

TODO:

Modulo distribution over division

4 Extended Euclid's cologorithm

```
func power Of ( a: Int, b: Int) -> Int
       return pow (a,b)
  func power of Ca: Int, b: Int) > Int {
   long let axs = 1
              ans *= ans
```

$$-9002 \Rightarrow 9e3t2$$

$$-0.0004 \Rightarrow 4e4$$

$$-1.002 \Rightarrow -2e-3-1$$

$$-49.00075 \Rightarrow -75e-5-49$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$= (4+5) \% 6 = (4+5) \% 6$$

$$= (4+5) \% 6$$

$$= (9\% 6) = 3$$

$$= 3$$

$$(a+b) \% M = (a \% M - b \% M + M) \% M$$

$$(1-5) \% 6 = 2 (1 - 5 + 6) \% 6$$

$$= 2 \% 6 = 2$$

$$= 2 \% 6 = 2$$

$$= (a \% M + b \% M) \% M$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$(a+b) \% M = (a \% M + b \% M) \% M$$

$$(5/6)/.2 = (5/.2) 2$$

$$= (2)/.2$$
 (will wo for world)

e modulo distribution over division y Extended Euclid's algorithm

> Use >M where the operation is happening.

like anc = (ans * a) > M

where ever there is possibility of overflowing.

The by default e is double, type cast it to inf (16917) int a = 166, int b = 167

int c = a*b × (does not work)

dong long c = a+b (etil not worth)
(s as intermediate resultis

· long long a = le6, b = le7 long czaxb will woll but no need for both a, b to be long long int b = letInt a = 1e6 (long long) a * b long long c type casting. divisors count ('2 C3) 1,2,3, 4, 6,8,12,24 (8)

divisor (ount (N:int) \rightarrow Int var count =0 for i in 1... N \S if N:1=0 \S count t=1

> constraints
$$1 \le N \le 10^9$$

Niferations = 10^9 iterationy

 $1 \le N \le 10^9$ instructions [sec

 10^9 iteration ≤ 1 sec 2×1

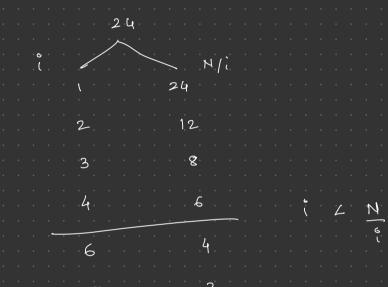
1 iteration ≥ 5 instructions

 10^9 iterations $\Rightarrow 5$ secs

Now if $1 \le N \le 10^8$

10 iterations
$$\Rightarrow$$
 2 sec
 2×10^9 iterations \Rightarrow 2 sec
 $10^9 \times 10^9$ iterations \Rightarrow 10^9 sees
 10^9
 $60 \times 60 \times 24 \times 365$ yrs \Rightarrow
 $= 31.71$ yrs





return court. Cint N) int arro, b = 40; long long C = 100; float z = 2.5 5 iterapor Not dependent on N Constant time complexity O(1)

func solve (Int N)
$$\xi$$

N = 10° 1 10 ξ

Int a = 20, b = 40;

long long c = 100;

float $z = 2.5$

for i in 0... 5.5

X ξ

N X ξ

N ξ

Linear Hime complexity ξ

for i in 0... ξ

for i in 0... ξ

N ξ

For i in 0... ξ

for i in 0... ξ

N ξ

N ξ

For i in 0... ξ

N ξ

The for i in 0... ξ

 $N^2 + N \approx O(N^2)$

$$N = 10^3$$

$$\frac{N}{N^2} \times 100 = \frac{10^3}{10^6} \times 100 = 0.1\%$$

N=104 -> 10-2 -> 001 /. with increasing value of N2, contribution of Nisles.

func solve (N:Int) { $1 \rightarrow \text{then } O(.10.9 \text{ N})$ for 1... $0 \rightarrow ... \rightarrow$

poten
$$2^{K} = N$$
 $\log_{2}^{2} = \log_{2}^{2} N$
 $\log_{2}^{2} = \log_{2}^{2} N$

linear Search linear Search Car: [Int], N: Int, key; Int) { > Bool in o arrount 1 { → OCN)

A Big - O defination:

return true

return false

based on the input size after a certain throshold.

Puts an upper bound on complexity of an algorithm

· Best case = O()

· Average case = O(N/1)

· Morst case = OCN)

upper bound

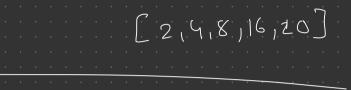
if arci] == K &

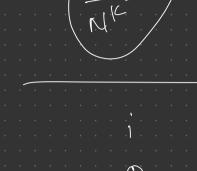
$$\begin{bmatrix}
 a & b \\
 a & b \\
 b & c
 \end{bmatrix}
 = b - a$$

$$\begin{bmatrix}
 ca & b \\
 ca & b$$

N (0-N) N

$$N = \{0-N\}$$
 N/2 ($\log N$)
 $N = \{0-N\}$ N/2 ($\log N$)
 $N = \{1-1\}$ N/2 ($\log N$)





1 GHZ = 109 inst/sec

Bigo	N^3	N ²	log N	N ²	Nlog N	N	Ju	logn	1
Hiterations	10/8			1012					
	31.7			1000					
Time	yrs			<i>s</i> ec					· · ·
	NS	50 1	1=30						

Errors	
Mrong	Answer - Edge Cases
	-> Overflow (datatype)
* TLE	Infinite loop
	Happens more in Brute force appro
A RTE	> out of bound, division by a new pointer, undandled excep

T= 100, N= 30 > Tx2N -) (0² × 2 3) 109,10 7=100/1×=1018 > 1010 > log ? か(o = 7:10°, N:(03 9 10° X (7PM) > precomputation Importance of constaints 1 Data type to use Decide the logic

