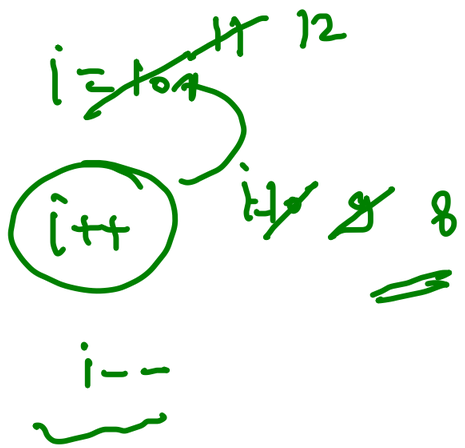
print

input

variables

loop → counting.



Prime Numbers →

t = 5.

① →	5] →	prime
② →	11] →	prime
③ →	9] →	Not prime
④ →	49] →	not prime
⑤ →	81] →	prime.

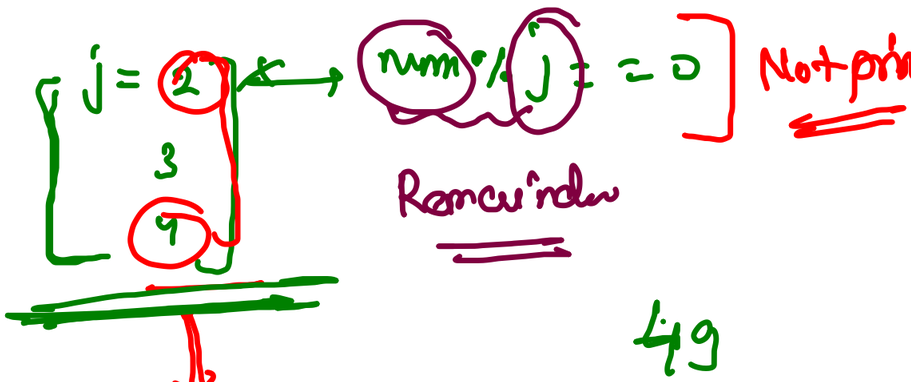
after calculation

At a moment →

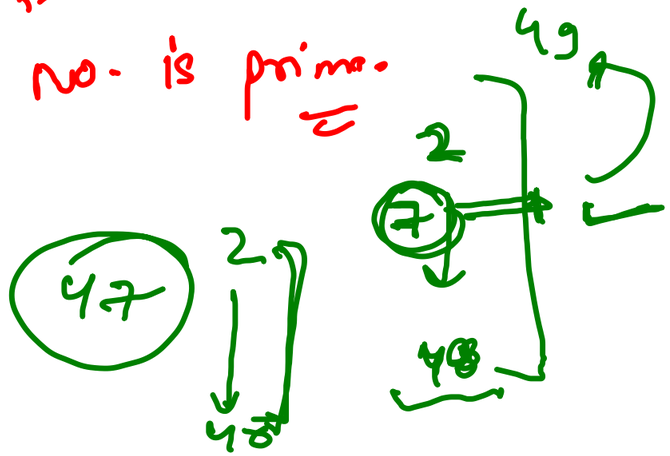
Completely divide. → Not prime

No divide. → prime ✓

num = 9



No. is prime.



num = 49

num % div == 0 → not prime.

div = 2 x

3 x

4 x

5 → x

6 ~x

7 → break;

:

:

:

48

if → True.

flag → false

num = 49

47 % 2 ≠ 0 } ← prime.

2

3

4

5

6

:

:

46

48

num = 37 → smallest factor = 2

$\frac{37}{2} = 18.5$

18 × 2 = 36

19 × 2 = 38

2

2

3

4

5

6

7

8

9

i

10

11

12

13

14

15

16

37

17

18

19

20

21

22

23

smallest factor

24

25

26

27

28

29

30

31

32

33

34

35

36

$\frac{37}{21} = 2$

num / i = 2

18

num/2

18

1 to 18

$\frac{18}{10} = 1.8$

2

3

4 x

5 x

6

7 x

8 x

9

num/2

10

11 x

12 x

13 x

14 x

15 x

16 x

17 x

18 = 1

19

20

avoid

num / num = 1

= 2

20

11

23

21

Method-3 (Prime Number)

36 num = 36

1 x 36 — 36 x 1 — Ignore

2 x 18 — 18 x 2 — 36

3 x 12 — 12 x 3

4 x 9 — 9 x 4

6 x 6 → root

Stop

4x9

num % 2 == 0

num % 28 == 0

37

2
3 x 12
4 → 4x9
6
9 → 9x4
12 → 12x3
18

fix 9 ans

Necessary ??

Repeated

2
3
4
5
6
7
8
9
:
18

$\sqrt{37}$ — 6

After $\sqrt{37}$

n = 18;

```
( int j = 2 ; j <= √num ; j++) {
    if (num % j == 0) {
        isPrime = false;
        break;
    }
}
```

9

37

ranteed

that 3

2
3
4
.
.
.
18

method 2

Guaranteed

2
3
4
5
.
.
.
36

method

→ (j) <= √num → condition <=

Squaring both side.

Equality change

j * j <= num

j > 0

2 - n-1
method ① → 10,000 - Iteration

2 - n/2
method ② → 5,000 - Iteration

2 - √n.
method ③ → 100 - Iteration

10000

√10000 = 100

36

$n_1 \times n_2$

$n_2 \times n_1$ Inverse Fact

2 x 18

3 x 12

4 x 9

6 x 6

18 x 2

12 x 3

9 x 4

Perfect Square

Style facts

37

2

3

4

6

12

18

$j <= \sqrt{num}$

$j * j <= num$

10 18

10 80

2

max = 37

C

38

2 → 2x18

3 → 3x12

4 → 4x9

5 → X

6 → 6x6

7 → X

8 → X

9 → 9x4

10

11

12

13

14

15

16

17

18

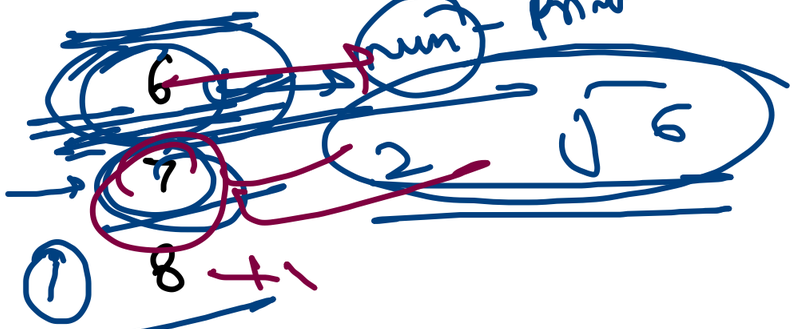
Ignore facts

Discard

$j <= \sqrt{num}$

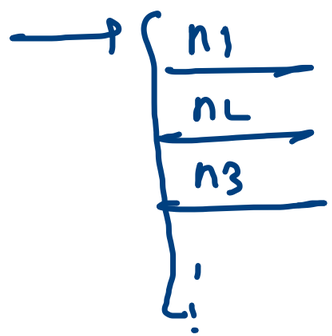
$\sqrt{num}$

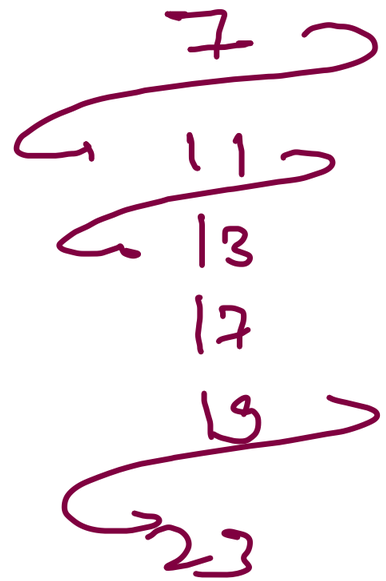
lo = 6 , hi = 24.



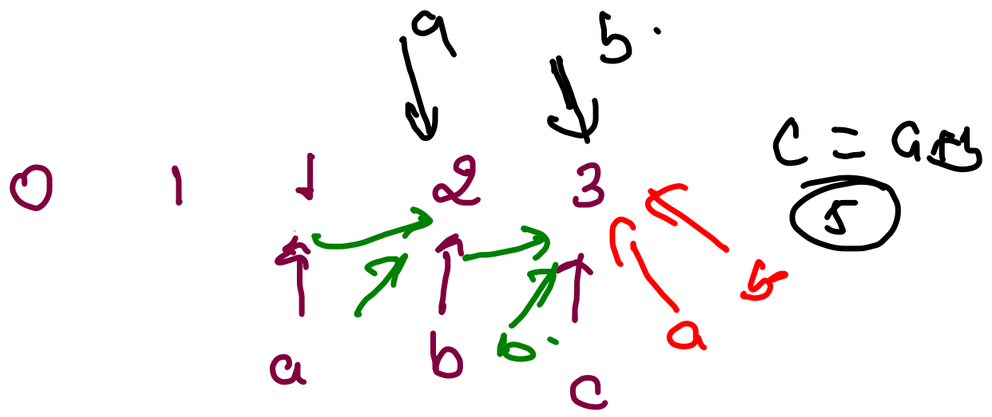
15
16
17
18
19
20
21
22
23
24

t - test cases





n = 10



a = ~~1~~ 3

b = ~~2~~ 3

c = a+b = 3

a = ~~1~~ 2

b = ~~2~~ 3

c = a+b = 3

b = c;

a = b;

a = b;

b = c;

X

15

```
n = scn.nextInt();
int a = 0;
int b = 1;
if(n >= 1){
    syso(a);
}if(n >= 2){
    syso(b);
}
for(int i = 3; i <= n; i++){
    int c = a+b;
    syso(c);
}
```

0 1 1 2 3 5 8 13 ..

a b

int a = 0

int b = 1

1 → print(a); update

2 → print(a); update

3 → print(a); update

4 → print(a); update

5 → print(a); update

6

7

8

9

10

Update.

int c = a+b;

a = b;

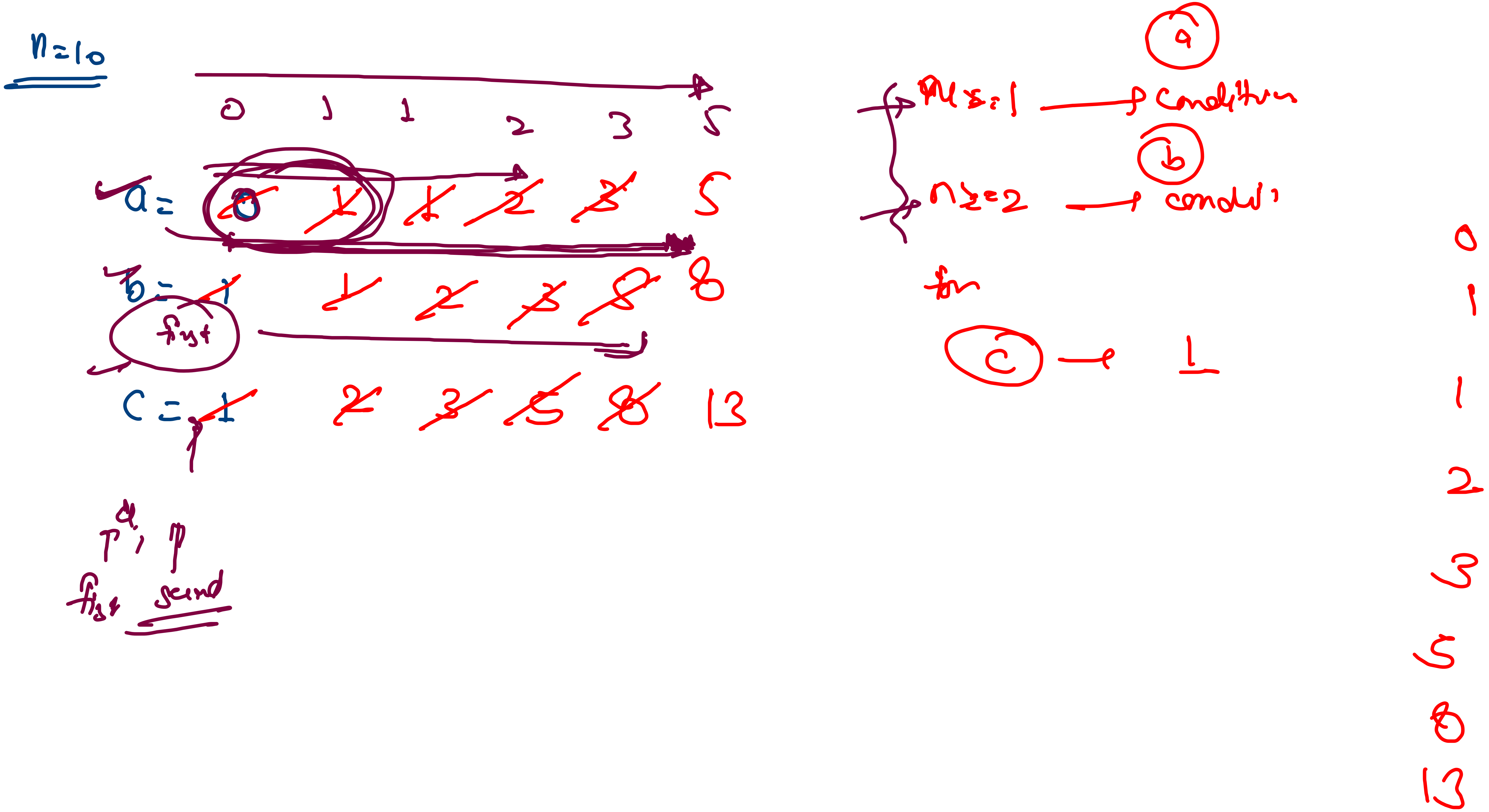
b = c;

0 1 1 2 3 5

a = 0 1 1 2 3 5

b = 1 1 2 3 5 8

i = 1; i <= 10; i++



n=10
0 1 1 2 3 5
a = 0 1 1 2 3 5
b = 1 1 2 3 5 8
c = 1 2 3 5 8 13
first
second
a
n1=1 → condition
b
n2=2 → condition
for
c → L
0
1
1
2
3
5
8
13

6 5 7 8 4 3 8 3

no. of digit ??

Splitting - based

Num : 6 5 7 8 4 3 8 3 → 10

count = count + 1 → 1

6 5 7 8 4 3 8 → 1

6 5 7 8 4 3 → 1

6 5 7 8 4 | 3 → 1

4 | 3 → num = 43/10 = 4

6 5 7 8 | 4 → 1

18 | 3 → num = 183/10 = 18

6 5 7 | 8 → 1

num = 18/10 = 1

6 5 | 7 → 1

6 | 5 → 1

6 → 1

no. of digit

integer = integer / integer

float = integer / integer

float = integer / float

float / integer

OR

```java
int num = scn.nextInt();

// splitting num = num / 10;

int count = 0;
while(num != 0){
    num = num / 10;
    count = count + 1;
}

System.out.println(count);
```

false

Count = count(f);

count += 1;

operator

a = a operator b;

count++;

a operator = b;

int count = 0 1 2 3 4 5 6 7

num = 6 7 5 8 3 2 9

num = 6 7 5 8 3 2

num = 6 7 5 8 3

num = 6 7 5 8

num = 6 7 5

num = 6 7

num = 6

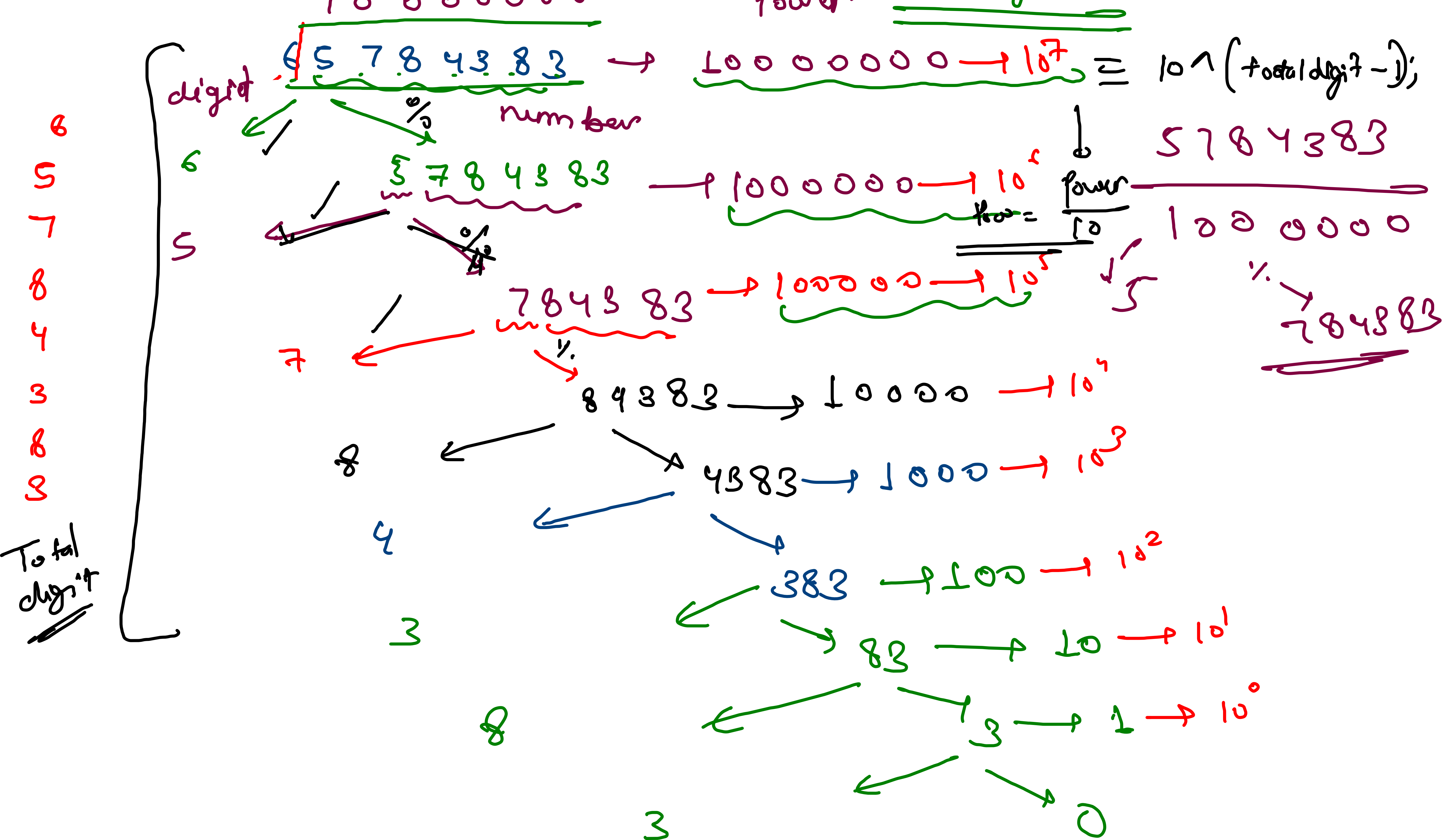
num = 0

7

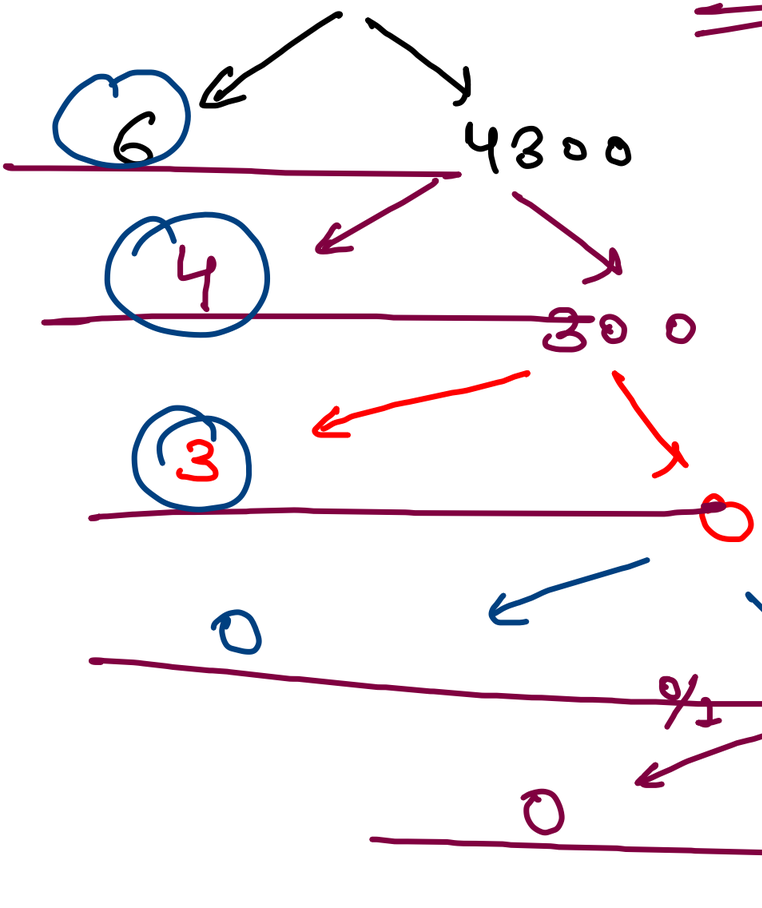
$\frac{6}{10} = 0.6 = 0$

$\frac{6}{10} = 0$

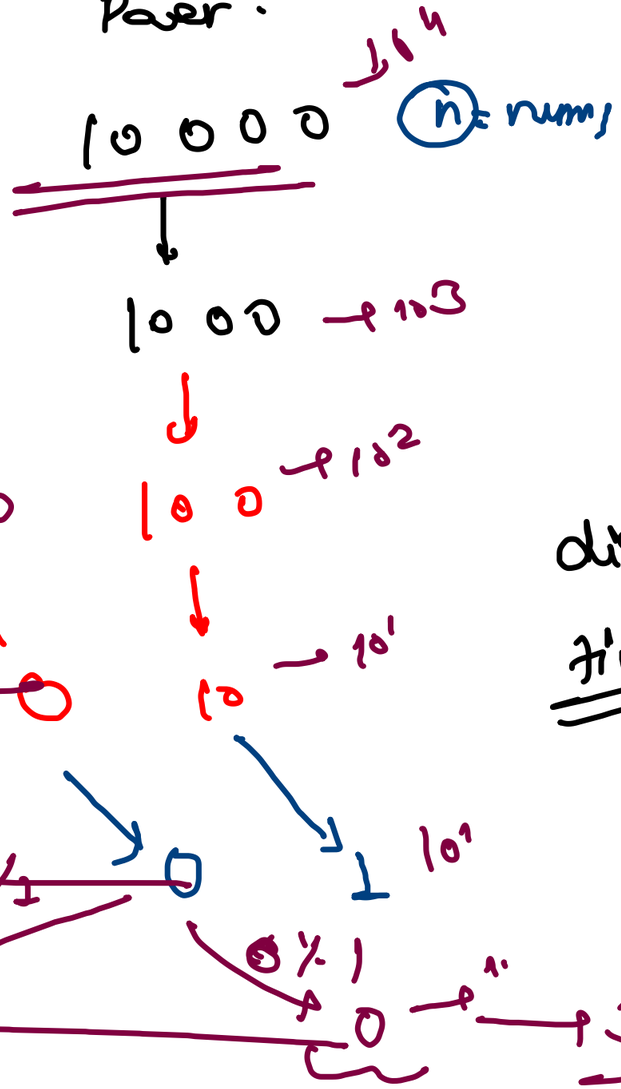
Digits of a Number.



num = 64300



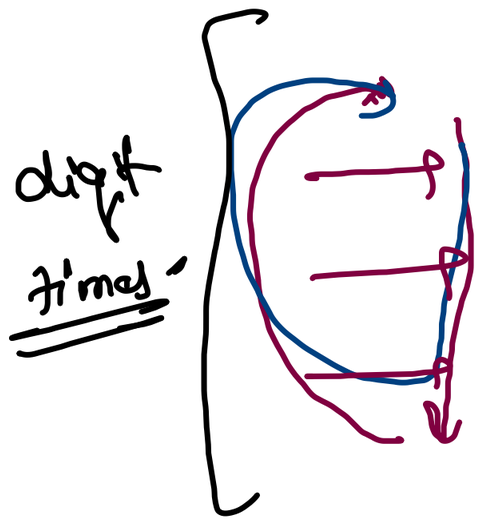
Power.



total digit = 5.

power = $10^{(t.d-1)}$ = 10^4

= 10000



digit = num / power.

num = num % power;

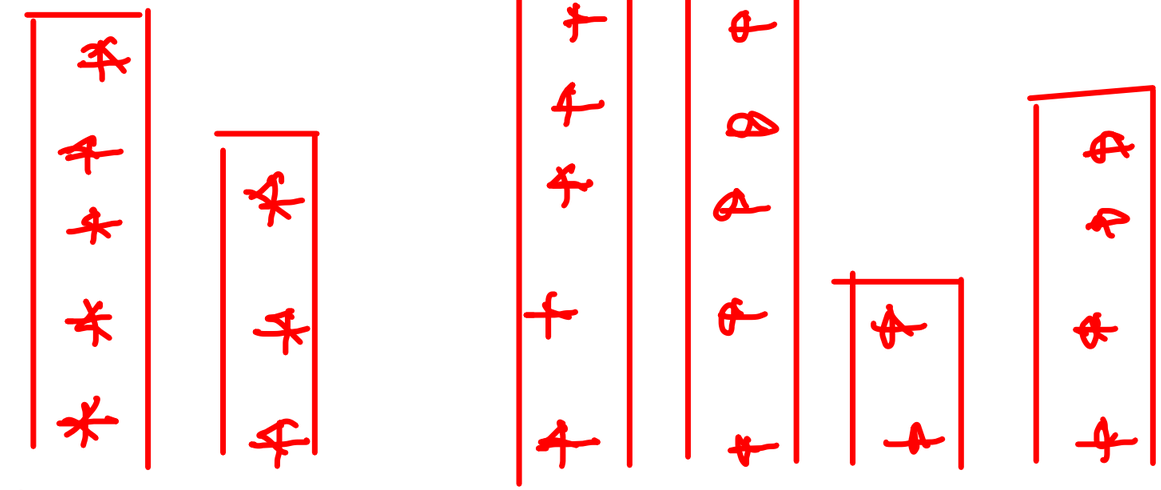
power = power / 10;

Steps.

→ count digit + power.

→ loop.

- digit
- numb
- power reduce



array →

5	3	0	7	6	2	4.

```java
// cd and power
int cd = 0;
int power = 1;
int n = num;

while(n != 0) {
    n = n / 10;
    power *= 10;
    cd++;
}

for(int i = 1; i<= cd; i++) {
    power = power / 10;
    int d = num / power;
    num = num % power;
    System.out.println(d);
}
```

6 1001

cd = 5

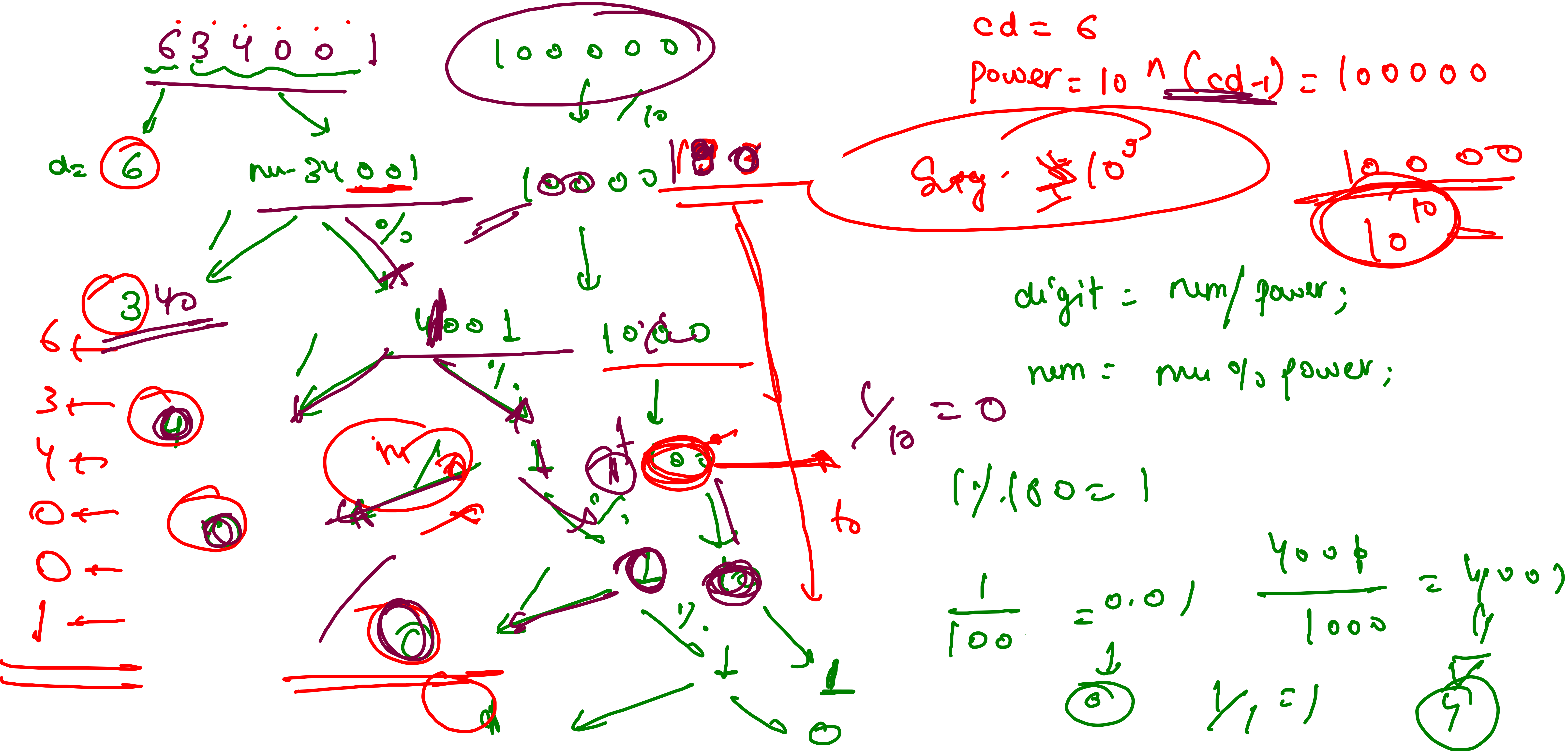
power = 100000

	Power	d	Num
1	10000	6	1001
2 →	1000	1	1
3 →	100	0	1
4 →	10	0	1
5 →	1	1	0

1001 = 1

1000

1001



6 3 4 0 0 1
100000
/10
d = 6
num = 34001
%
10000
1000
100
10
1
4001
cd = 6
power = 10 ^ (cd-1) = 100000
digit = num/power;
num = num % power;
1/10 = 0
1/100 = 0.01
4001/1000 = 4.001
1/1 = 1

5

num = 6 5 4 3 8 | 2

12347

num/10

6 5 4 3 | 8

num%10

2

num = 1 2 3 4 5

num/10

num%10

8

for(int i=0; i<num; i++) {

6 5 4 3

7. 10:30

i=0

i=1

i=2

i=3

i=4

i=5

}

= 12345

(1) → 64325

(2) → 64 325

(3) → 3 25

2 5

5 0

(4) →

(5) →

1000

power = 1000

d = num / power.

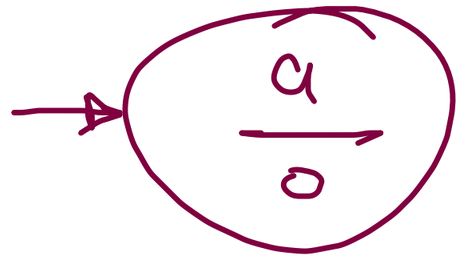
100

10

$\frac{1}{10} = 0$

D/0 = Arithmetic

Entropy



binary.