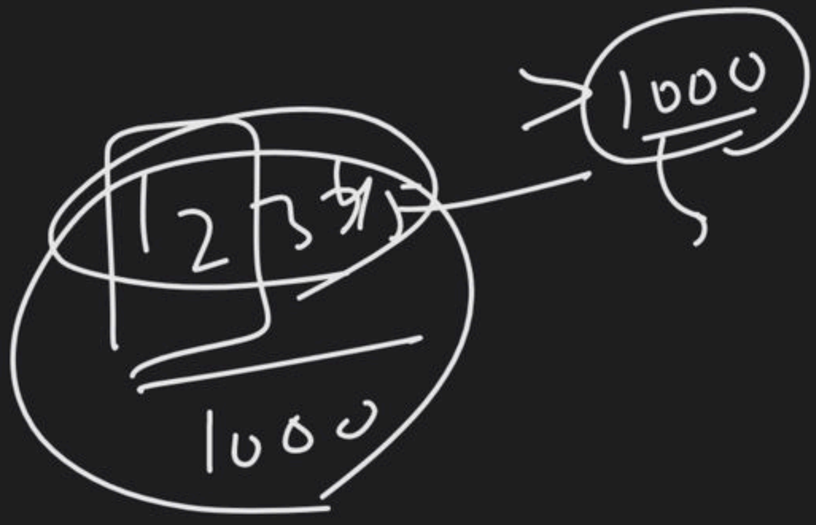

12000
1000
rem = 0
Thousands
Twelve
+
f(000)
Zero
12 < 100
Twelve
12000
1000
12
< 100
rem == 0
if (n == 0)
return "Zero"


102
100
102
100
1
One
Hundred
f(02)
f(2)
2<100
Pass


vector/
map
<int, string>
1 → one
10 —
20, 30, 40, 50 —
100, 1000, 1000000, 10000000
9
15
50
<100
One
two
Three
ayer
R.C
f(23

- (~~n < 100~~ ?





$\frac{12341}{1000} = 12$

+ string second = i.second;

Tuesday Thursday

2 hr → 2.5 hr