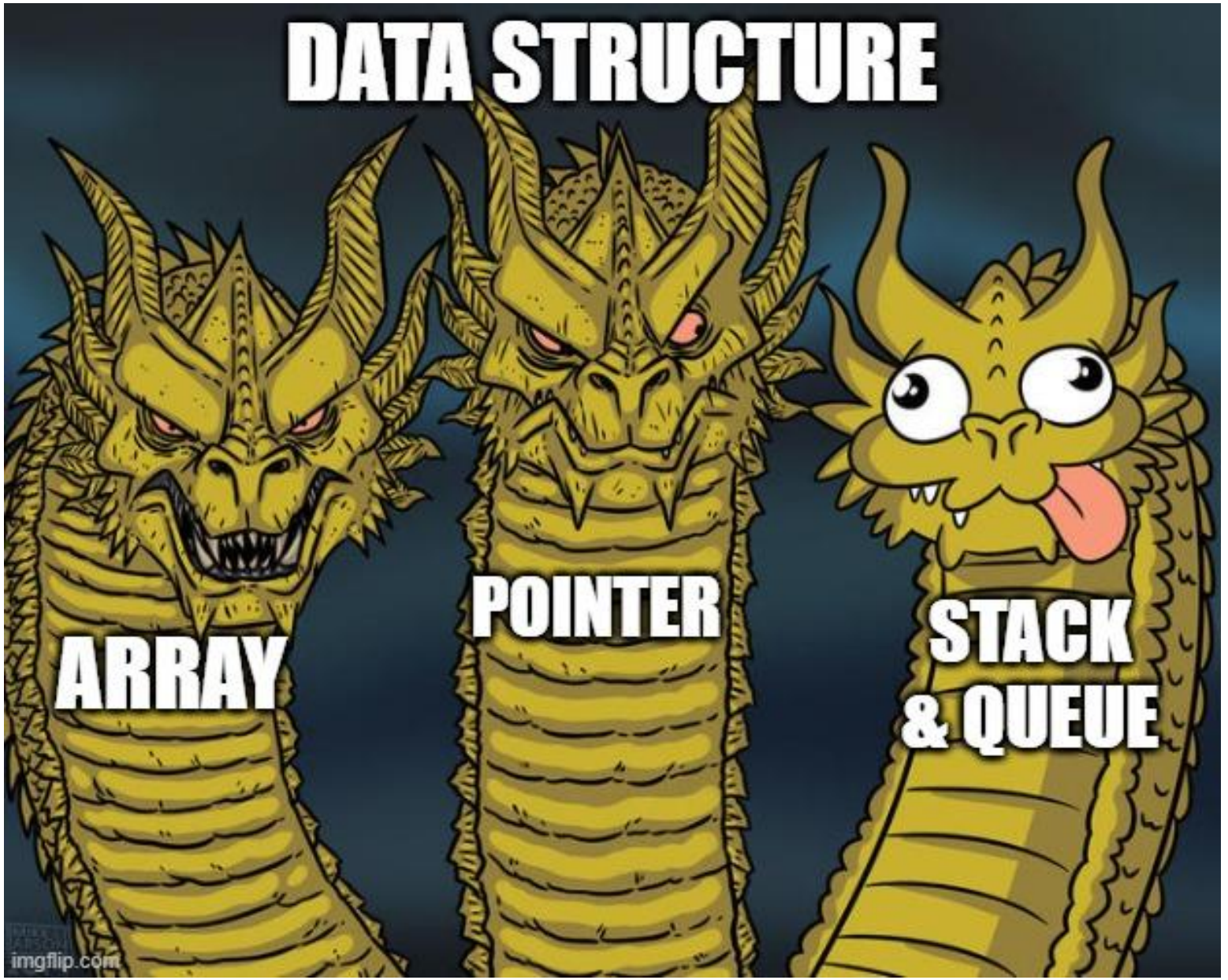
OOP & data struct

10. Stack & Queue (by array)

BY SOMSIN THONGKRAIRAT

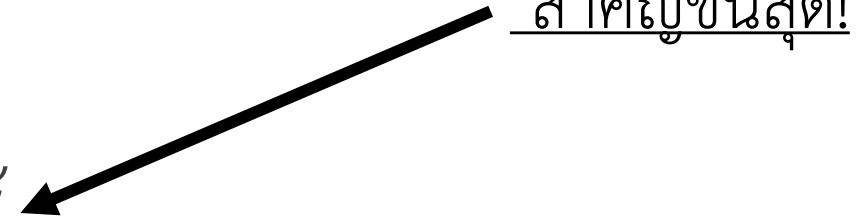

DATA STRUCTURE
ARRAY
POINTER
STACK
& QUEUE
imgflip.com

Stack & Queue

- fundamental data struct
- simple concept
- fundamental operator
- use to build more advance struct

Stack & Queue

- เป็น data struct พื้นฐาน
- เข้าใจง่าย
- มี operator พื้นฐานให้ใช้
- เป็นพื้นฐานเพื่อใช้เพื่อสร้าง data struct ขั้นสูงต่อไป



Queue

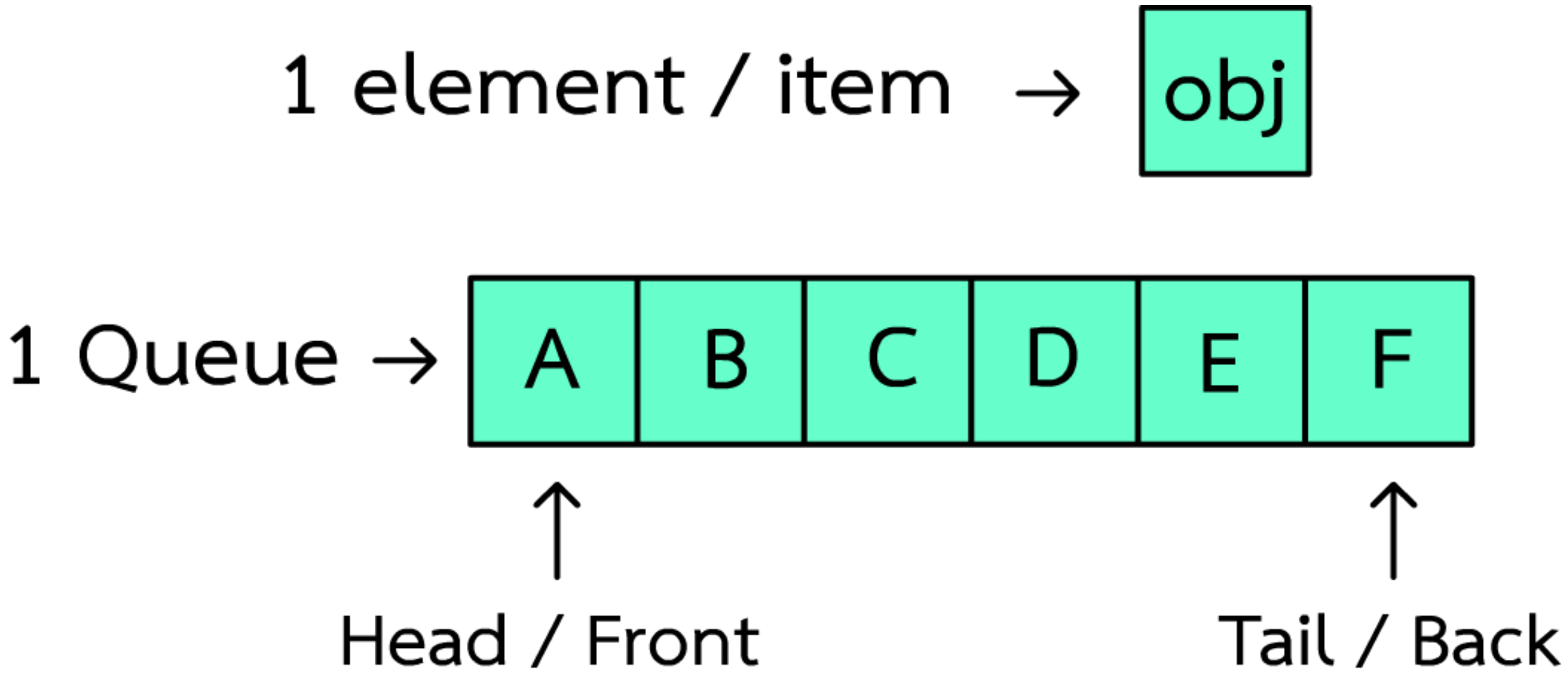
- linear data structure / เป็น data structure แบบ linear
- AKA. FIFO (first in first out)
- open both front and back of structure
- เข้าถึงได้ทั้งหน้าและหลังของ structure

void Enqueue(item) & item Dequeue()

Queue

Queue Data Structure

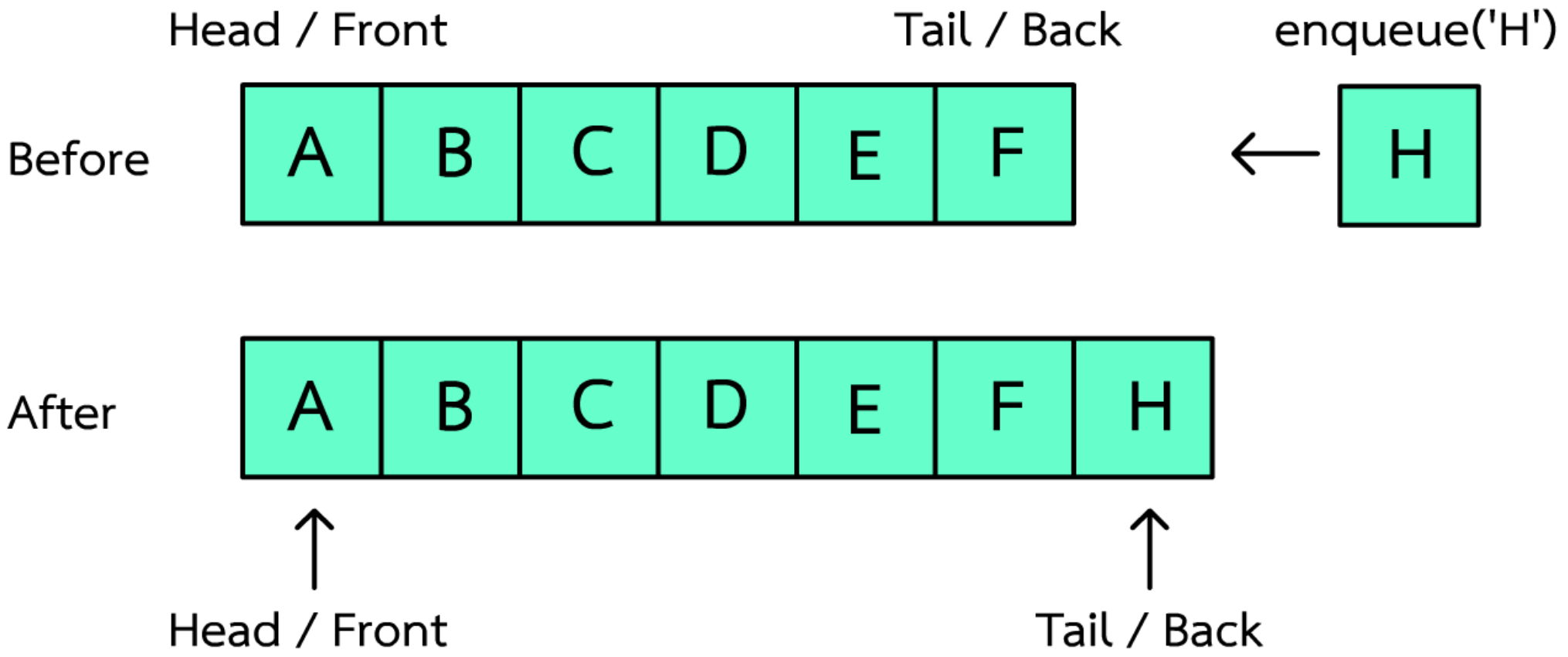
https://www.geeksforgeeks.org/queue-data-structure/


1 element / item →
obj
1 Queue →
A
B
C
D
E
F
Head / Front
Tail / Back

Enqueue (H)

- add item into queue after tail
- เพิ่ม item เข้าไปใน queue ต่อจากหางแถว
- return bool -> success or not

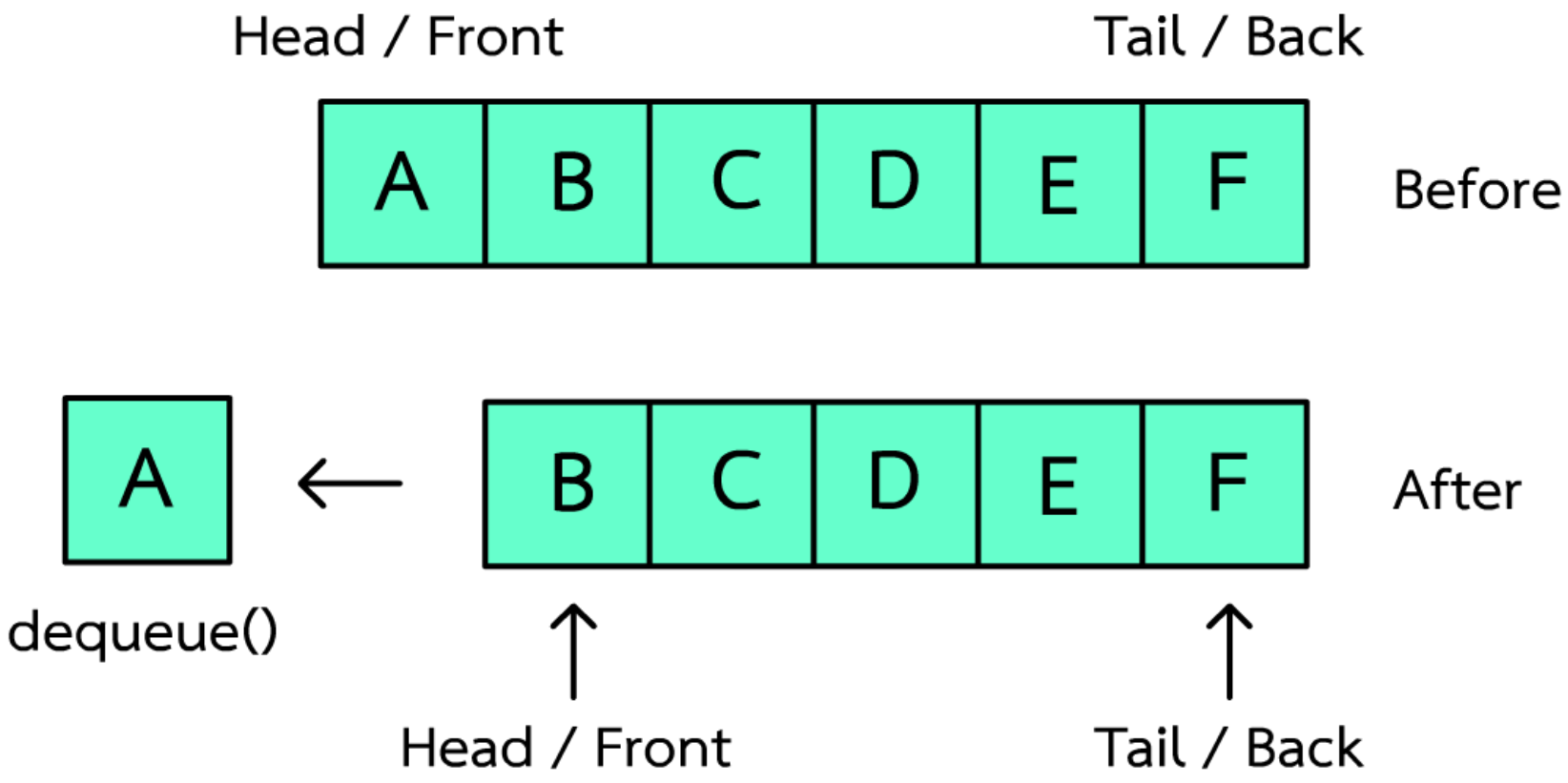
Enqueue (item)



Dnqueue ()

- take head item out of queue
- เอา item ที่อยู่หัวแถวออกมาก
- return item

dequeue ()

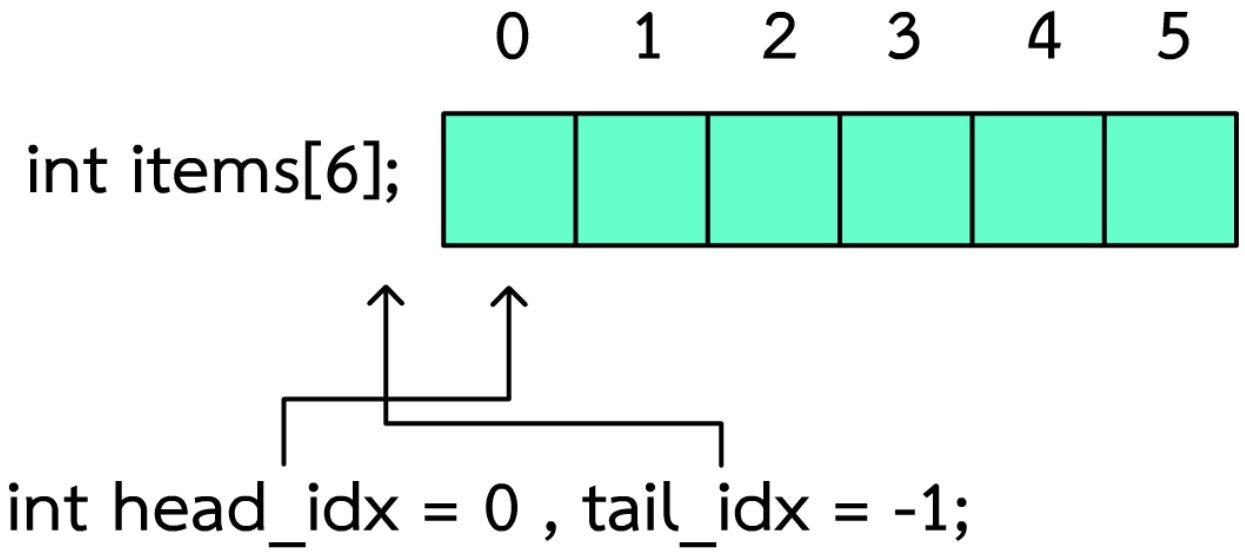


Return C

Implement by array

- create array of item (such as array of int, char, float)

- create index variable (int) to point head and tail



Implemented by array

```
void enqueue(int n){
    tail_idx++;
    items[tail_idx] = n;
}


int dequeue(){
    int return_item = items[head_idx];
    head_idx++;
    return return_item;
}
```

initial

```
Queue(int capacity){
    head_idx = 0;
    tail_idx = -1;
    items = new int[capacity];
}
```

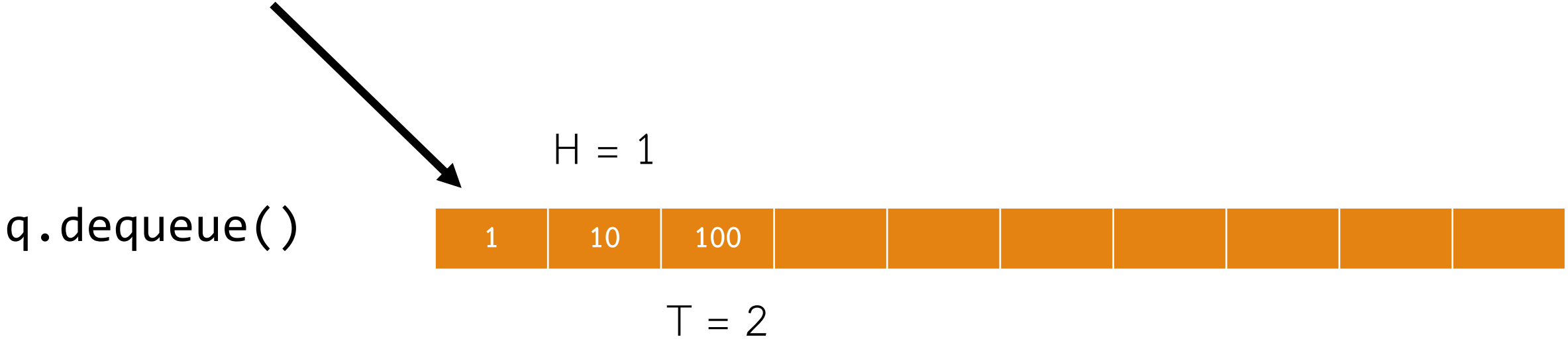
```
head_idx -> 0
tail_idx -> -1
```

```
q.enqueue(1);
head_idx -> 0
tail_idx -> 0
```


```
q.enqueue(10);
head_idx -> 0
tail_idx -> 1
```

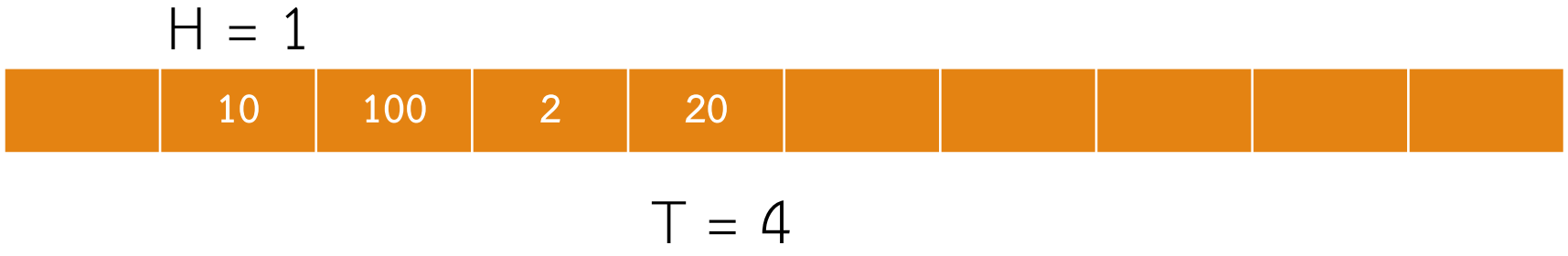

```
q.enqueue(100);
head_idx -> 0
tail_idx -> 2
```

Return

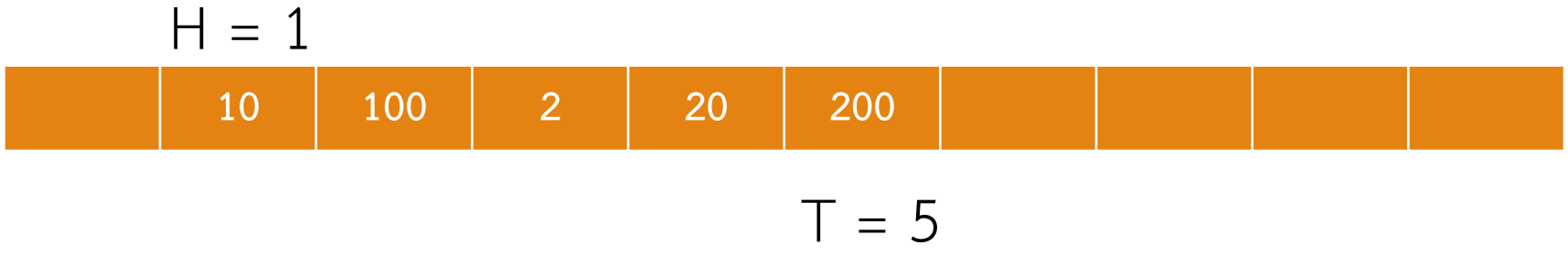


```
q.enqueue(2);
head_idx -> 1
tail_idx -> 3
```


```
q.enqueue(20);
head_idx -> 1
tail_idx -> 4
```




```
q.enqueue(200);
head_idx -> 1
tail_idx -> 5
```



Code :

```
Queue q(10);
q.enqueue(1); q.enqueue(10); q.enqueue(100);
cout << q.dequeue() << endl;
q.enqueue(2); q.enqueue(20); q.enqueue(200);
cout << q.dequeue() << endl;
cout << q.dequeue() << endl;
cout << q.dequeue() << endl;
cout << q.dequeue() << endl;
cout << q.dequeue() << endl;
```

Result :

```
1
10
100
2
20
200
```

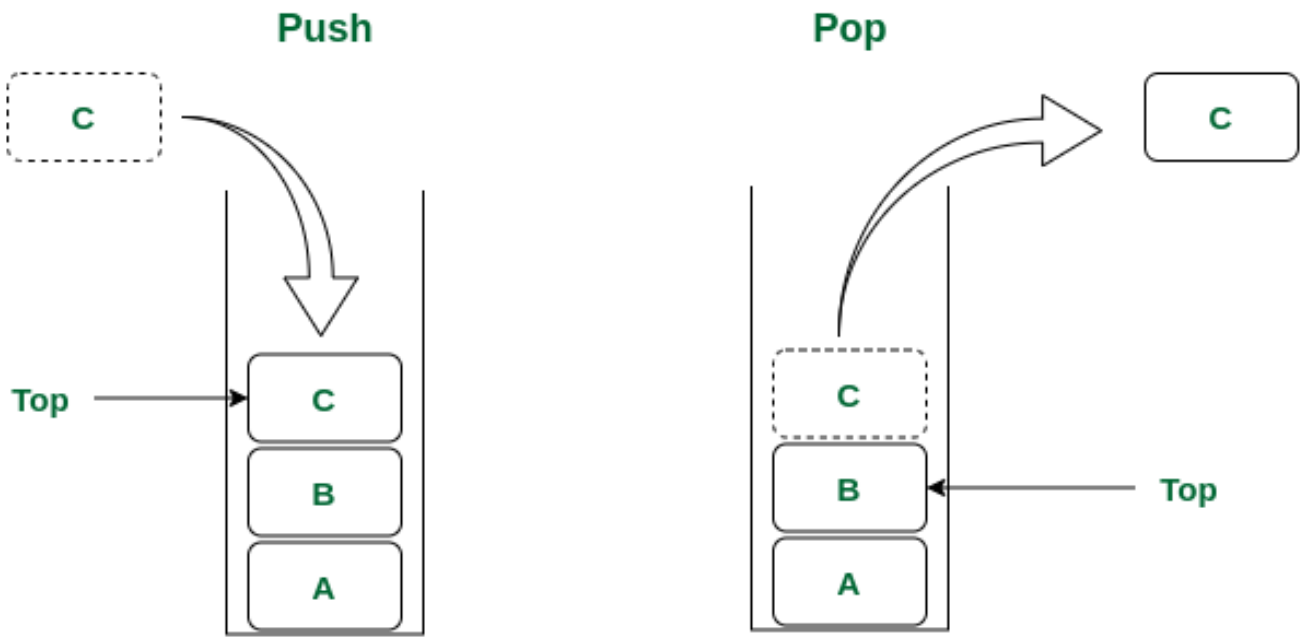
First in First out!

Stack

- linear data structure / เป็น data structure แบบ linear
- AKA. FILO (first in last out) or LIFO (last in first out)
- open just font side
- เข้าถึงได้จากด้านหน้าเท่านั้น

void push(item) & item pop()

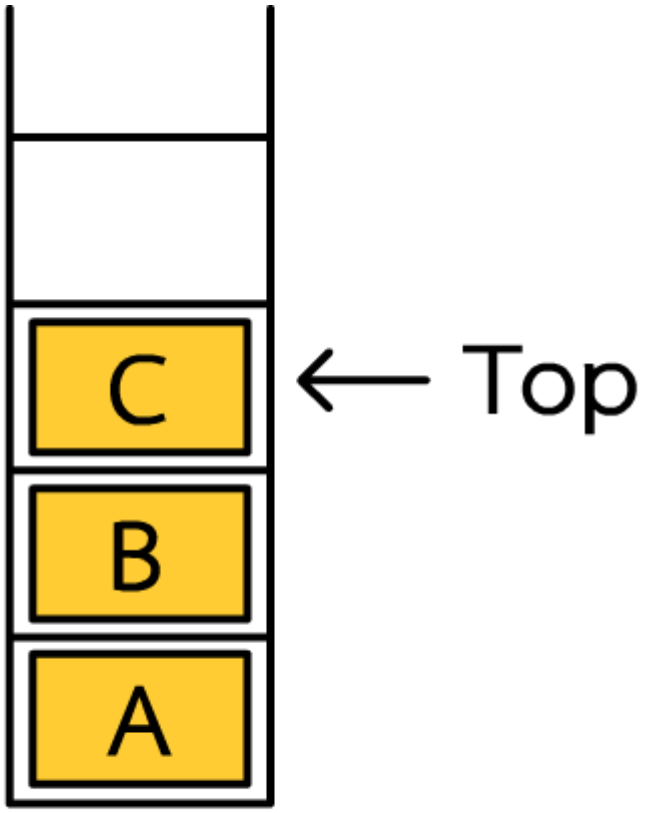
Stack



Stack Data Structure

https://www.geeksforgeeks.org/stack-data-structure/

1 element / item



1 stack

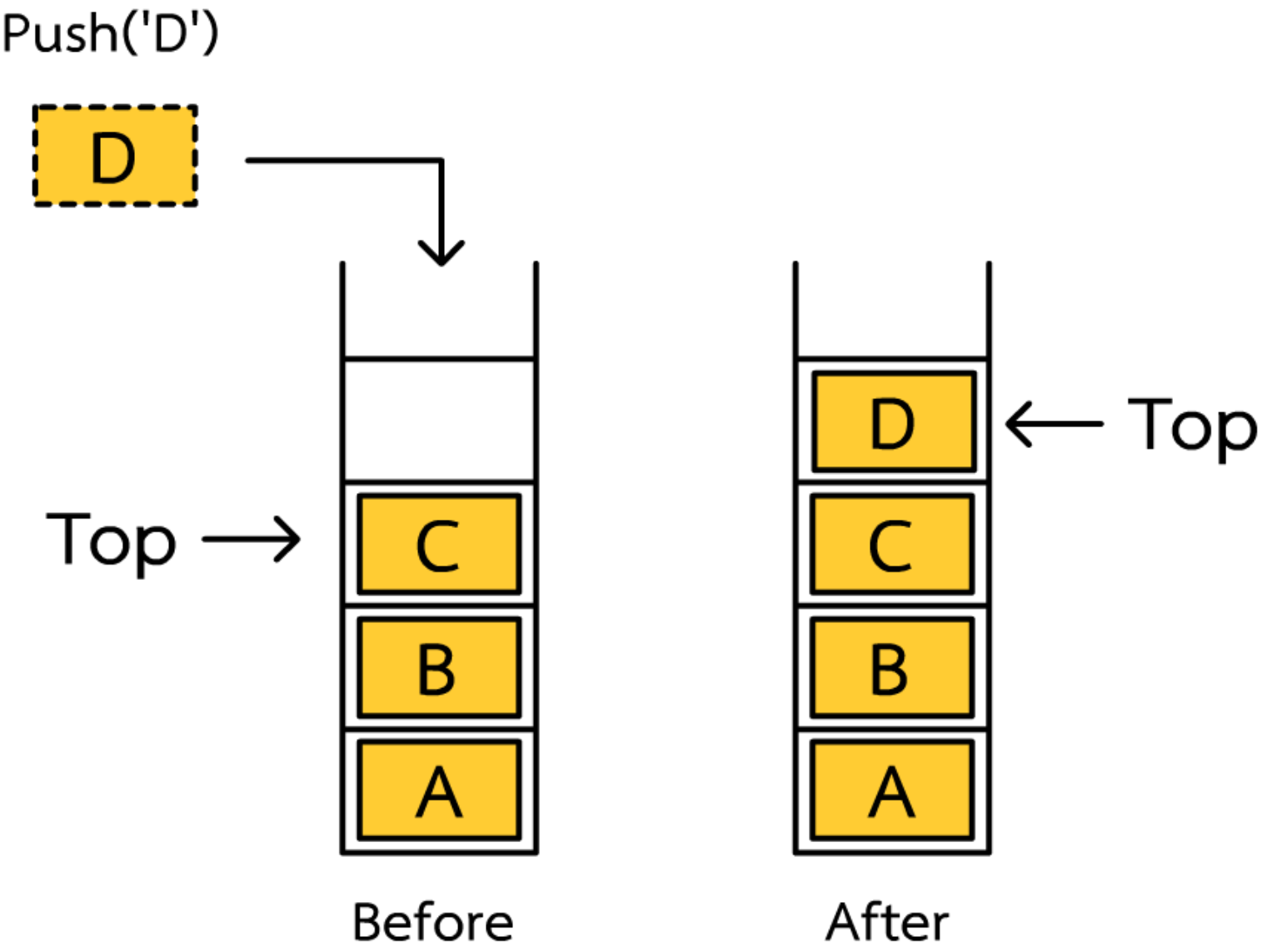
Push (item)

- add item into top of stack
- เพิ่ม item เข้าไปด้านบนสุดของ stack
- return bool -> success or not

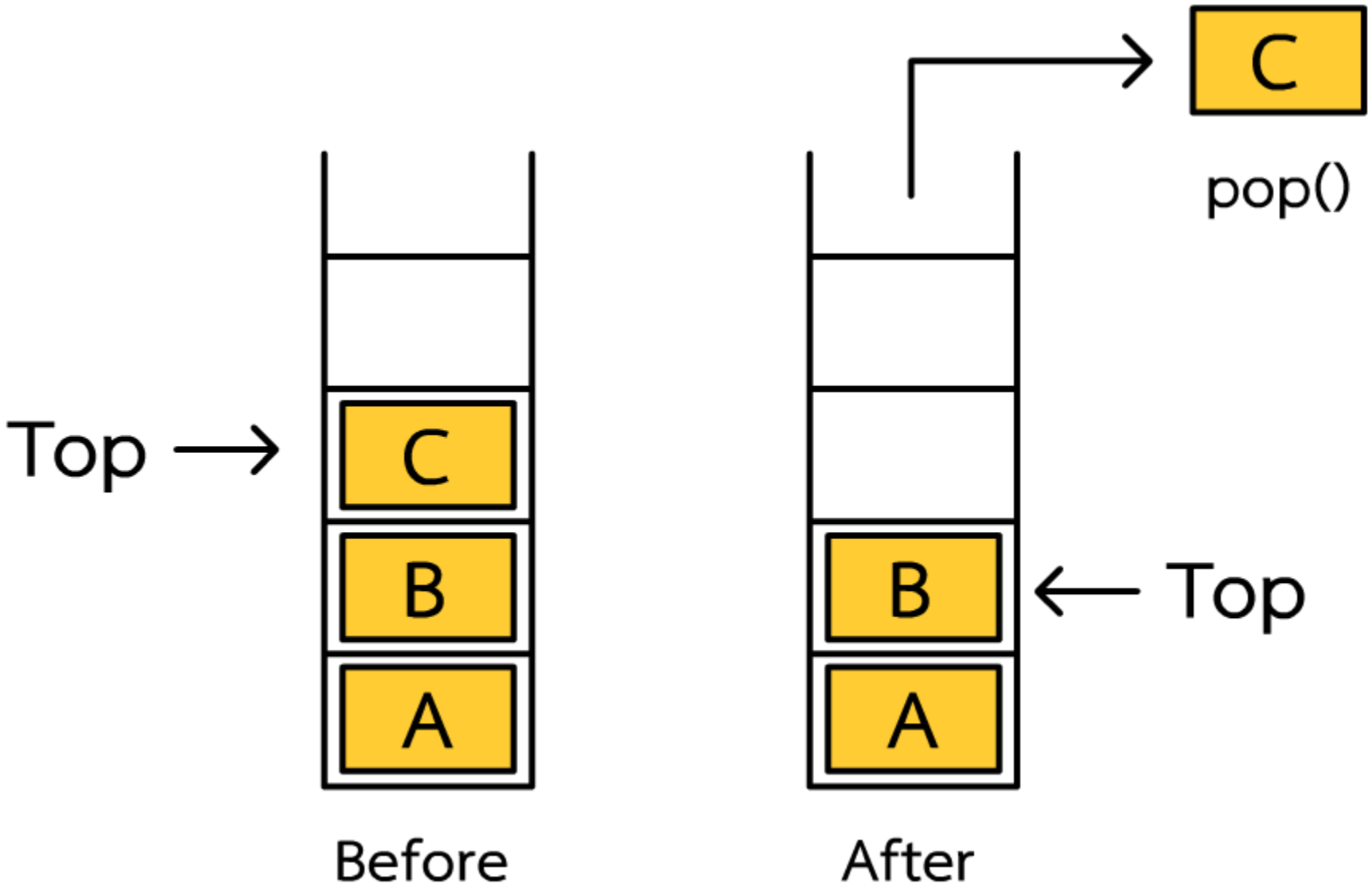
pop ()

- take top item out of stack
- เอา item ที่อยู่บนสุดออกมา
- return item

Push(item)



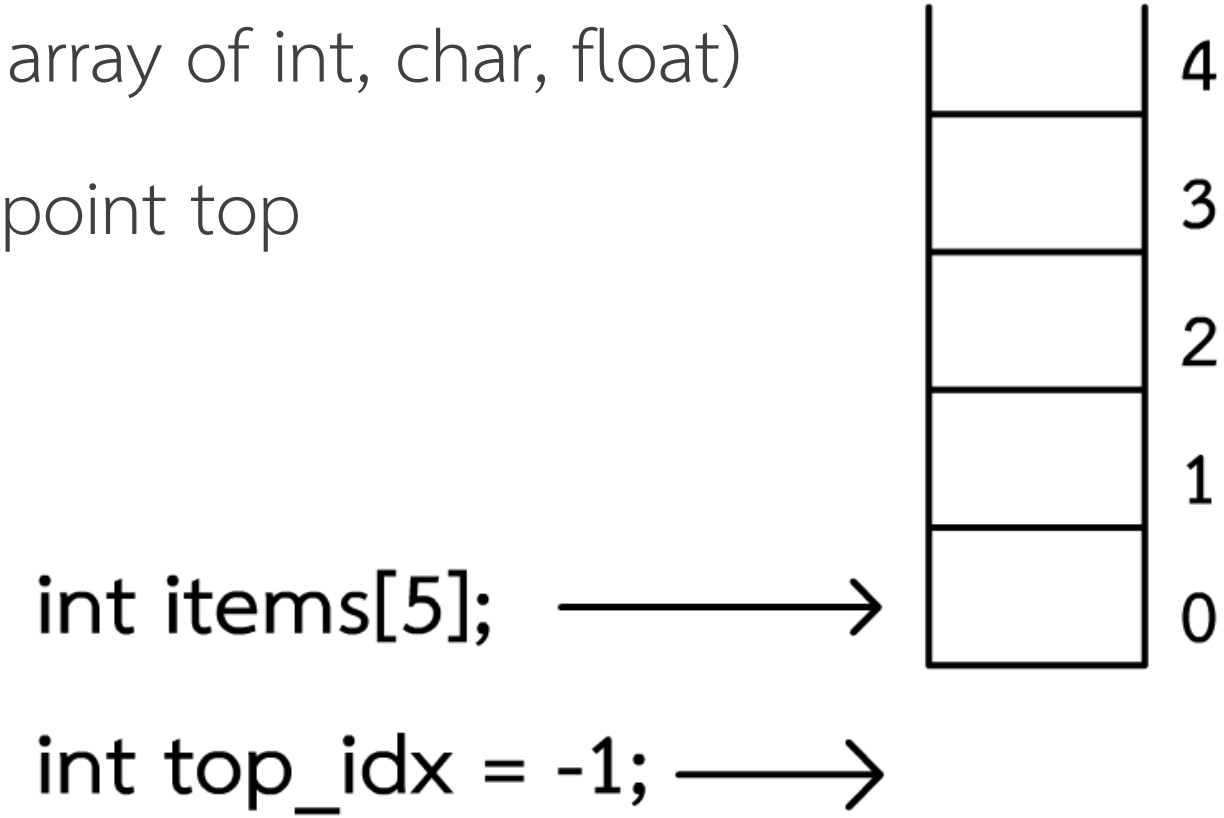
pop()



Return C

Implemented by array

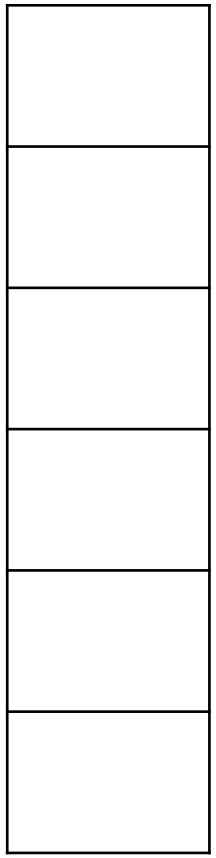
- create array of item (such as array of int, char, float)
- create index variable (int) to point top



Implemented by array

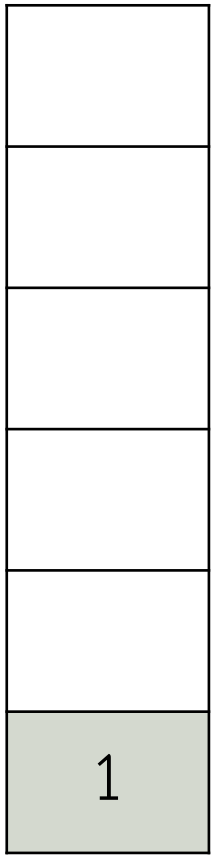
```
void push(int n){
    top_idx++;
    items[top_idx] = n;
}


int pop(){
    int return_item = items[top_idx];
    top_idx--;
    return return_item;
}
```



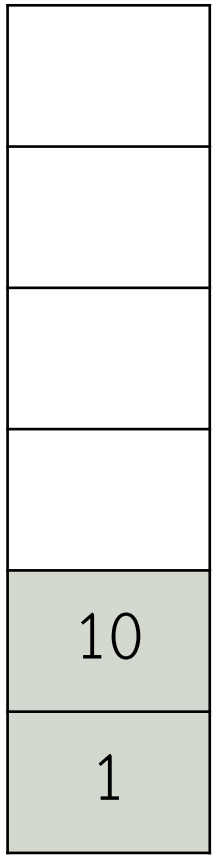
`start`

Top = -1



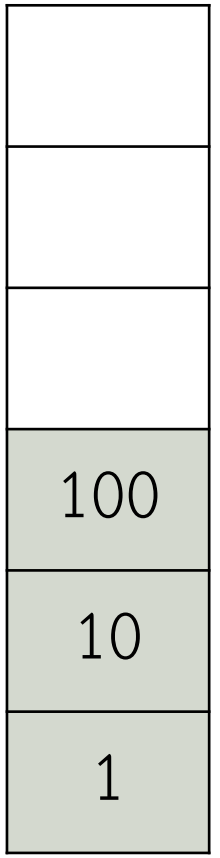
`s.push(1);`

Top = 0



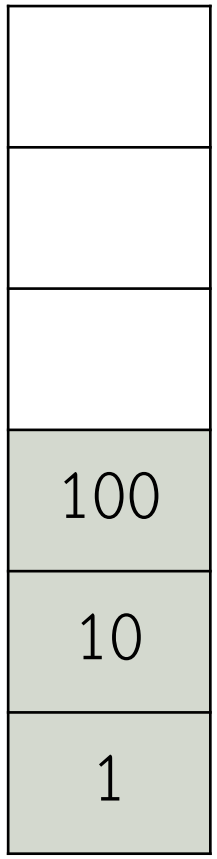
`s.push(10);`

Top = 1



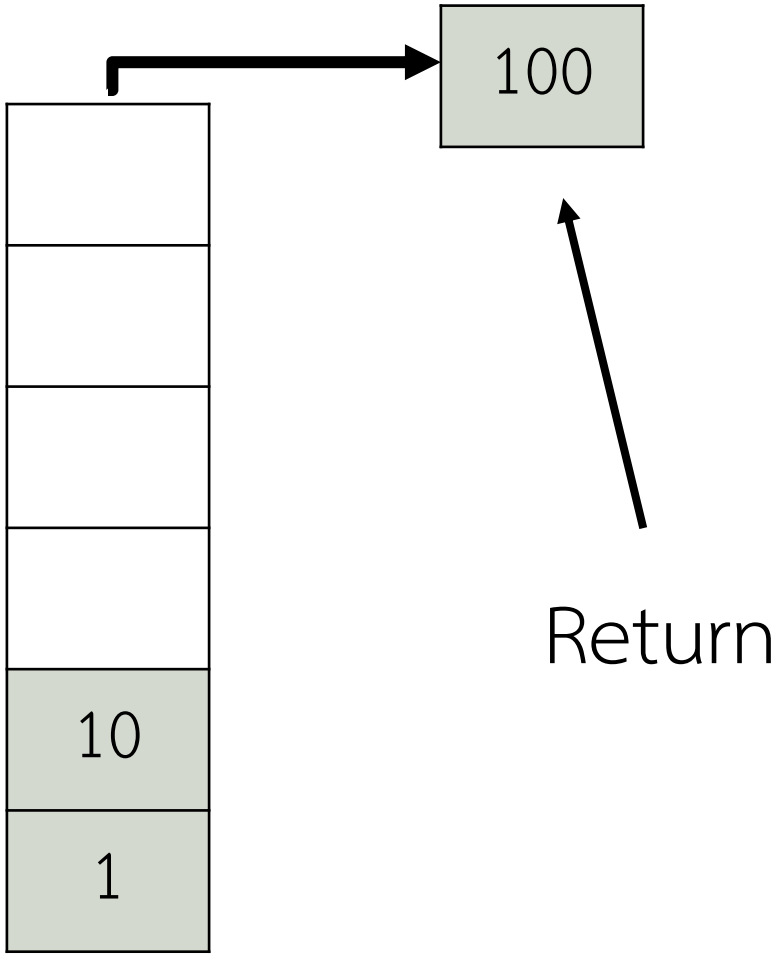
`s.push(100);`

Top = 2



`resume`

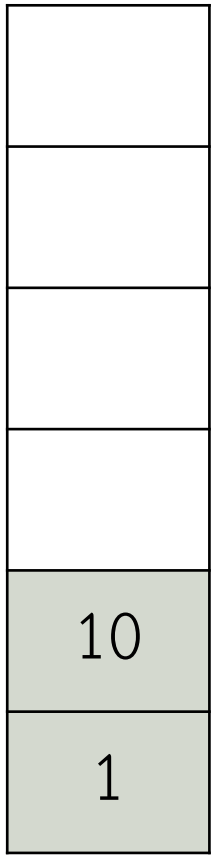
Top = 2



`s.pop()`

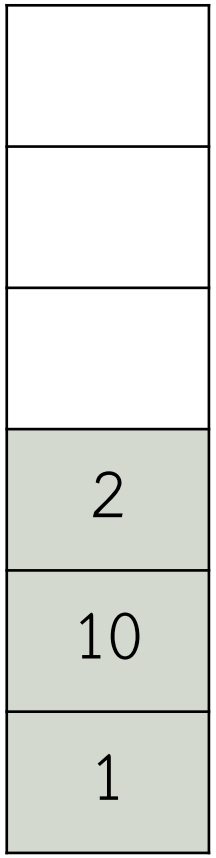
Top = 1

Return



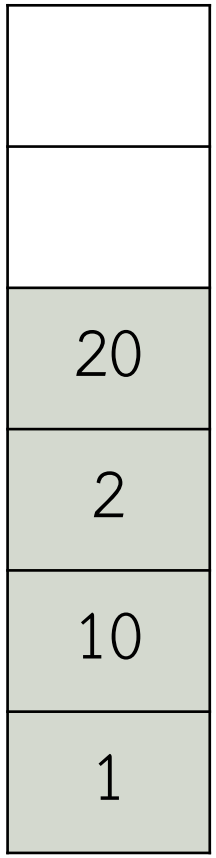
`resume`

Top = 1



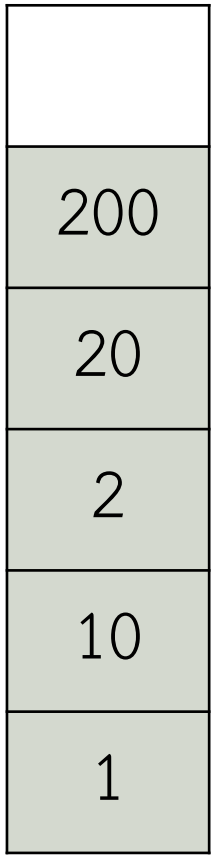
`s.push(2);`

Top = 2



`s.push(20);`

Top = 3



`s.push(200);`

Top = 4

Code :

```
Stack s(10);
s.push(1); s.push(10); s.push(100);
cout << s.pop() << endl;
s.push(2); s.push(20); s.push(200);
cout << s.pop() << endl;
cout << s.pop() << endl;
cout << s.pop() << endl;
cout << s.pop() << endl;
cout << s.pop() << endl;
```

Result :

```
100
200
20
2
10
1
```

Quiz

Exceed capacity?

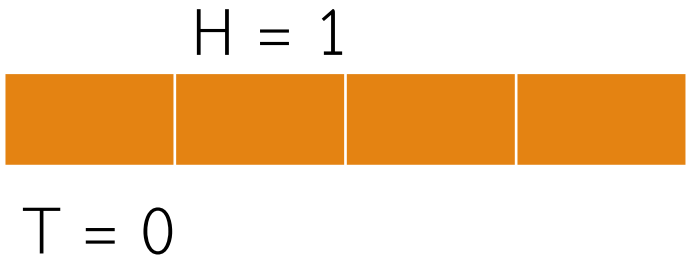
- if we pop or dequeue while it don’t have any data, what will happen?
- ถ้า pop หรือ dequeue ในขณะที่ไม่มีข้อมูล จะเกิดอะไรขึ้น?
- if we push or enqueue item more than struct capacity, what will happen?
- ถ้า push หรือ enqueue item มากกว่าความจุของ struct จะเกิดอะไรขึ้น?

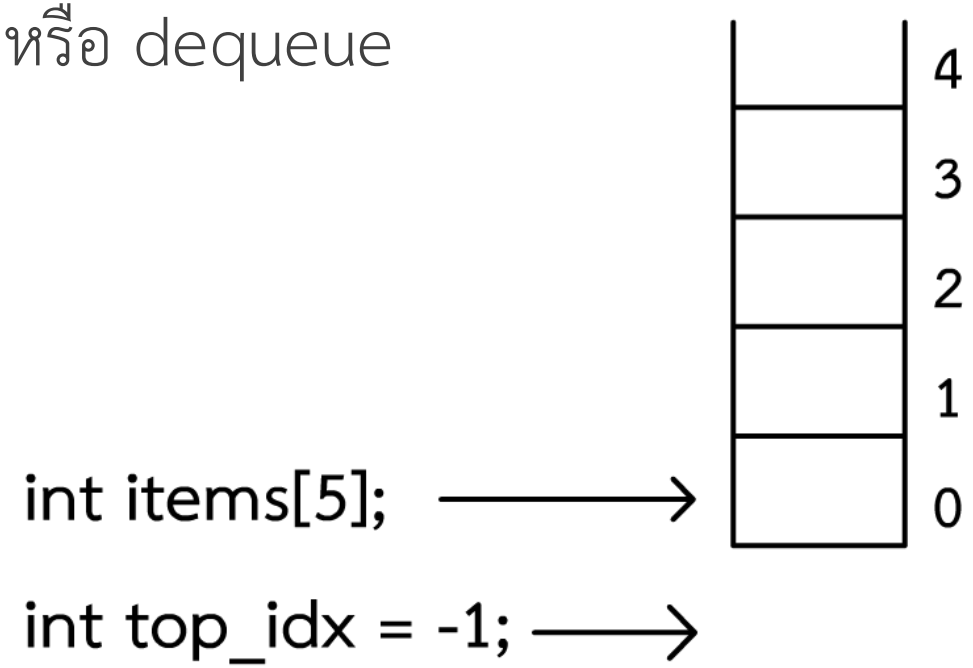
- how to prevent it?
- สามารถป้องกันได้อย่างไร?

is_empty()

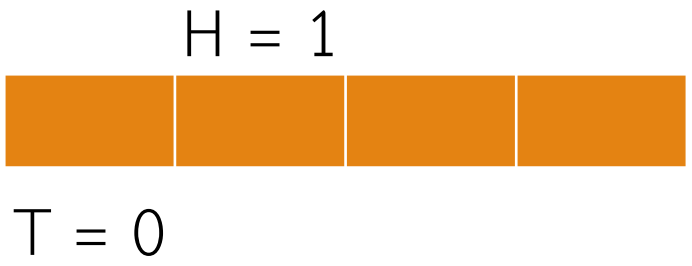
- check if item empty before pop or dequeue

- เช็คว่ามี item อยู่หรือไม่ก่อนการ pop หรือ dequeue



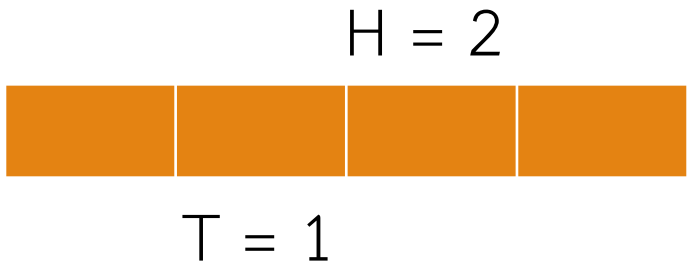


is queue empty




```
bool is_empty(){
    return tail_idx < head_idx;
}
```

OR



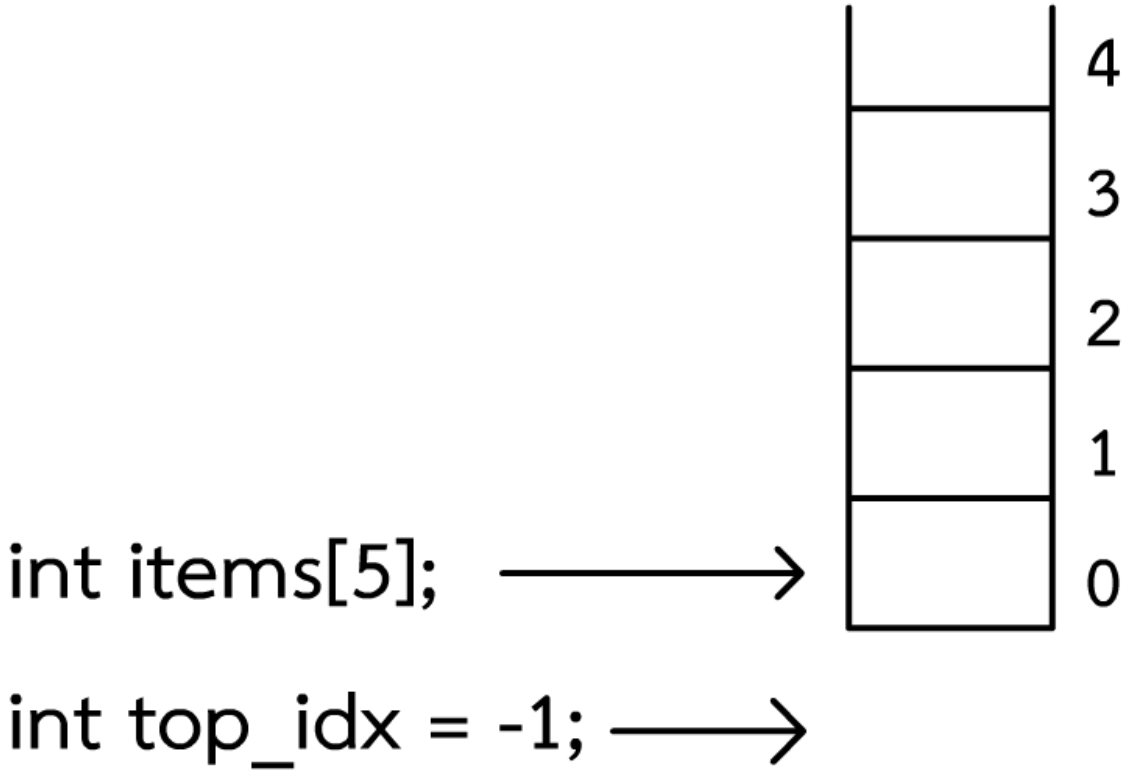

```
bool is_empty(){
    if(tail_idx < head_idx) return true;
    return false;
}
```

is stack empty

```
bool is_empty(){
    return top_idx < 0;
}
```

OR

```
bool is_empty(){
    if(top_idx < 0) return true;
    return false;
}
```



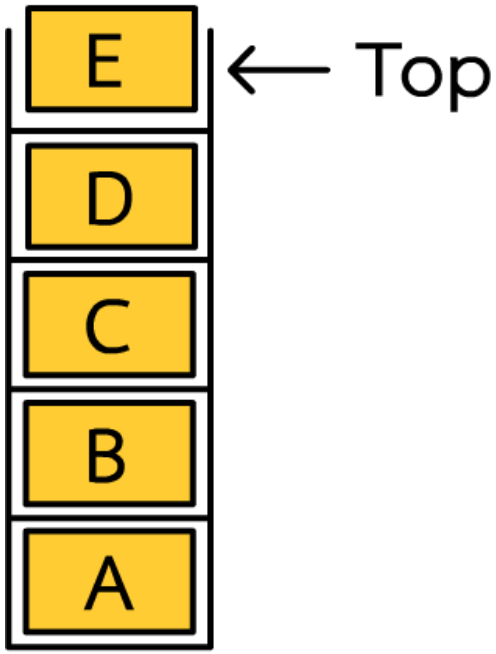
is_full()

- check if item full before push or enqueue

- เช็คว่ามี struct เต็มหรือไม่ก่อนการ push หรือ enqueue

H = 0

T = 4

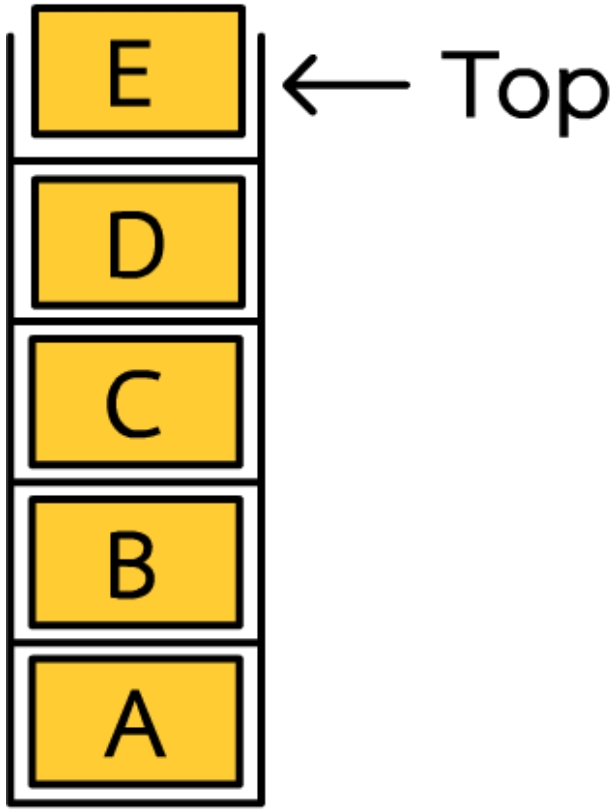


Is stack full

```
bool is_full(){
    return (top_idx + 1) >= capacity;
}
```

OR

```
bool is_full(){
    if((top_idx + 1) >= capacity) return true;
    return false;
}
```



Is queue full

```
bool is_full(){
    return (tail_idx) >= capacity;
}
```

OR

```
bool is_full(){
    if((tail_idx) >= capacity) return true;
    return false;
}
```

Is queue full

- solve ? -> NO!

- ปัญหาถูกแก้ไขหรือไม่ –> ไม่!

- queue still full no mater how many time you dequeue

- queue จะยังเต็มอยู่เสมอไม่ว่าจะ dequeue ไปมากขนาดไหน

Increase capacity?

H = 0

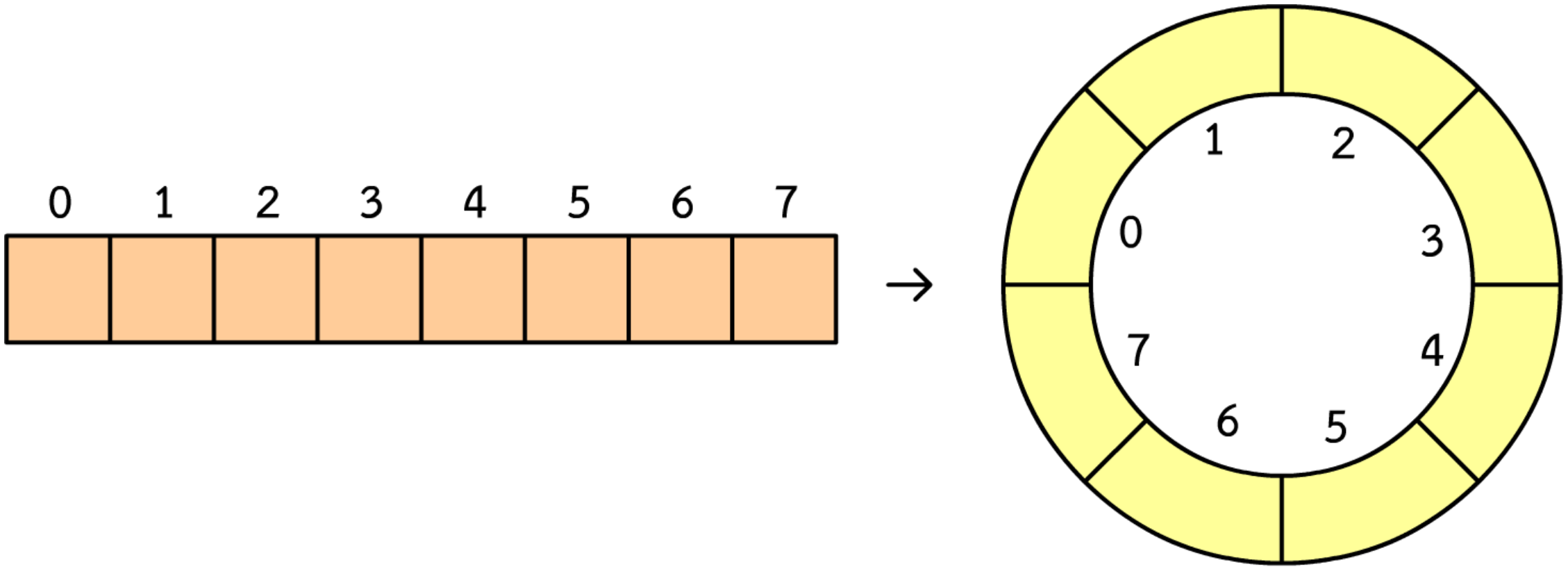
T = 4

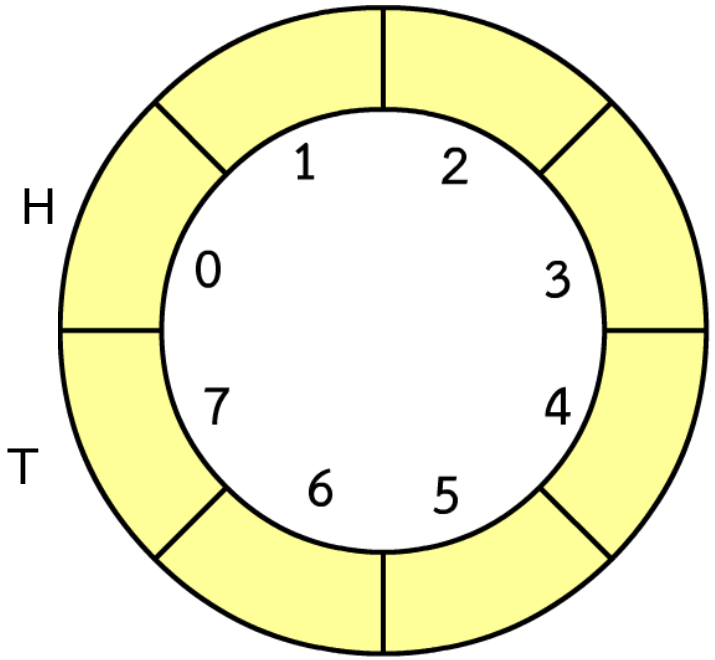
H = 0

T = 4

Solved but not sustain / แก้ไขได้แต่ไม่ยั่งยืน

Circular queue

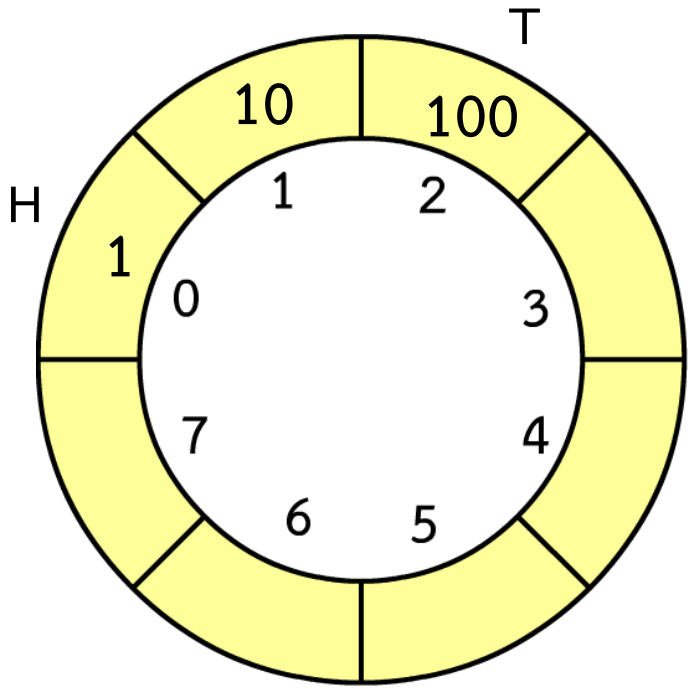




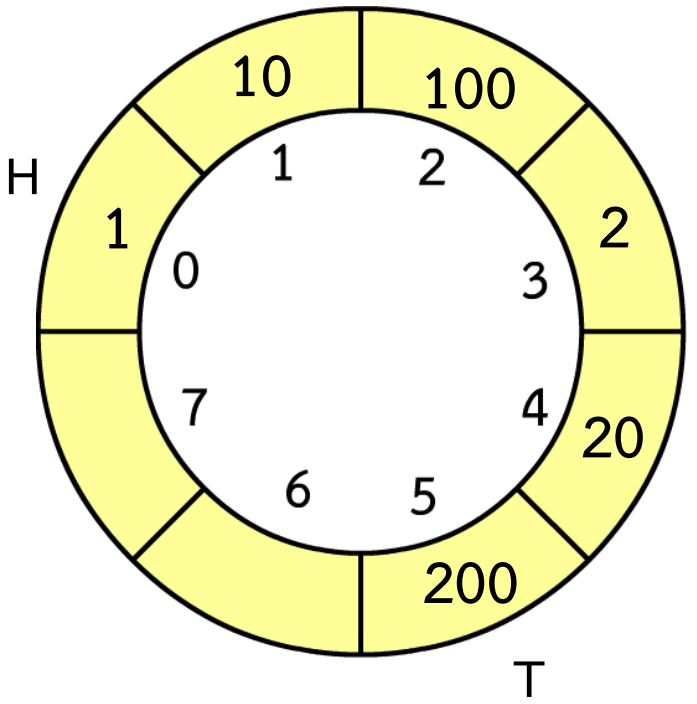
Start

Head = 0

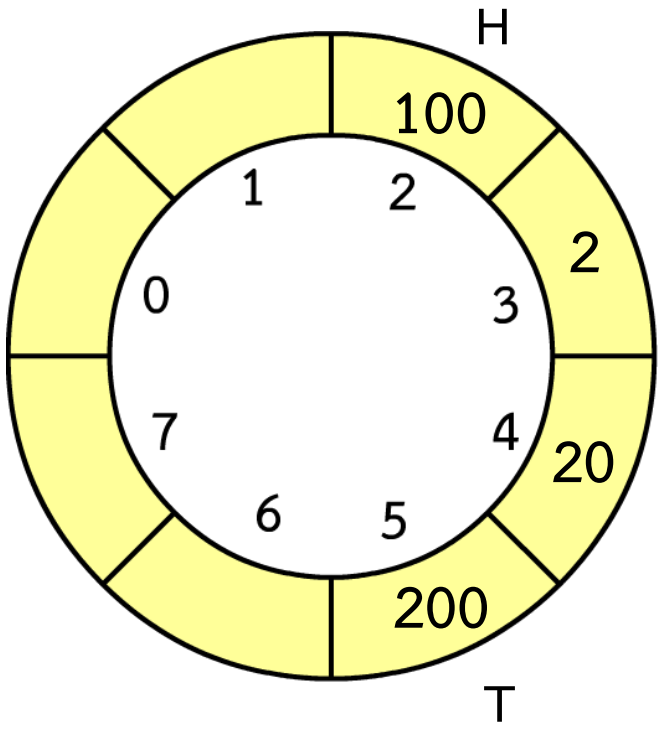
Tail = 7



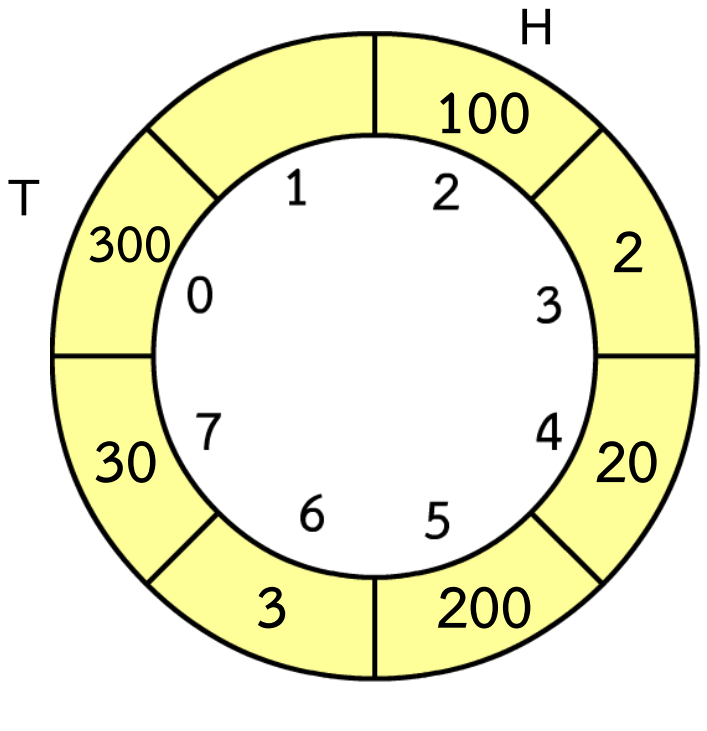

```
q.enqueue(1);
q.enqueue(10);
q.enqueue(100);
   Head = 0
   Tail = 2
```




```
q.enqueue(2);
q.enqueue(20);
q.enqueue(200);
   Head = 0
   Tail = 5
```




```
q.dequeue(); -> 1
q.dequeue(); -> 10
     Head = 2
     Tail = 5
```




```
q.enqueue(3);
q.enqueue(30);
q.enqueue(300);
    Head = 2
    Tail = 0
```

Solved!

Big O for basic operation

Queue

- enqueue , dequeue -> O(1)

Stack

- push , pop -> O(1)

Big O for search

Search ?

O(n)

When n is capacity of structure

เมื่อ n คือความจุของ structure

conclude

- queue
- stack
- prevent exceed memory

Lab