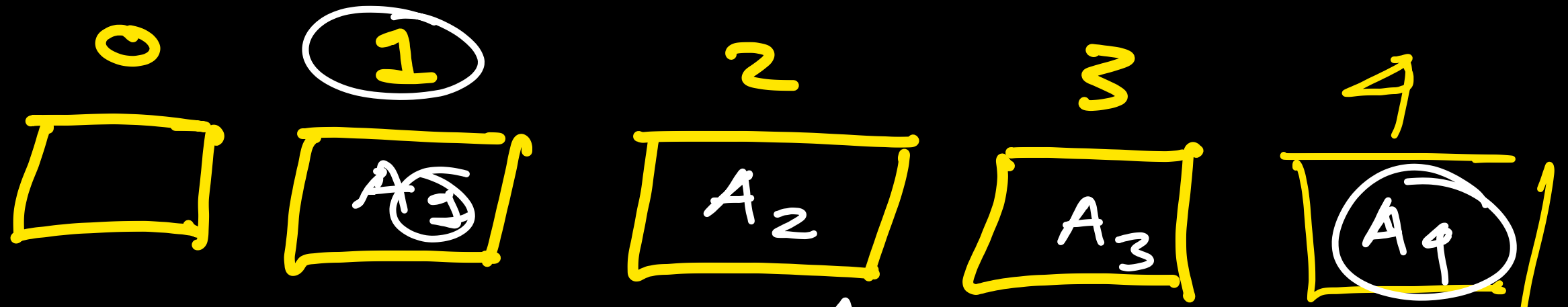


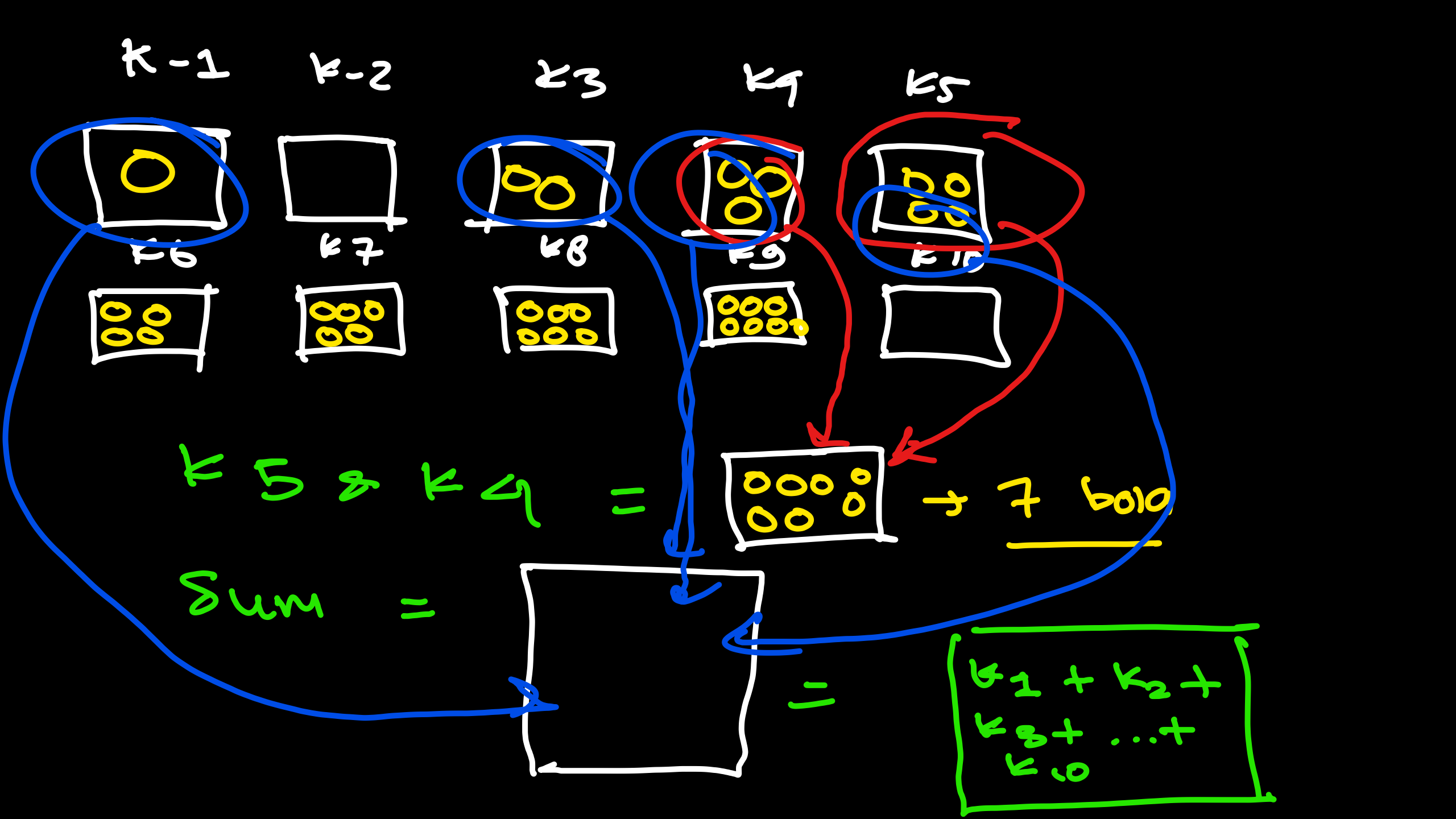
N = 4

N + 1 = 5

A0, A1, A2, A3
A1, A2, A3, A4

Out of bound

indeks = urutan - 1


k-1
k-2
k3
k4
k5
k6
k7
k8
k9
k10
k5 & k4 =
7 bolo
Sum =
=
k1 + k2 + k3 + ... + k10

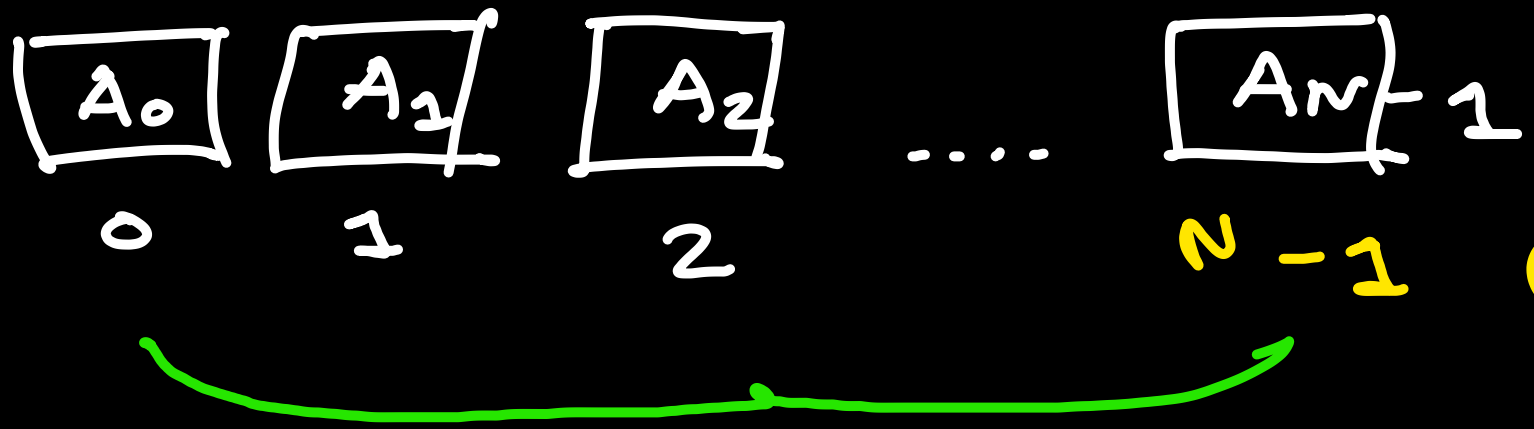
Kita keluarkan semua bola dari masing - masing kotak

Cetak semua isi kotak urutan ke-i untuk (1 <= i <= 10)

Index → $0 \leq i \leq 9$

$0 \leq i < 10$ } sama aja bro

Ada N kotak, masing - masing kotak ke-i, memuat bola sebanyak Ai



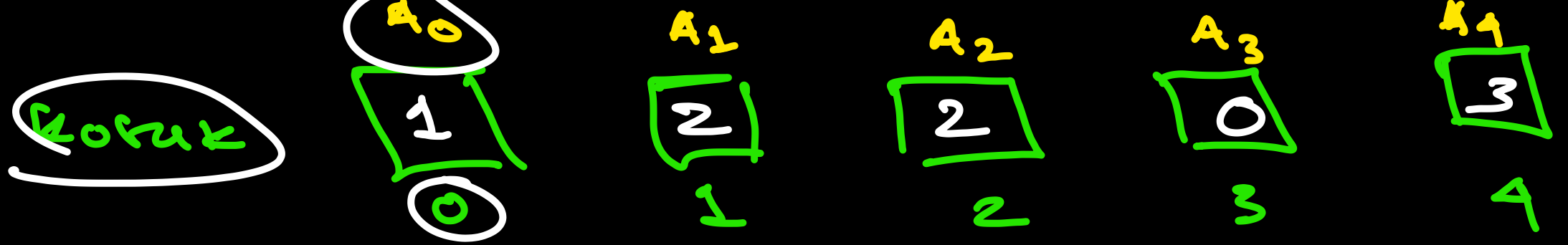
kotak[i] = A[i]

kotak[0] = A[0]
kotak[1] = A[1]

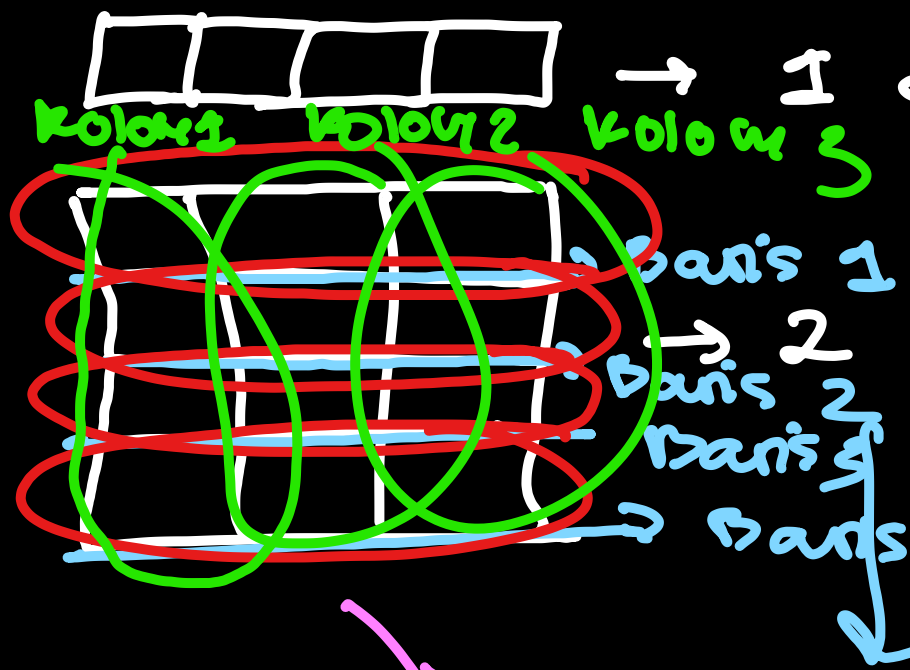
1 to A1

N

A = {1, 2, 2, 0, 3} → N = 5 (5 kotak)



Array 2 dimensi → tabel, kolom & baris, matrix



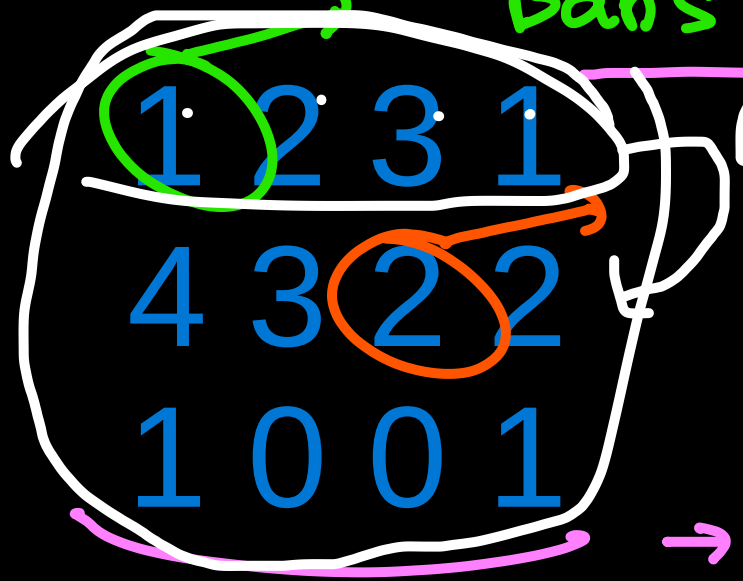
→ 1 dimensi 1 x .... satu (index - i)

→ 2 dimensi r x c dua (index (i,j))

Kolom →

Baris

4 baris, 3 kolom → 4 x 3



Baris = 1 , kolom = 1 ukuran

endl

3 x 4

index = (0,0)

→ A → A[0][0]

koordinat kartesian

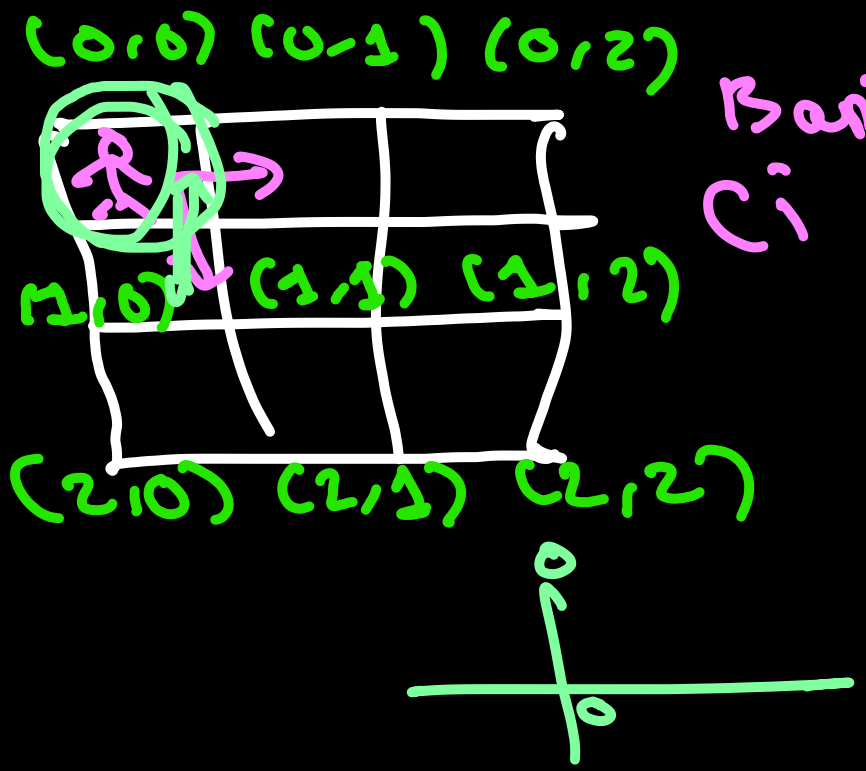
Grid

A[1][2] index: (1,2

Baris = 2, kolom = 3

i = 2-1 , j = 3-1

= 1 , j = 2



Baris, kolom
(i, j)

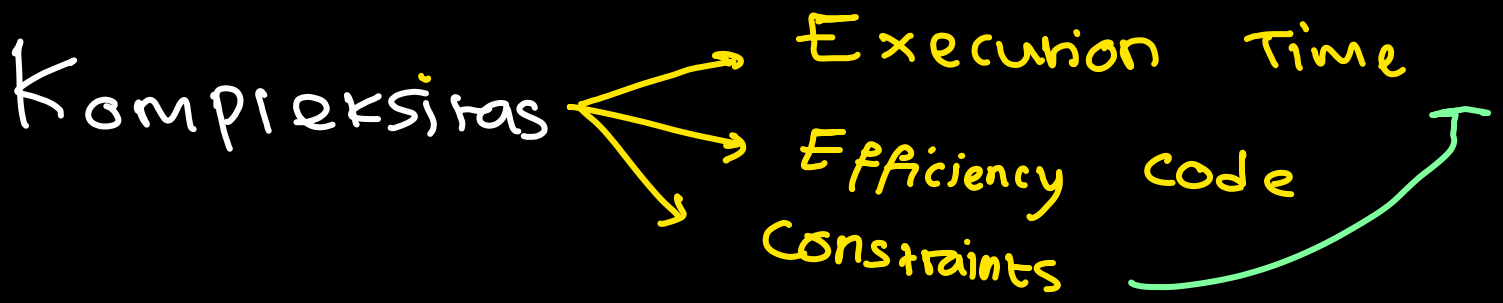
Kanan baris kolom dari pojok kiri atas

Kanan : (0,0) → (0,1)
(x, y) → (x, y+1)

Kiri : (0,1) → (0,0)
(x, y) → (x, y-1)

Atas : (1,0) → (0,0)
(x, y) → (x-1, y)

Bawah : (0,0) → (1,0)
(x, y) → (x+1, y)



Big-O : O(....)

↳ kompleksitas

* O(1)

- cout<<"Hello"<<endl;
- int x = 1;
  x += 2;
  cout<<x<<endl;
- Sebuah operasi yang dapat dijalankan dalam satu proses itu akan memiliki kompleksitas yang linear

```
int a, b, c, d;
a = 1, b = 2, c = 3, d = 4;
a += 1;
b += a;
....
c += 9;
cout<<...<<endl;
```

- if, else, operand lainnya
- Selama tidak ada loop, tidak ada recursive function, tidak ada Bounding, Pointer Accessing.

} Linear O(2,3,..,5)

O(1)

O(f(N))

N = Value yang diterima → Input / Jumlah operasi

```
int N;
for(int i = 1; i <= N ; i++){

}
```

O(N) → karena loop sebanyak N x

```
for(int i = 1; i <= N; i++){
  for(int j = 1; j<= M; j++){

  }
}
```

O(NM)

for Pertama → i
for Kedua → j

i = 1 → for j
j = 1
j = 2
. . . .
j = M

i = 2 → for j
j = 1
j = 2
. . . .
j = M

. . . .

i = N → for j
j = 1
j = 2
. . . .
j = M

Untuk Setiap i, ia akan menjalankan loop sebanyak M kali.

Sekarang banyaknya i adalah N.
Total Loop (Aturan Perkalian) = Loop i x Loop j = N x M.

→ O(NM)

$O(N^2)$ → double for dgn constraint

```
for(i = 1; i<= N; i++){
  for(j = 1; j<=N; j++){

  }
}
```

$1 \le i \le N$

$1 \le j \le N$

```
for(i = 1; i<= N; i++){
  for(j = 1; j<=N; j++){
  for(j = 1; j<=N; j++){
  for(j = 1; j<=N; j++){
    ..... (Sebanyak k)
  }
  }
  }
}
```

$O(N^k)$ → exponential

```
int N;
while(N){
N /= 2;
}
```

bakalan berhenti saat N=0 → while false

~~$O(N/2)$~~

$O(\log N)$

N = 10 →

10/2

5/2

2/2 = 1

1/2 = 0

loops = 4

N = 12

12/2

6/2

3/2

1/2

loops = 4

```
int N;
while(N){
N /= 2;
}
```

Program akan melooping
N dibagi 2 terus sehingga habis → $N/2/2/2/2/2/\ldots \rightarrow 0$

$$\frac{\frac{N}{2}}{2} \rightarrow \frac{N}{2} \times \frac{1}{2} \rightarrow \frac{N}{2^2}$$

$$\frac{\frac{\frac{N}{2}}{2}}{\vdots \; 2} \rightarrow \frac{N}{2^K} \rightarrow \frac{N}{2^K} \approx 0$$

} sebanyak (K) → Jumlah loop

kompleksitas → berapa kali N dibagi 2 ?

$$\frac{N}{2^k} = 1 \; ?$$

$k = 0$

$2^0 = 1$

$N = 2^k \rightarrow k = {}^{2}\log N$

Kompleksitas → $O({}^{2}\log N)$ → logaritmik

$O(\log N)$

Di Ilmu Komputer (Analisis Kompleksitas)
Basis Log -> 2, (Binary 1 dan 0)

1. Binary search
2. DP – Memoisasi

Rule of Thumbs : Komputer Modern umum Bisa memproses $N = 10^8$ operasi dalam 1 detik

Ex: Constraints : $1 \leq N \leq 10^8$

Complexity : $O(N^2)$

$(10^8)^2 = 10^{16}$ > 10^8 (> 1 s)

$O(N) \rightarrow 10^8 \rightarrow$ 1 detik

$1 \leq N \leq 10^{12}$

$10^{12} > 10^8$ (> 1 detik)

Program ($O(\log N) \rightarrow$)

$1 \leq j^2 \leq N \rightarrow$ Banyak loop = Banyak j

Banyak $j = O(\sqrt{N})$

$1 \leq j^2 \leq N \rightarrow \sqrt{1} \leq \sqrt{j^2} \leq \sqrt{N}$

$1 \leq j \leq \sqrt{N}$

* Cek Prima : Cek apakah banyak Faktor = 2 ?
  → Ya = prima
  Bukan = Bukan prima

* Naive Sol : Cari banyak bilangan 1 ≤ ... ≤ bilangan yang habis dibagi

4 → 1, 2, 3, 4 → banyak Faktor = 3 (> 2) Bukan prima

7 → 1, 2, 3, 4, 5, 6, 7 → banyak Faktor = 2 Prima

* Fit Sol : Instead cek $1 \leq \dots \leq N$ → $1 \leq \dots \leq \sqrt{N}$

Kita tahu bahwa 1 & dan bilangan itu sendiri adalah Faktornya → 2 Fak

Faktor = 2 , Faktor = 3 > 2 → Bukan prima

10 → 2, ..., ...

} Kalau genap > 2 udah pasti bukan

16 → 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ..., 15, 16

15 → 1, 2, 3, ~~4~~, 5, 6, 7, ..., 15

stop

77 → 1, 2, 3, 4, 5, 6, 7
stop

65 → 1, 2, 3, 4, 15
stop

N = 10 → stop 2

N = 16 → stop 2

N = 15 → stop 3

N = 77 → stop 7

N = 65 → stop 5

Bilangan yang punya insan faktor dgn N → $\sqrt{N}$

Semua

Bilangan Kalau difaktorisasi prima ... Pasti

> 1 digit

2

Pangkat minimal = 2

fac(5) → 5 * fac(4) → fac(3) → fac(2) → fac(1) 5 *

1 2 3 4 5

fac(6) → fac(5) → 6 *

1 5

fac(x) → fac(x-1) → f(x-2) → fac(x-3) → ... → fac(1)

fac(x) s/d fac(1) : x kali pemanggilan function

O(x)

Program O(ab)

Test Case
T = 5
1 → fac(1) = 1
2 → fac(2) → fac(1) = 2x1
3 → fac(3) → fac(2) → fac(1)
4 → fac(4) → fac(3) → ...
5 → ...

} setiap testcase dia ngitung ulang rekursif

Test Case
T = 5
1 → fac(1) = 1
2 → fac(2) → fac(1)
3 → fac(3) → fac(2)
4 → fac(4) → fac(3)
5 → fac(5) → fac(4)

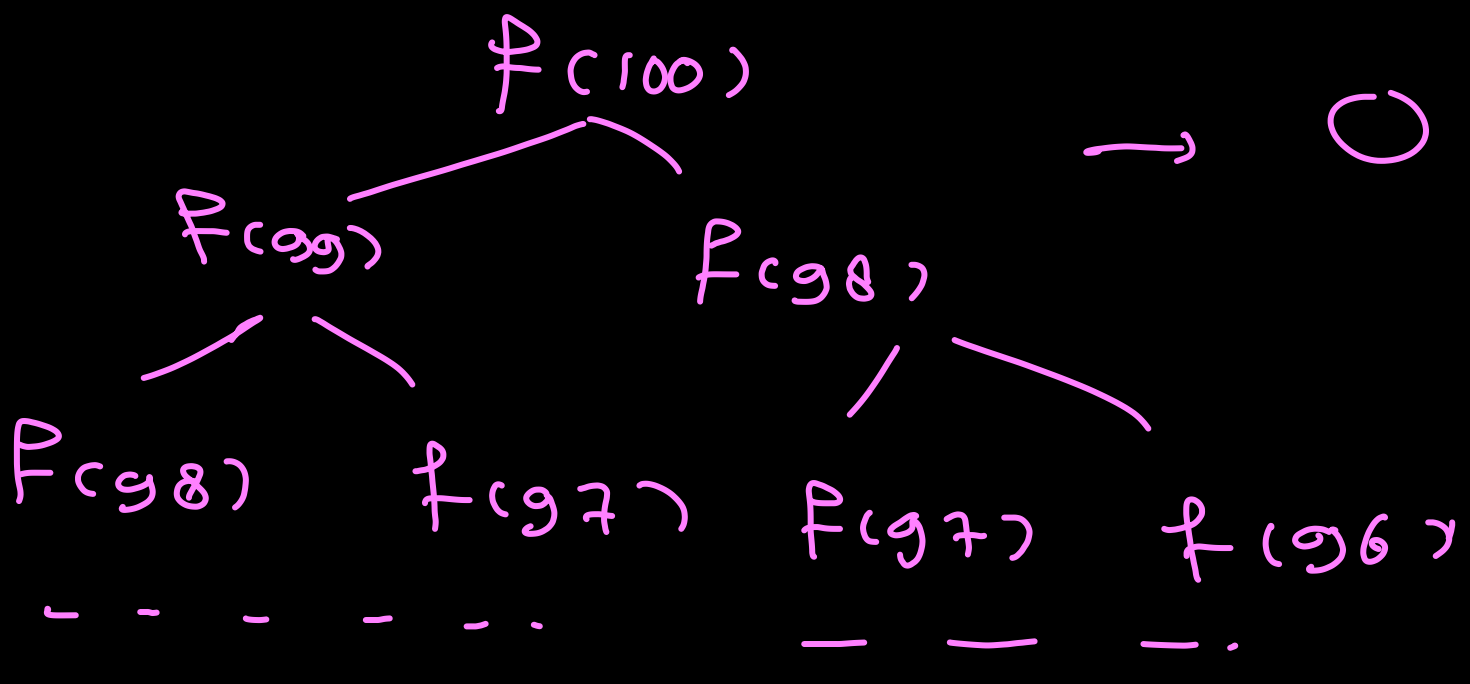
O(a) → 10^7 < 10^8 (< 1 detik)

	1	2	3	4	5
fac	1	2	6	24	120

} Memo

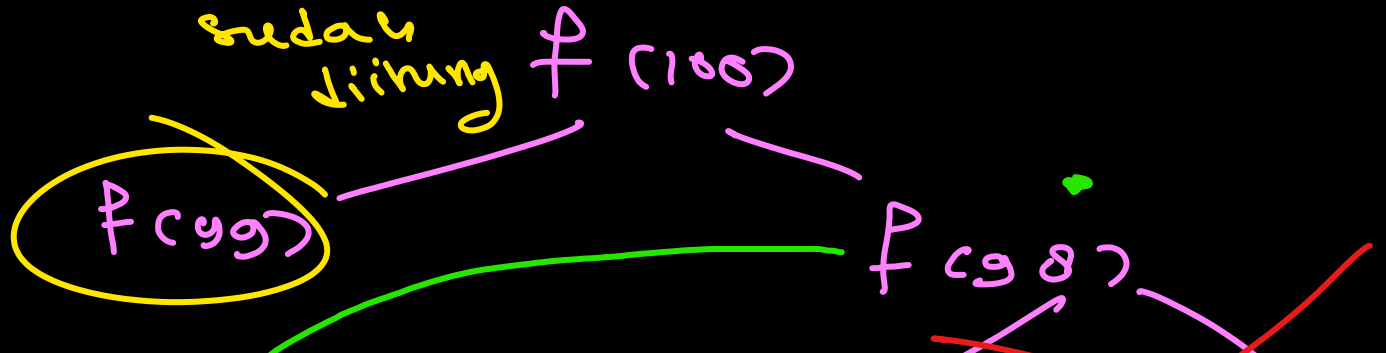
= 2 x 1 = 2
= 3 x 2 = 6
= 4 x 6 = 24
= 5 x 24 = 120

fibo without Memo



→ $O(2^N)$

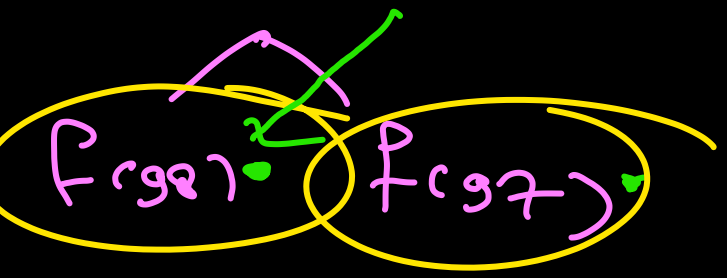
fibo with Memo (Pruning)

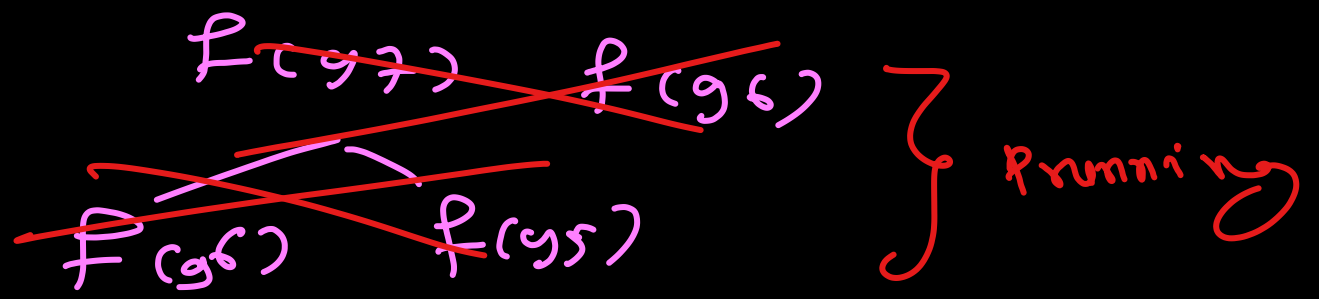


Terburuk $O(N)$

Terbaik $O(\log N)$

Morong 1/2 1/2 1/2


f(98)
f(97)


f(97)
f(96)
f(96)
f(95)
Pruning