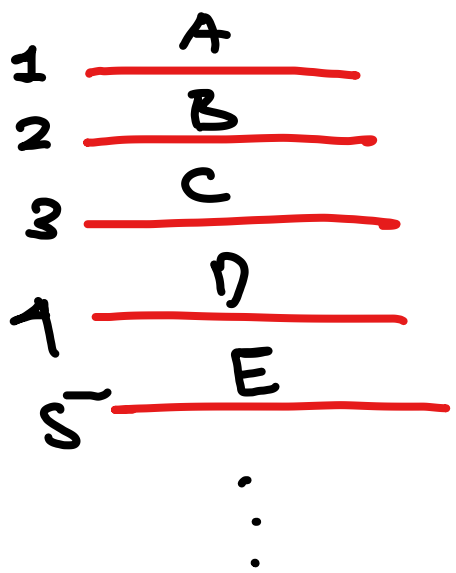
Data Structure

Array, Dynamic Array (List), Stack, Queue

T kali input →

Input baris ke-i

T = 5



Output

A!
B!
C!
D!
E!

$$n! = n * (n-1) * (n-2) * (n-3) * \dots * 1$$

factorial(n)

$$factorial(n) = n * factorial(n-1)$$

$$f(0) = 1 \quad \vee \quad f(1) = 1$$

Recursion

Base case

$$f(n) \rightarrow f(n) \rightarrow f(n) \rightarrow \dots$$

Max Recursion Depth

$$f(n) = 2f(n-1)$$

Recursão

Jumlah iterasi → Perenggian fungsi

T = 5

1 ⟶ faktorial(1) = 1 → 1 * iterasi

2 ⟶ f(2) = 2 * f(1) → 2 * 1 → 2 * iterasi

3 ⟶ f(3) = 3 * f(2) → 3 * 2 * f(1) → 2 * iterasi

memo

4 ⟶ f(4) → f(3) → ~~f(2) → f(1)~~

5 ⟶ f(5) → f(4) → ~~f(3) → f(2) → f(1)~~

Stop di Base case

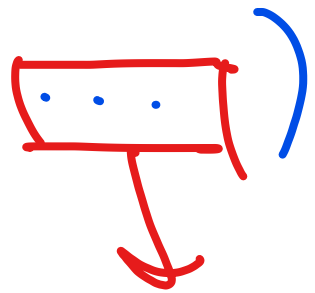
⋮

N → f(N) → f(N-1) → f(N-2) → ... → f(1) → N * iterasi

ada T input dan setiap input akan iterasi sebanyak N x

Total = N * T

O (N * T)

O ()

constraint

10^8 ≃ 1 detik

Faktorial (100) < 1 detik

O (N)

$100 < 10^8 \rightarrow$ < 1 detik

O(N)

Faktorial (1000.000.000) > 1s

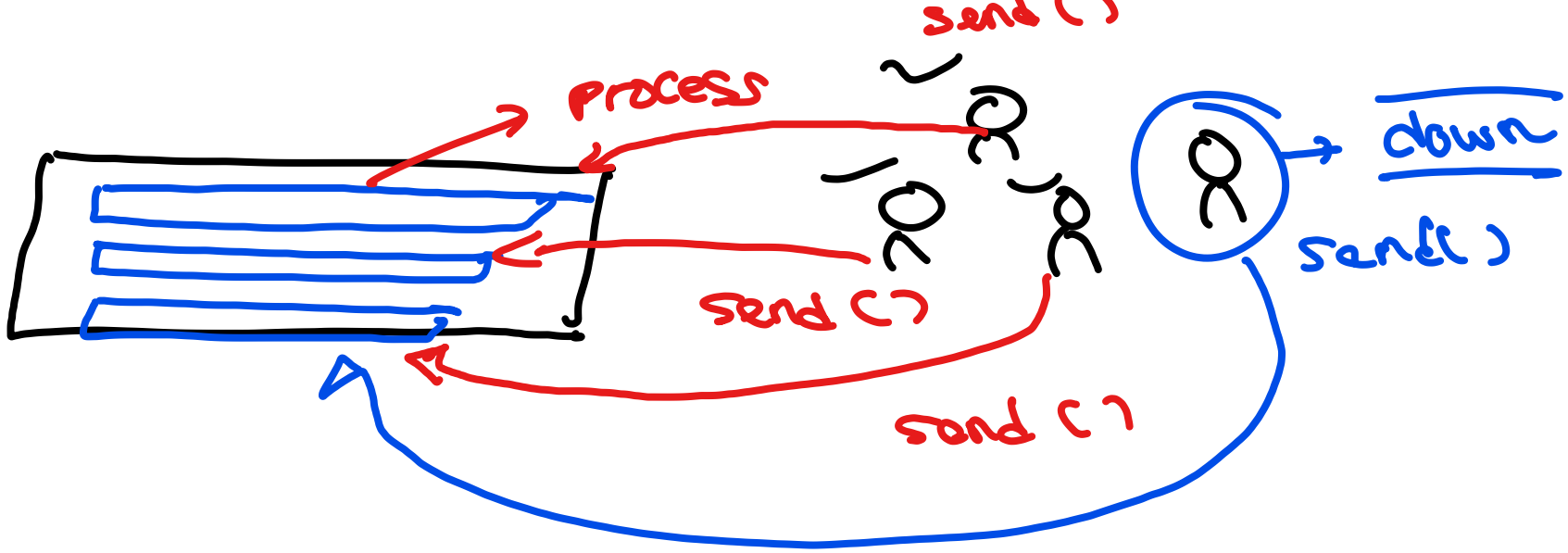
$> 10^8$

O(N * T)

$N_{max} = 10^3$

$T_{max} = 10^7$

> 1 Jahr



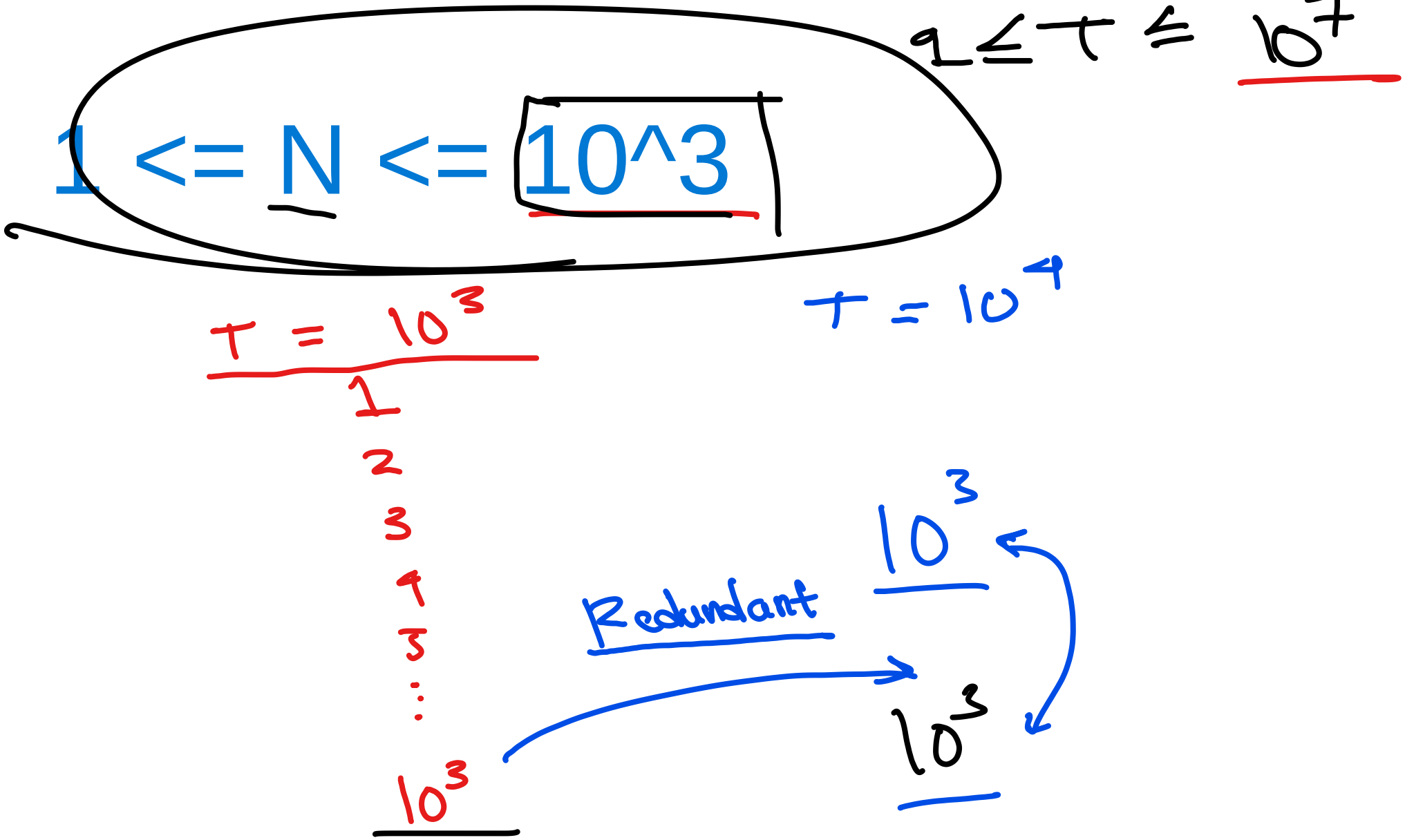
handling Recursive Worst Complexity

(1) Pre - compute

(2) Memo → dynamic programming < top-down / Bottom-up

(3) Threading

Technical


1 <= N <= 10^3
1 ≤ T ≤ 10^7
T = 10^3
T = 10^4
1
2
3
4
5
10^3
Redundant
10^3
10^3

Pre compute

$1 \leq N \leq 10^3$

$O(T + 10^3)$

hiring

data

$T_{max} = 10^7$

$O(10^7 + 10^3)$

$< 10^8 \rightarrow$ < 1 detik

Analisis Kompleksitas => Seberapa cepat programmu jika dijalankan (Sudah dicompile)

Rules of thumb → O(...)

→ $10^8 \simeq$ 1 detik

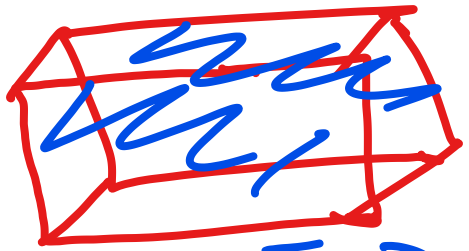
Jumlah iterasi ⇒ upper bound() → sebuah-bukanya-

Sebuah-lah bat

Big-O Notation O(...)

CPU → 101010 0001

GPU



CUDA → C++

- Sesuatu yang konstan
1) Deklarasi Variable
2) if-else
3) Operasi hitung

O(1) → tidak ada for loop

- Linear

```
for i in range(1,100):
    print("Hello")
```

→ 99 iterasi

```
for i in range(1,N):
    print("Hi)
```

→ N-1 iterasi

O(N-1) = O(N)

```
for i in range(1,x):
    print("Oke")

for j in range(1,y):
    print("Hi")
```

O(x+y)

```
for i in range(1,x):
    for j in range(1,y):
        print("Oke")
```

O(xy)

- Polinomial

```
for i in range(N):
    for j in range(N):
```

O(N²)

N = 5 →

i = 0
i = 1 } iterasi 1
j = 0

i = 2 } iterasi 2
j = 1
j = 2
⋮

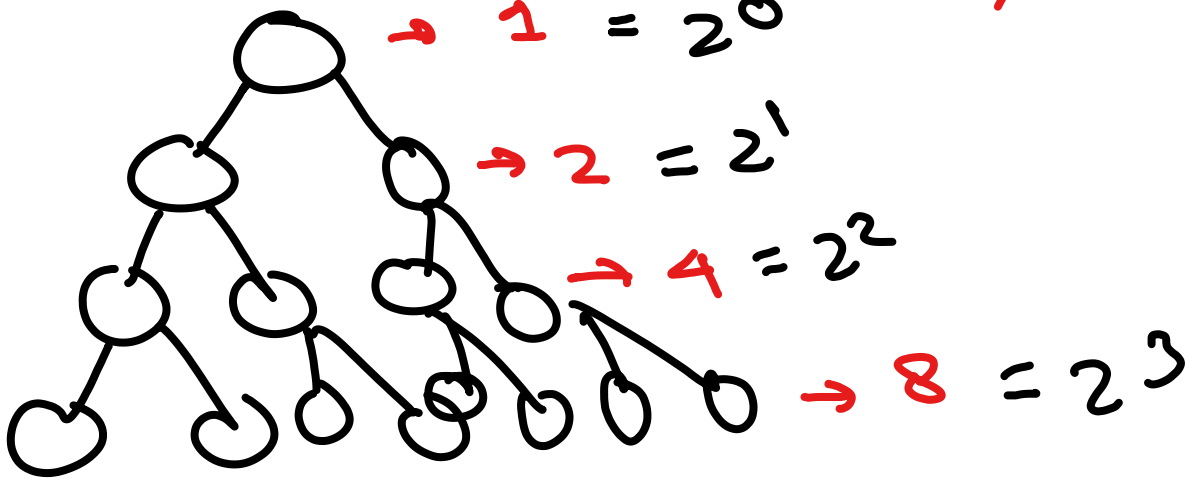
```
for i in range(N):
    for j in range(i):
```

Jumlah iterasi = $0 + 1 + 2 + 3 + \dots + N = \frac{N * (N+1)}{2}$

$O\left(\frac{N * (N+1)}{2}\right) = O\left(\frac{1}{2}N^2 + N\right) = O(N^2)$

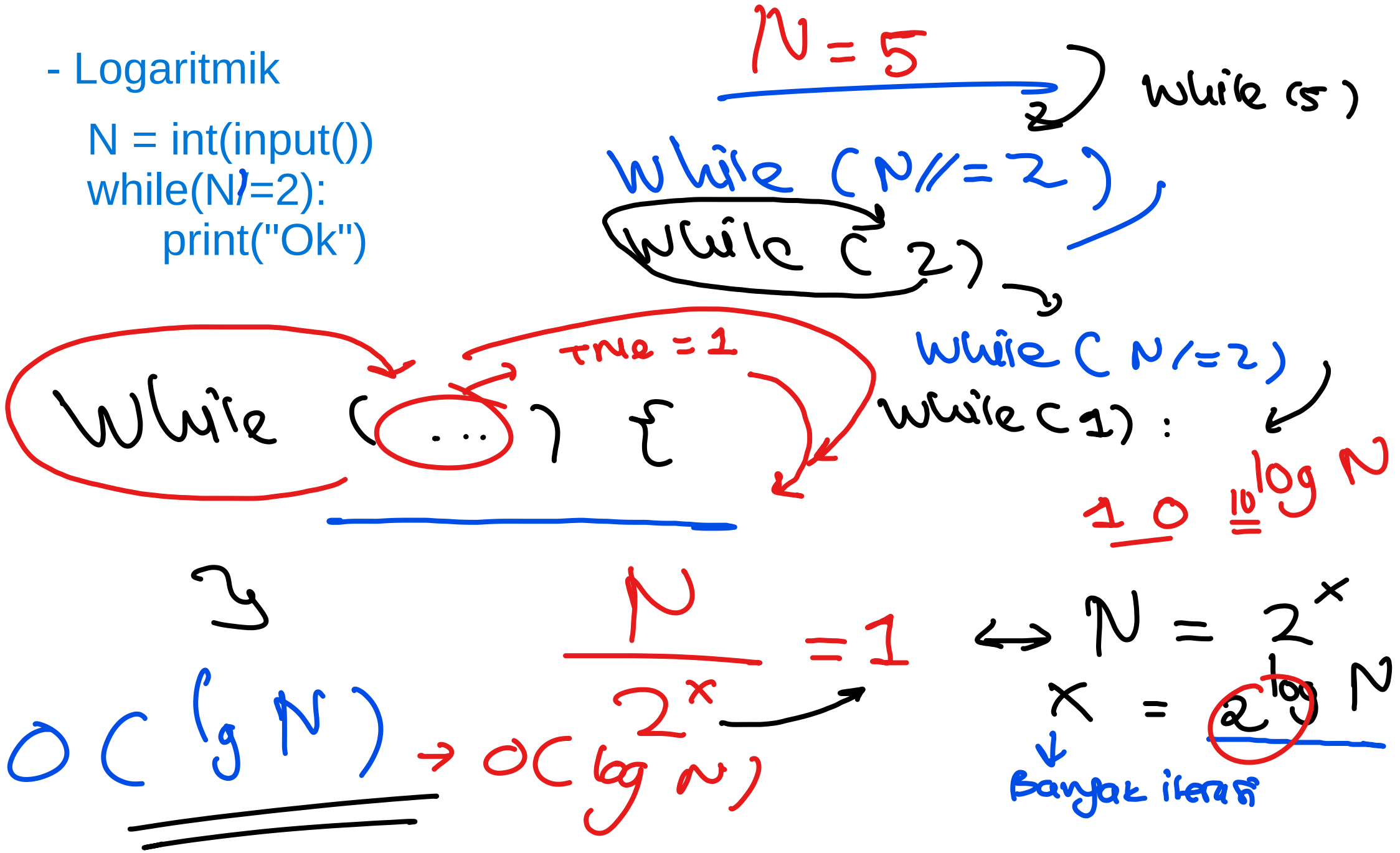
-Ekponensial

$O(2^N)$



- Logaritmik

```
N = int(input())
while(N!=2):
    print("Ok")
```

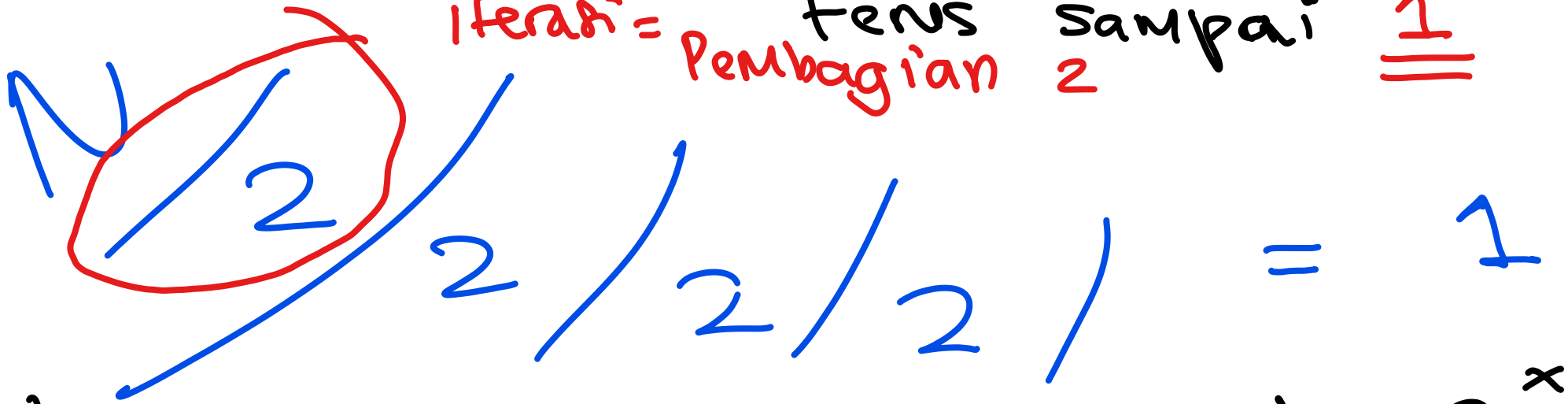


Banyak iterasi = Banyak Pembagian 2

While Loop ⇒ akan bagi 2 N

tetap sampai 1

iterasi = Pembagian 2



$N * 2 * 2 * 2 * \ldots * 2 = N * 2^x$

sebanyak x

$N * 2^{-x} \rightarrow N / 2^x$

$N / 2 / 2 / 2 / 2 / \ldots / 2 = 1$

sebanyak (x) jumlah iterasi

$N / 2^x = 1$