: General Q - Priority Q
deq

Struktur Data
- linier : stack and Queue, Array
  - static & dynamic
  - vector = dequeue + fitur tambahan
- non linier : graf, tree

static

x = arr[2] → x = 3

1	2	3	4	5
0	1	2		

Dynamic

Queue

antrian game bola nyelam

1 2 3 4 5

1 2 3 4

1 2 3

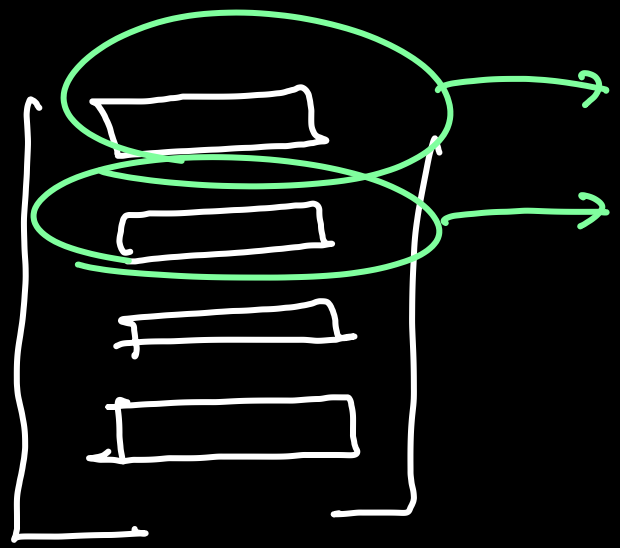
Static Array → Indexing

A[i]

A[r][c]

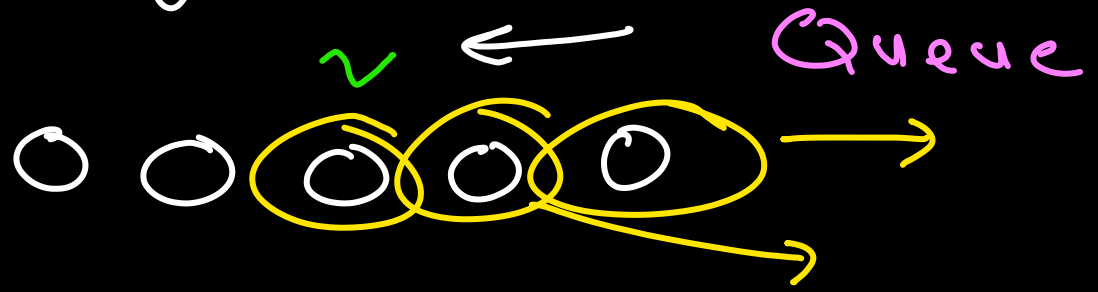
Dynamic Array → Melakukan langkah dinamis

Stack



harus dikeluarin

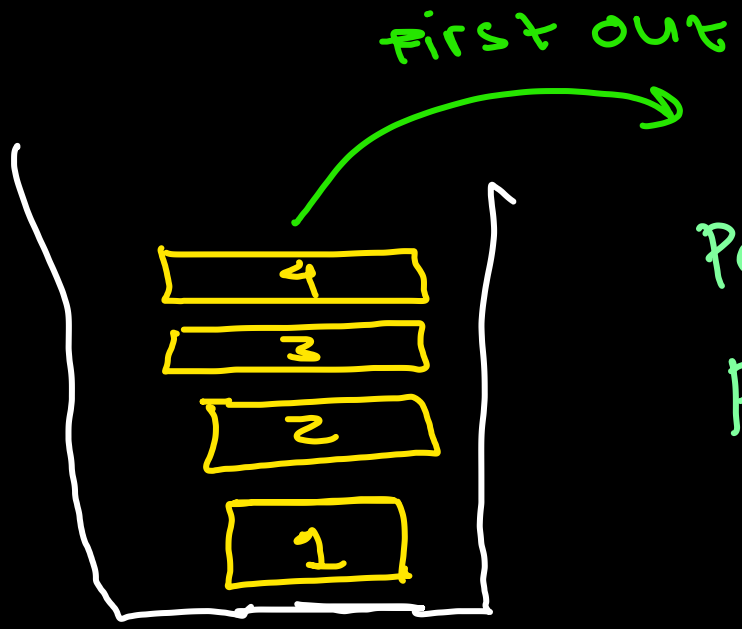
Array = A[s]



* Stack (Tumpukan)

→ First out
→ Last in

→ LIFO



→ dari paling bawah

POP()

PUSH()

Last in First out

* Queue

antrian
Wota
Zee

→

First in
First out

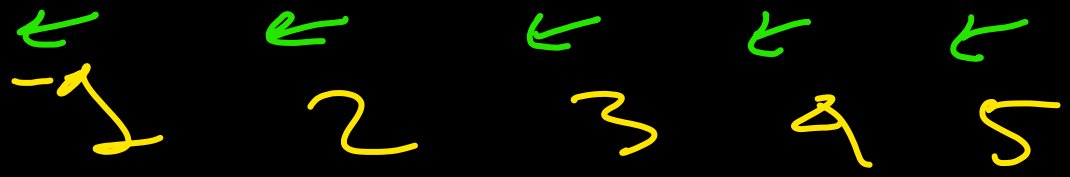
→ FIFO

Pertama datang
Pertama ketemu
Zee

Keluar Pertama
Pertama Pulang

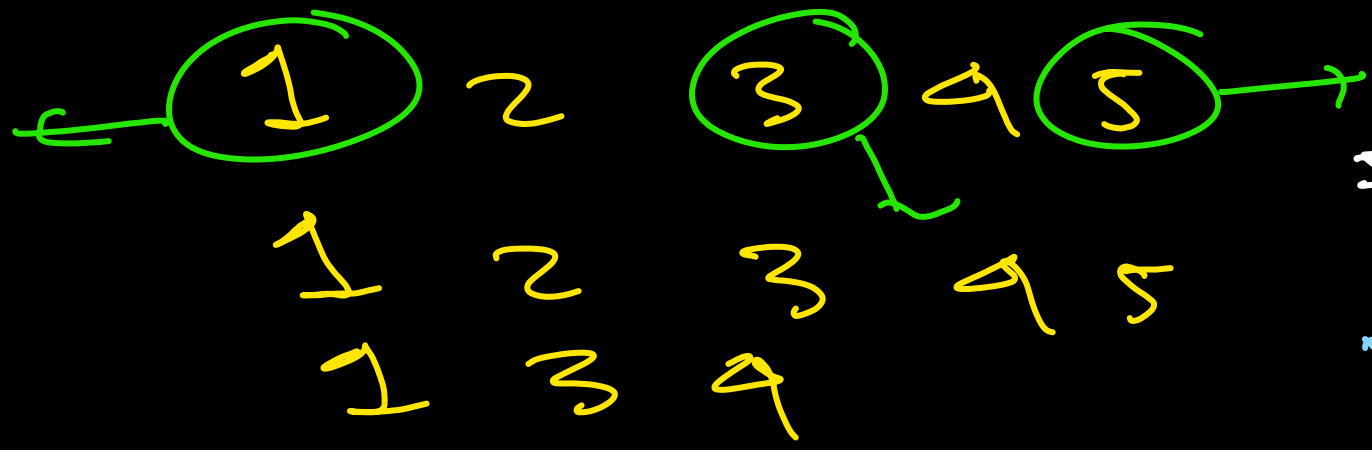
FIFO

General Queue hanya berurutan



(1,2,3,4,5)

DeQueue



5 keluar
2 keluar

Push()
insert()
erase(pos)

hapus dari depan ← Pop()

P. Queue

Q : 1 2 3 4 5

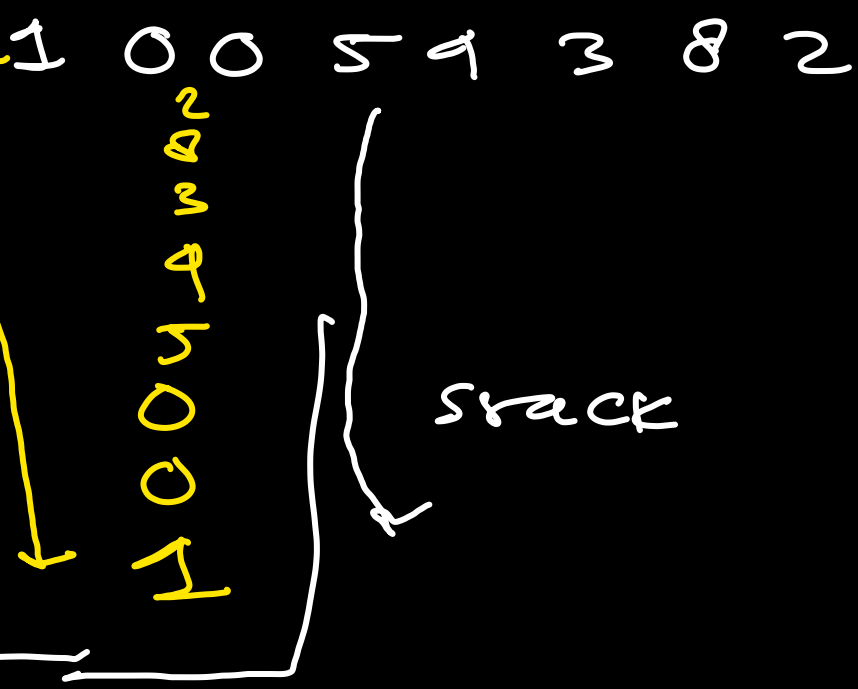
PQ : 5 4 3 2 1

Kita push dari array paling depan

array = [1,0,0,5,4,3,8,2]
stack = []

stack = [2, 8, 3, 4, 5, 0, 0, 1]

Last in – First out



arr = 1 0 0 5 4 3 8 2

Stack.push(arr)
arr.pop()

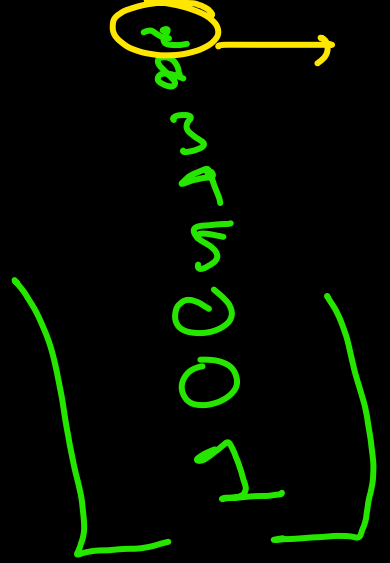
Pop stack → hapus pig blkg

Q pop → hapus pig depan

arr = 1 0 0 5 4 3 8 2

Q = ~~1~~ 0 0 5 4 3 8 2

Q.pop() →



2 8 3 4
5 0 0 1

Q: ~~1~~ 0 0 5 4 3 8 2

Q.pop()
~~0~~ 0 5 4 3 8 2

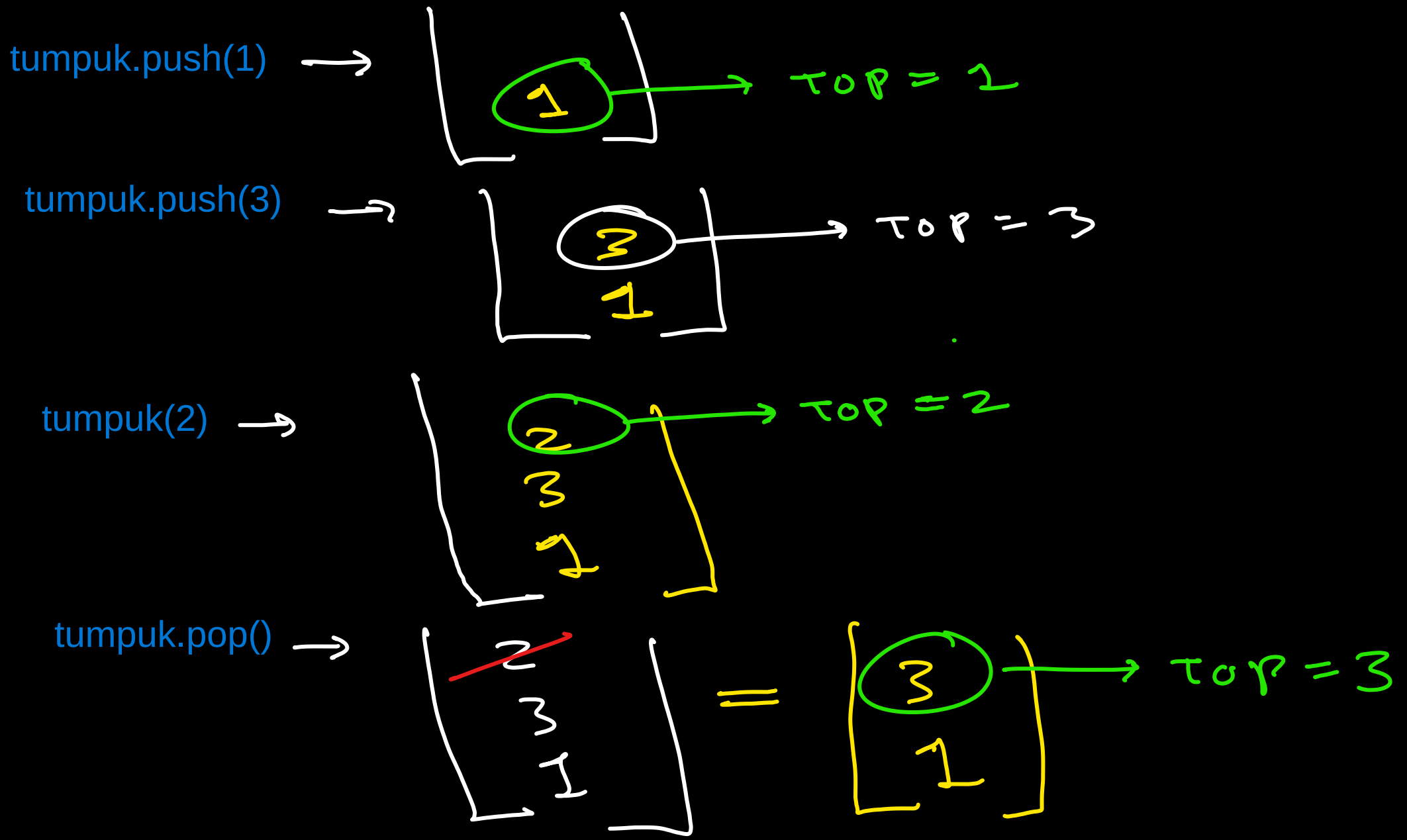
Q.pop()
0 5 4 3 8 2

arr.pop()

1 00 54 3 8

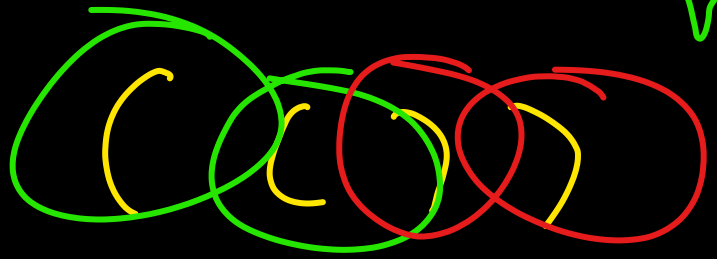
arr.push(9)

1 00 54 3 8 9

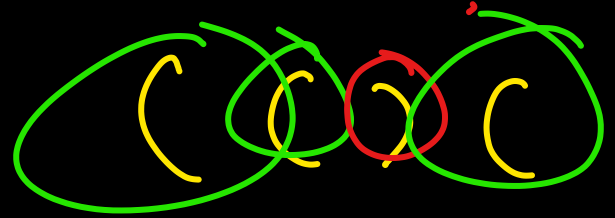

tumpuk.push(1)
1
top = 1
tumpuk.push(3)
3
1
top = 3
tumpuk(2)
2
3
1
top = 2
tumpuk.pop()
2
3
1
=
3
1
top = 3

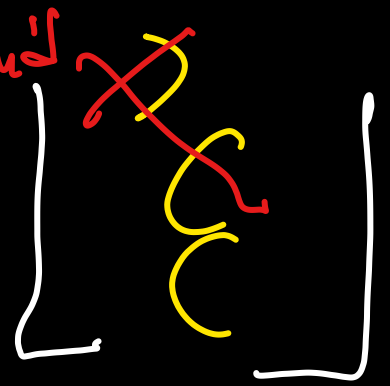
(( )) = valid

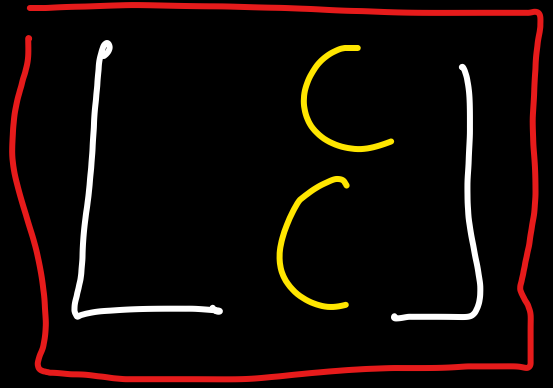
(( ) (= invalid



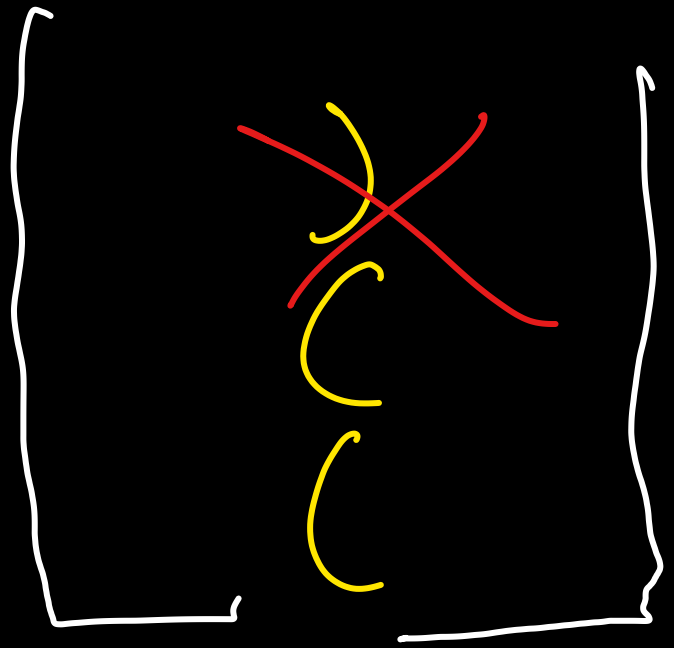
- Kalau ketemu tanda kurung buka masukin ke stack
- Kalau ketemu tanda kurung tutup hapus elemennya





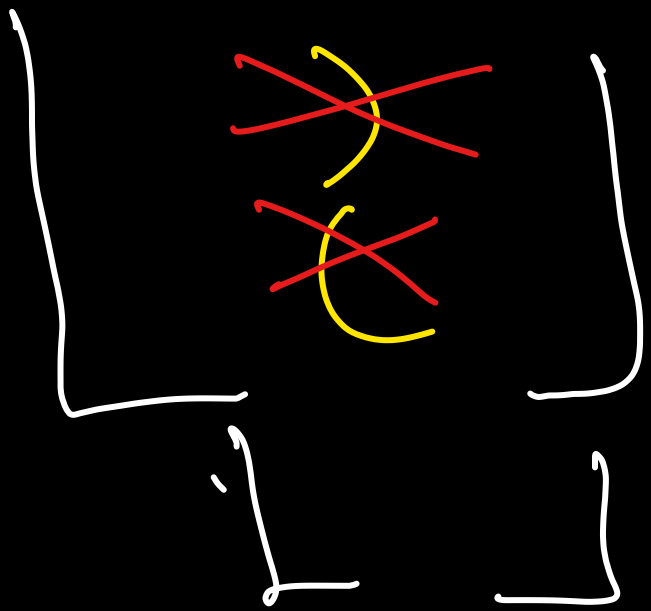


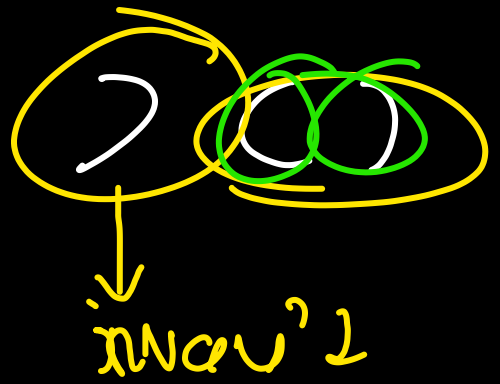
Menyisakan tanda lainnya buka

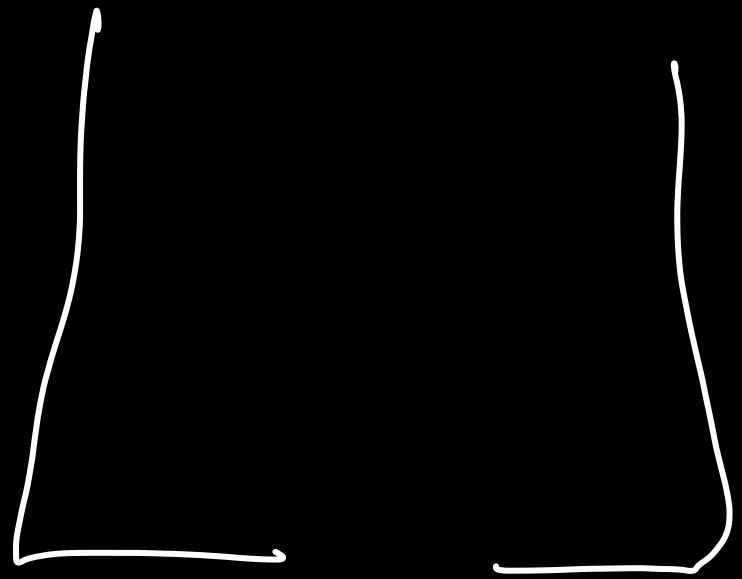


JIka di akhir stack itu kosong = valid

Jika di akhir tidak kosong = invalid




ΔNaν'2



Osnp

Time limit : 2 s

Memory : 256 Mb

C / C++

O(N) → N kai operasi

$N = 10^8$ → 1 detik

$N = 10^{16}$ → 2 detik

T

$N_i \rightarrow N_i!$

$f(n) = n \times f(n-1)$

$f(0) = 1$

T

N_1

N_2

N_3

...

N_7

$f(n) = n \times f(n-1)$ → Kompleksitas $O(N)$

$f(n) = 1 \times f(n-1)$

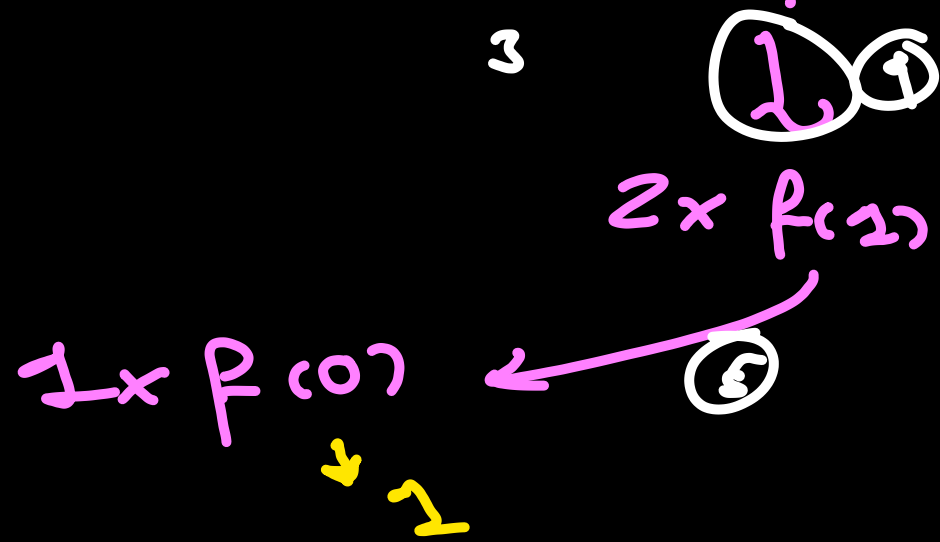
$= 1 \times (n-1) \times f(n-2)$

$= 1 \times (n-1) \times (n-2) \times f(n-3) \times \ldots$

$f(5)$ → 1 : $5 \times f(4)$ → 2 : $4 \times f(3)$ → 3 : $3 \times f(2)$

$f(5)$ = 5 operasi aritmatika

$f(6)$ = 6 operasi aritmatika



Fcm = kompleksitas O(N)

T

Setiap kasus uji ke i → O(N)

O(N) 1
O(N) 2
O(N) 3
- - -
O(N) T

O(N x T) $10^8 = 1s$

$N_{max} = 10^3$
$T_{max} = 10^7$

$O(10^3 \times 10^7) = 10^{10}$
TLE ← $=> 1s$

5

1 → 1! = 1
2 → 2! = 2 × 1
3 → 3! = 3 × 2 × 1
4 → 4! = 4 × 3 × 2 × 1
5 → 5! = 5 × 4 × 3 × 2 × 1

} O(N+1)

5

5 → 5! = 5 × 4 × 3 × 2 × 1
5 → 5! = 5 × 4 × 3 × 2 × 1
5 → 5! = 5 × 4 × 3 × 2 × 1
5 → . . . .
5 — — — —

} O(N)

5

1 → 1! = 1

2 → 2! = 2.1! = 2

3 → 3! = 3.2! = 3.2 = 6

4 → 4! = 4.3! = 4.6 = 24

5 → 5! = 5.4! = 5.24 = 120

Ingat

1! = 1, 2! = 2,

3! = 6

5

5 → 5x4x3x2x1 = 120

5 → 5! = 120

5 →

5 → = 120

5 → = 120

= 120

Ingat

5! = 120

Ingat → Memoization → ≤ 1s

$N_{max} = 10^3$ $T_{max} = 10^7$

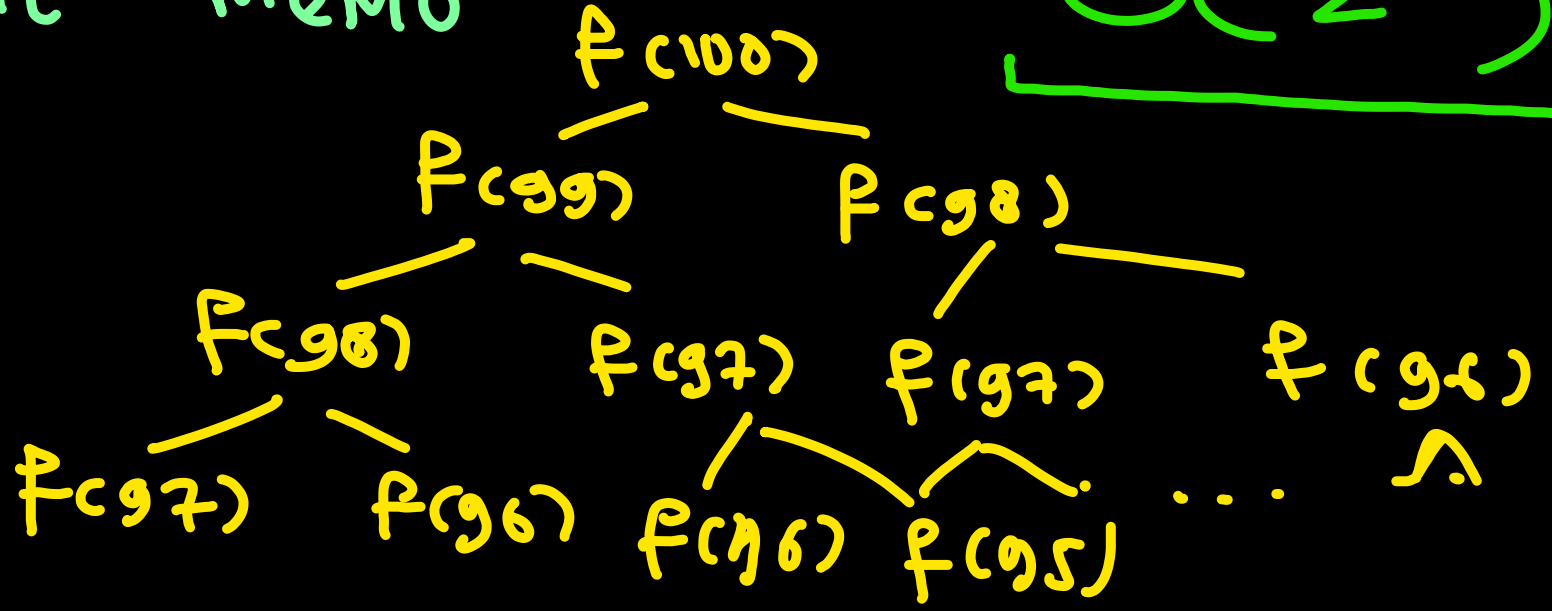
CP → competitive programming

jawaban Tepat & Efisien

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

* Without Memo

Const: $O(2^N)$



Cabanya Banyak Bgt $\rightarrow 2^{100}$

* With Memo

Best complexity : $O(\log_2 N)$

With Memo : O(log N) → Dp

Without memo : O(2^N) → Dnc

tanpa memo — pakai memo

N = 100 → 2^{100} = 2^{100} operasi

$2^{\log 100}$ = 7 operasi

Dp bisa pakai Dnc

Dnc belum tentu Dp

$$2 \; \text{denx} = 10^{16}$$

M x N

start

0 1 2 3

2 2 3 2 3

2+4 4 5 2 4

2+4+7 7 6 1 3

2+4+7+6

(1,1) ⟶ (x,y)

x = 2, y = 3

Bars 2 Room 3

→ 2+4+7+6+1+3

Top down

$dp(x, y)$ = Total item max. yg bisa didapat dari (1,1) → (x, y)

dp(x,y)

0	1	2	3
2	3	2	3
4	5	2	4
7	6	1	3

Sedean mana jalan ke kiri / ke atas

Jalan ke-kiri : (x-1, y)

Jalan ke-atas : (x, y-1)

Menentukan langkah mana yang bisa kita dapatkan sehingga total angka yang diperoleh mendapatkan sebesar mungkin.
Ke Kiri atau ke atas?

Max((x-1, y) , (x, y-1))

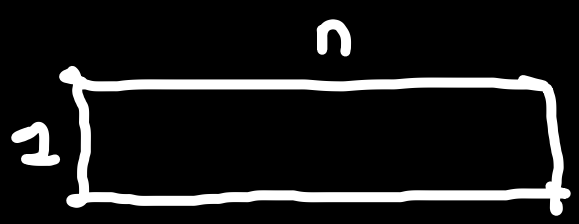
dp[x,y] = max(dp[x-1, y] , dp[x, y-1])
+ item[x-1][y-1]

koordinat
baris, kolom (1, M) , (1, M)

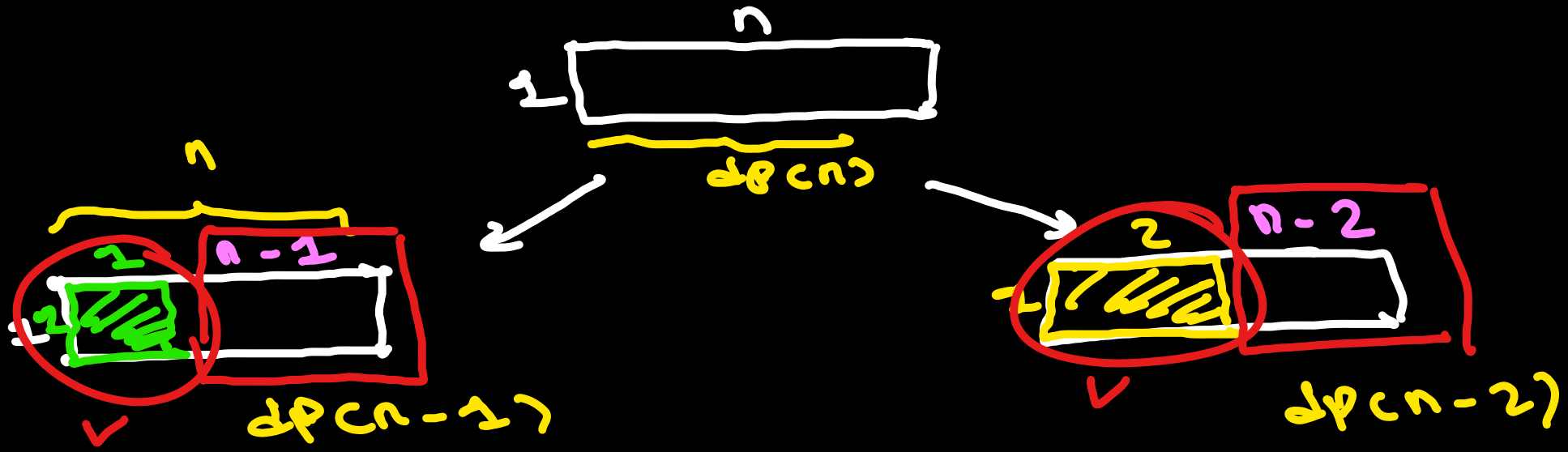
dp(x,y) = max(dp(x - 1, y), dp(x, y -1)) + item[x][y]

dp(1,1) = item[0][0]
dp(1,2) = item[0][0] + item[0][1]
dp(2,1) = item[0][0] + item[1][0]

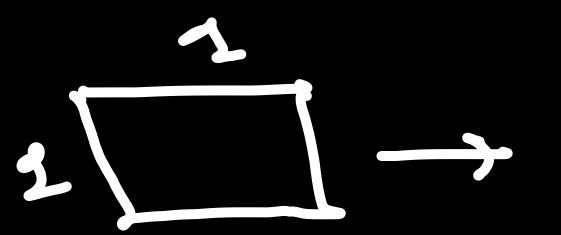
Diberikan lantai berukuran 1 x N, berapa banyak cara memasang ubin berukuran 1 x 1 dan 1 x 2 dengan syarat. Pemasangan tidak boleh tumpang tindih dan semua lantai harus terisi ubin.



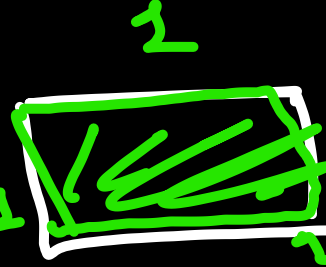
dp(n) = menyatakan banyaknya cara mengisi lantai berukuran 1 x N



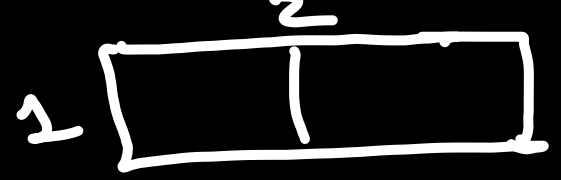
$$dp(n) = dp(n-1) + dp(n-2)$$



→ 1 cara isi →



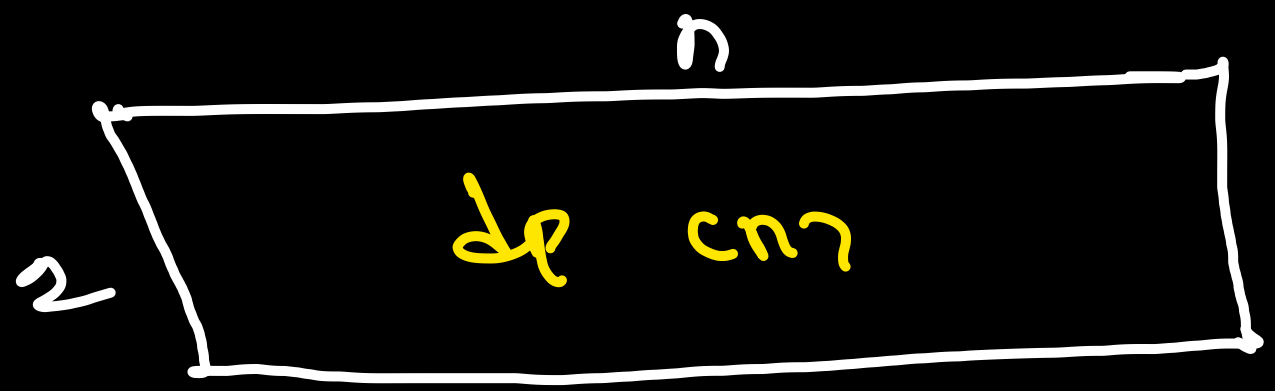
→ dp(1)=1

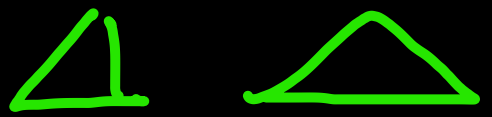


→ 2 cara isi

→ dp(2)=2

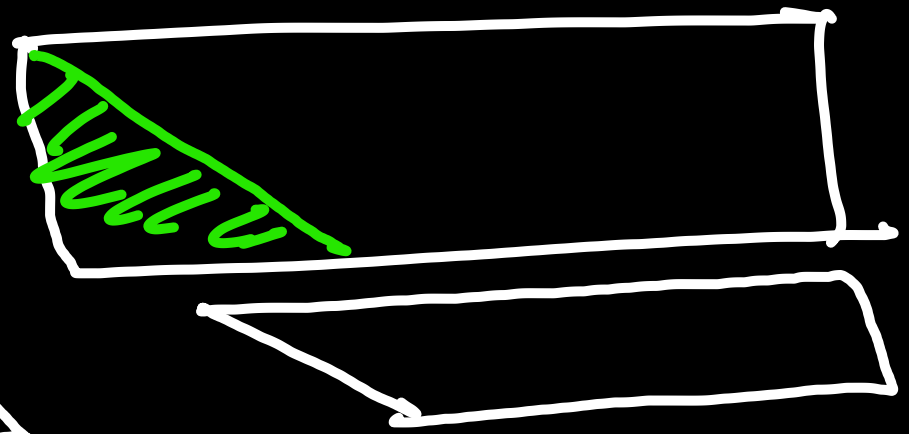
Subcase 1 (N = 2)



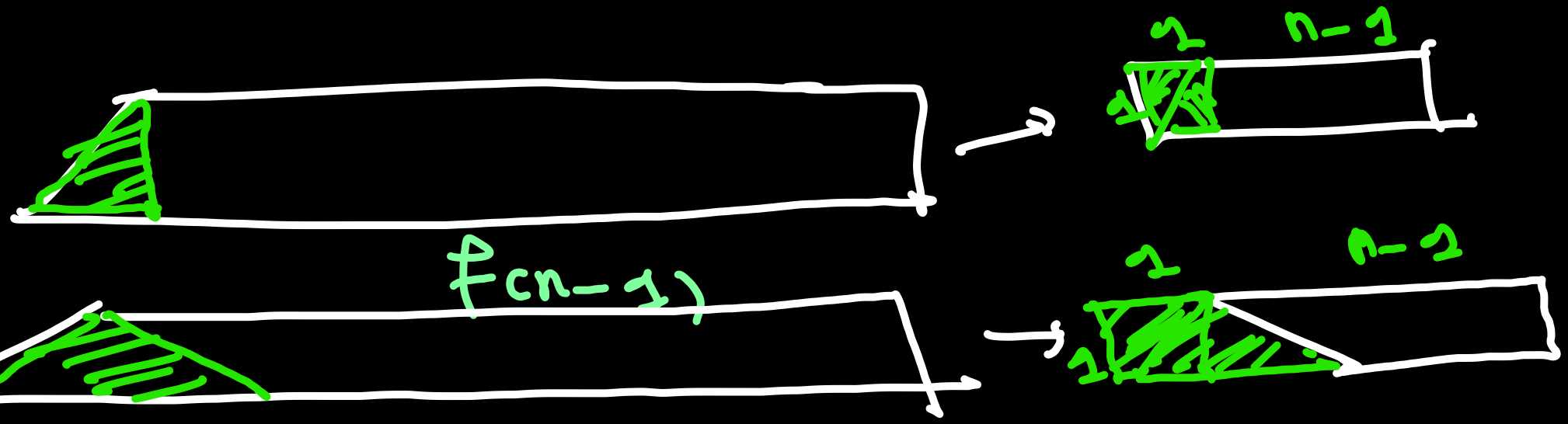


$d(n) = 2g(n)$

$g(n)$



$g(n)$



$f(n) = 2g(n)$

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

$3f(n-1) = 6g(n-1)$

$3f(n-1) = 2g(n)$

$g(n) = f(n-1) + g(n-1)$

$g(n) = 2g(n-1) + g(n-2)$

$g(n) = 3g(n-1)$

$6g(n-1) = 2g(n)$

$$dp(n) = 3\,dp(n-1)$$

$$dp(1) = 2$$

A dan B connected (directed)

adj[A][B] = Wi

- Adj Matriks

adj[u][v] = True

adj[A][B] = True

adj[A][B] = 5

adj[B][A] = 5

Node = {1, 2, 3, 4, 5}

Node sebanyak N

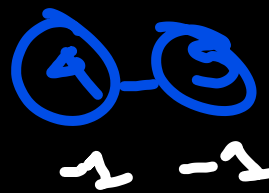
ukuran Matriks ⇒ N * N

adj Matriks ⇒ O(N²)

0 → N * N

1 → (N+1) * (N+1)

adj



cout << adj[3][4]