VIETNAM AUSTRALIA Vocational College

Các kỹ thuật xử lý, sắp xếp ngăn xếp và hàng đợi

Mentor: Nguyễn Bá Minh Đạo

Nội dung:

1. Tổng quan về cấu trúc dữ liệu ngăn xếp
2. Các kỹ thuật để thao tác với ngăn xếp
3. Tổng quan về cấu trúc dữ liệu hàng đợi
4. Các kỹ thuật để thao tác với hàng đợi
5. So sánh ngăn xếp và hàng đợi

Tổng quan về cấu trúc dữ liệu ngăn xếp

❑ Ngăn xếp (Stack) là gì?

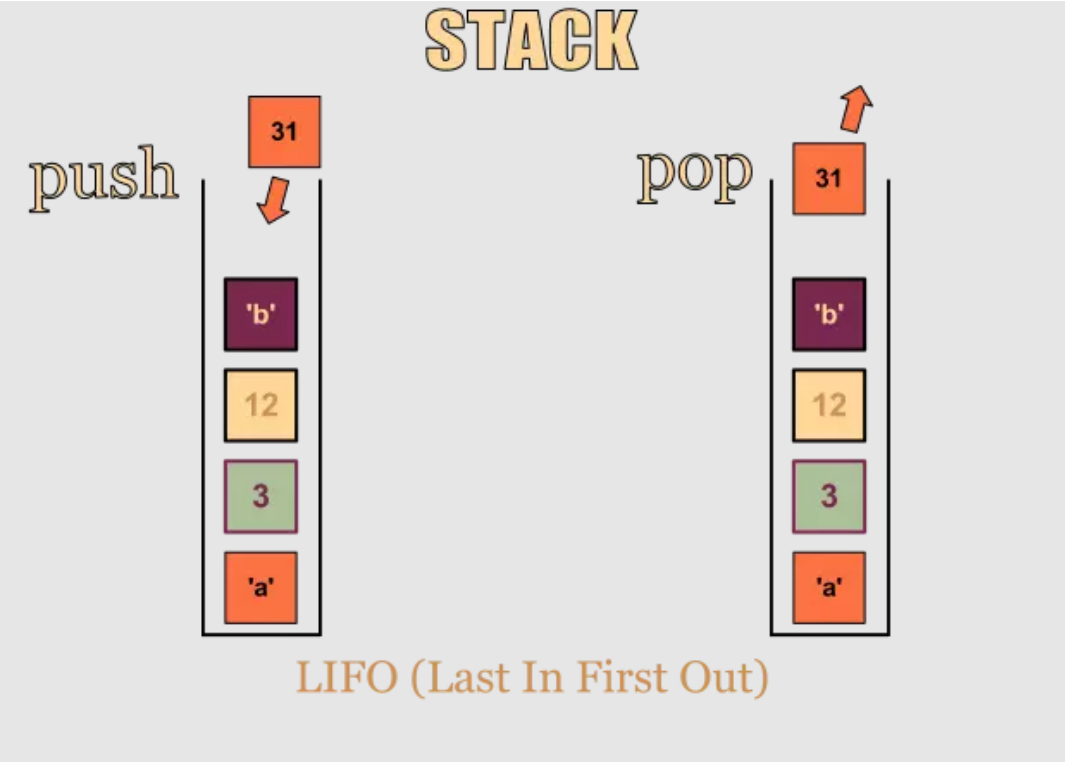
➢ Ngăn xếp (stack) là **một cấu trúc dữ liệu trừu tượng** hoạt động theo nguyên lý "**vào sau ra trước**". (**LIFO**: **L**ast **I**n **F**irst **O**ut)

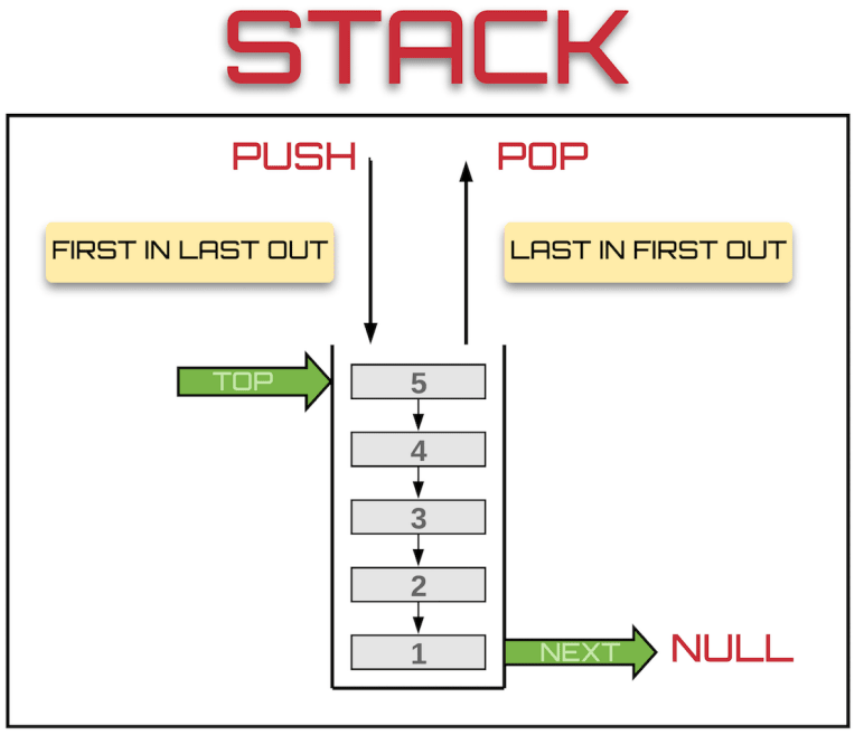
➢ Nguyên lý "**vào sau ra trước**": nghĩa là **phần tử cuối cùng được chèn vào ngăn xếp sẽ là phần tử đầu tiên được lấy ra khỏi ngăn xếp.**

Các kỹ thuật thao tác với ngăn xếp

❑ **Các kỹ thuật thao tác với ngăn xếp (Stack):**

➢ **push**: **thêm 1 phần tử vào đỉnh của ngăn xếp**, số phần tử của ngăn xếp tăng thêm 1.

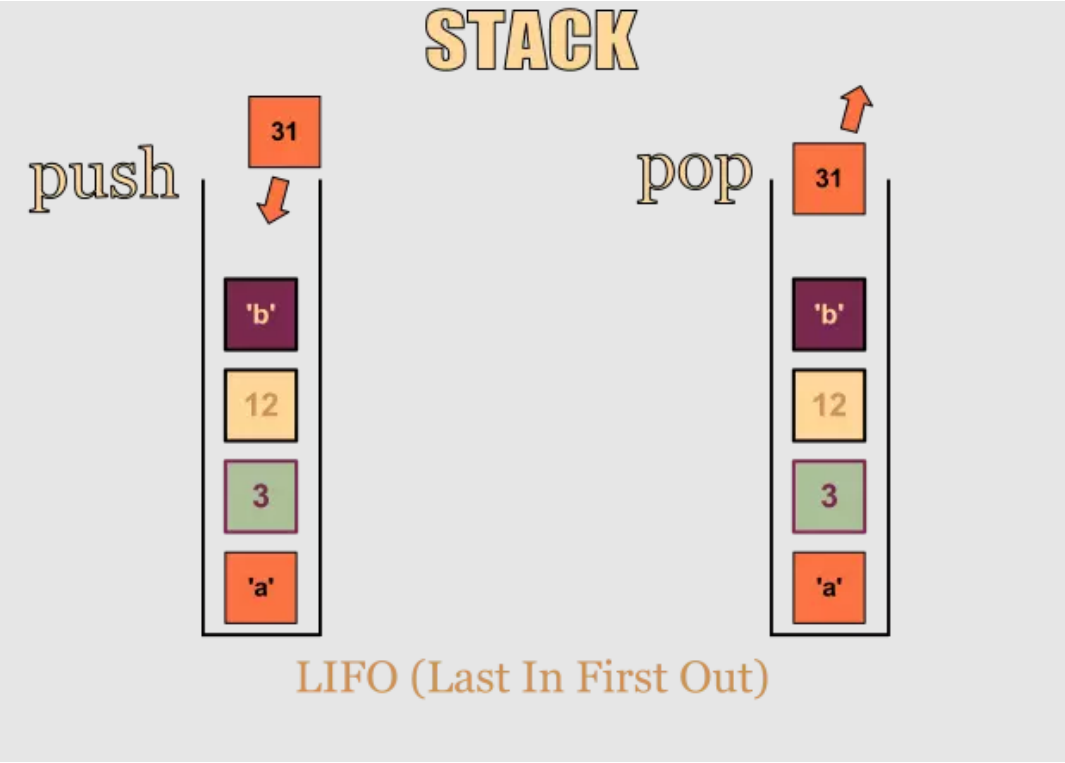


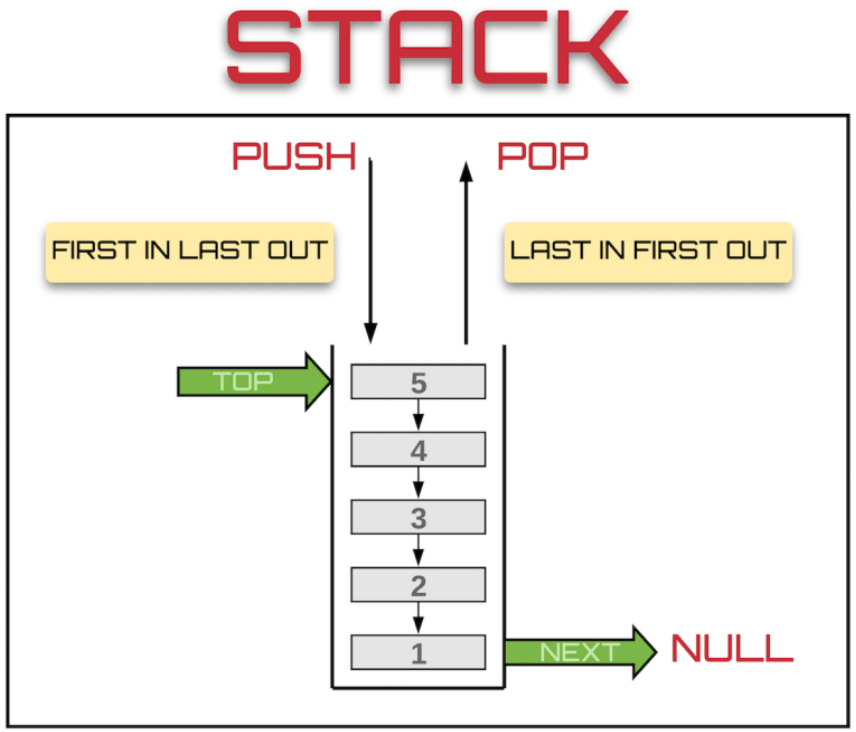


Các kỹ thuật thao tác với ngăn xếp

❑ **Các kỹ thuật thao tác với ngăn xếp (Stack):**

➢ **pop**: **xoá phần tử đầu tiên ở đỉnh của ngăn xếp**, số phần tử của ngăn xếp giảm 1.

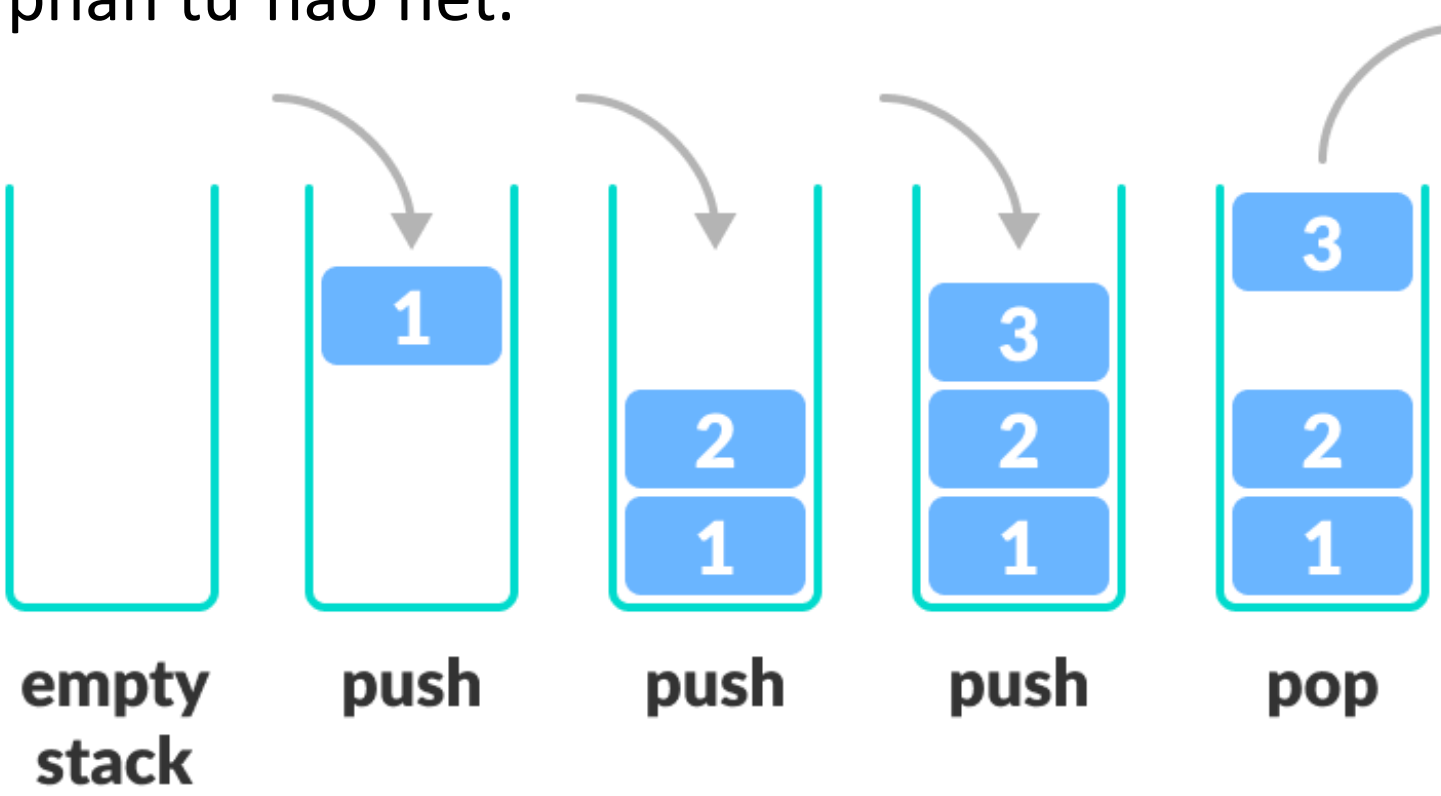




Các kỹ thuật thao tác với ngăn xếp

❑ **Các kỹ thuật thao tác với ngăn xếp (Stack):**

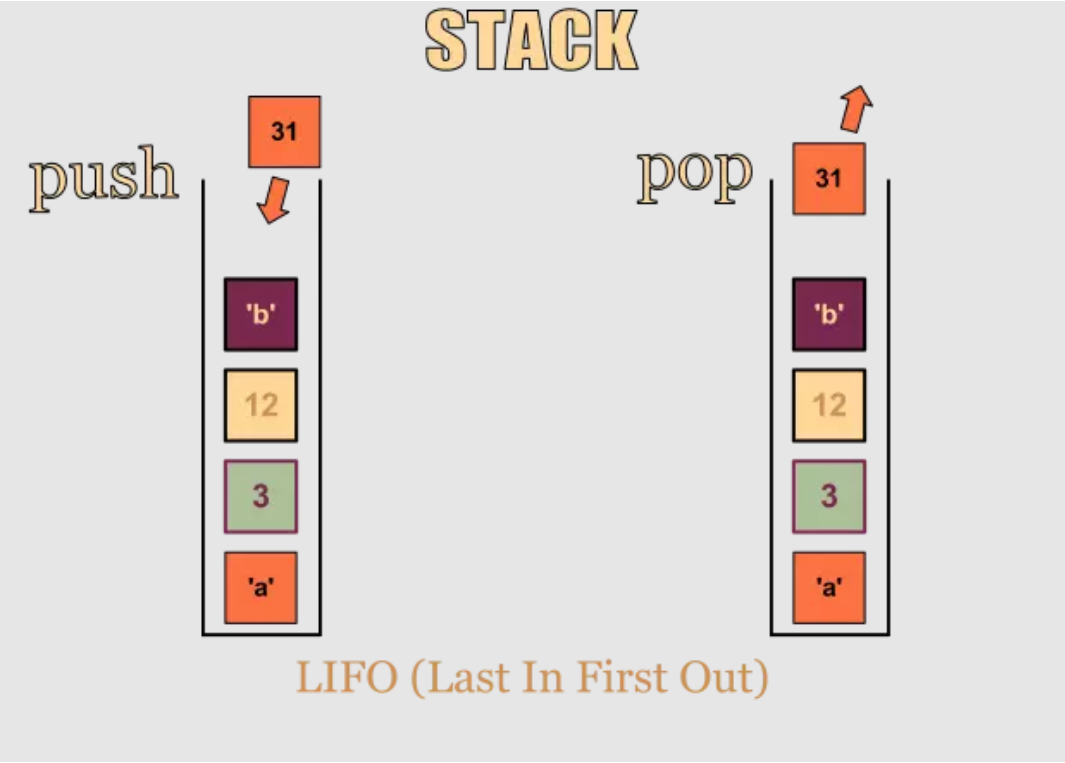
➢ **isEmpty**: **kiểm tra ngăn xếp có trống hay không**, ngăn xếp trống là ngăn xếp không có phần tử nào hết.

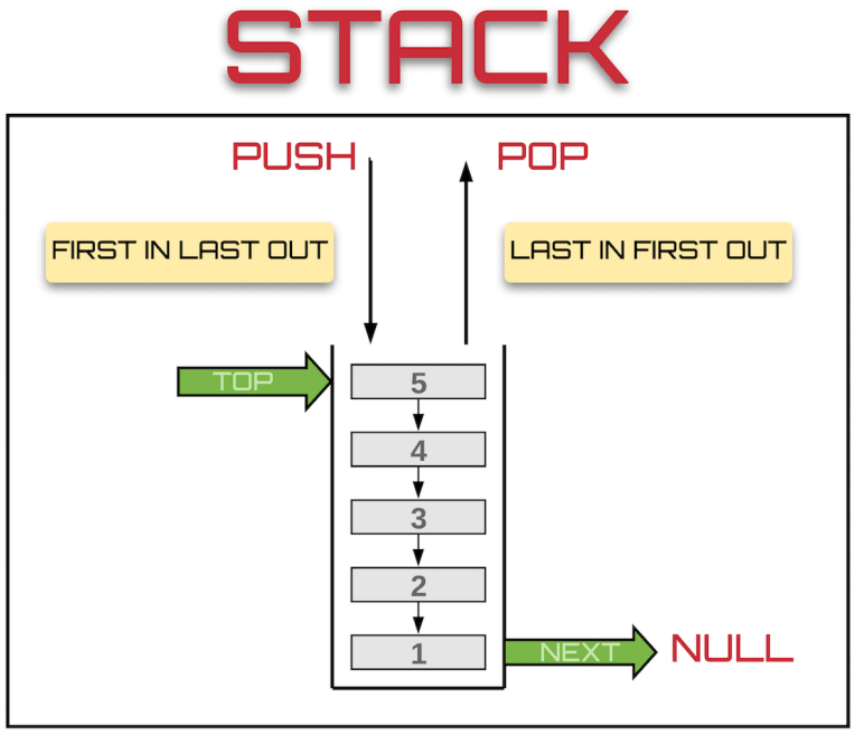


Các kỹ thuật thao tác với ngăn xếp

❑ Các kỹ thuật thao tác với ngăn xếp (Stack):

➢ **isFull**: **kiểm tra ngăn xếp đã đầy hay chưa**, ngăn xếp đầy là ngăn xếp đã có số lượng phần tử bằng với giới hạn của mảng (length) đã khởi tạo ban đầu.





Các kỹ thuật thao tác với ngăn xếp

❑ Các kỹ thuật thao tác với ngăn xếp (Stack):

➢ Khai báo ngăn xếp:

```
let stack = [];
var capacity; // Sức chứa của ngăn xếp
```

Stack of books

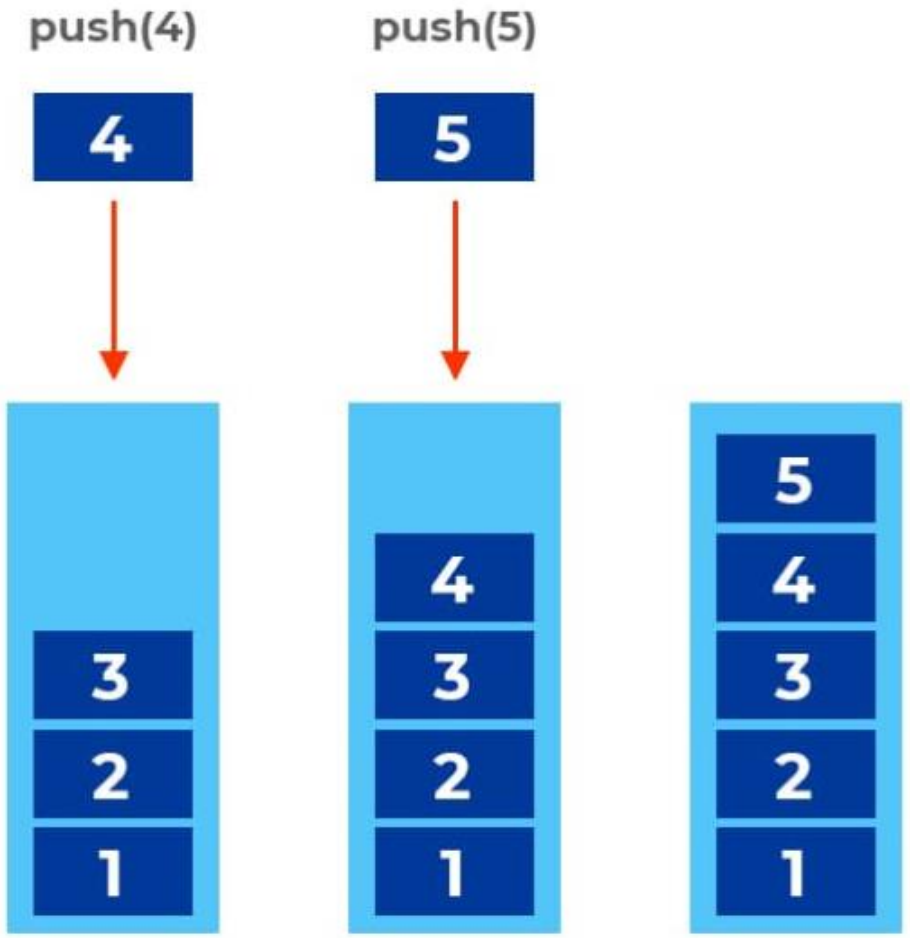
Stack of Coins

Các kỹ thuật thao tác với ngăn xếp

❑ Các kỹ thuật thao tác với ngăn xếp (Stack):

➢ Kiểm tra ngăn xếp **đã đầy**?

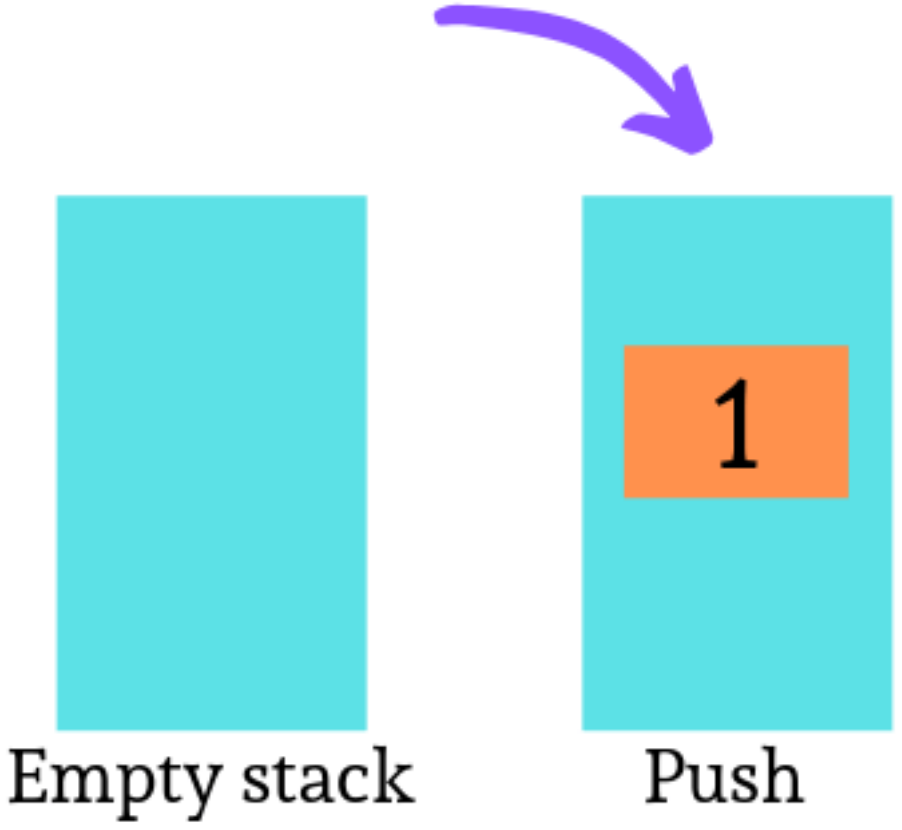
```
function isFull() {
    // số lượng phần tử >= sức chứa
    if (stack.length == capacity) {
        return true;
    } else {
        return false;
    }
}
```



Các kỹ thuật thao tác với ngăn xếp

- **Các kỹ thuật thao tác với ngăn xếp (Stack):**
  - Kiểm tra ngăn xếp **đang rỗng**?

```
function isEmpty() {
    // số lượng phần tử = 0
    if (stack.length == 0) {
        return true;
    } else {
        return false;
    }
}
```



Các kỹ thuật thao tác với ngăn xếp

❑ Các kỹ thuật thao tác với ngăn xếp (Stack):

➢ Thêm phần tử vào ngăn xếp (**push**)

```
function push(item) {
    if (isFull() ==  true) {
        console.log("Stack is full");
    } else {
        stack.push(item);
    }
}
```

Stack of books

Stack of Coins

Các kỹ thuật thao tác với ngăn xếp

❑ Các kỹ thuật thao tác với ngăn xếp (Stack):

➢ Thêm phần tử vào ngăn xếp (**push**)

```
stack.push(x);
```

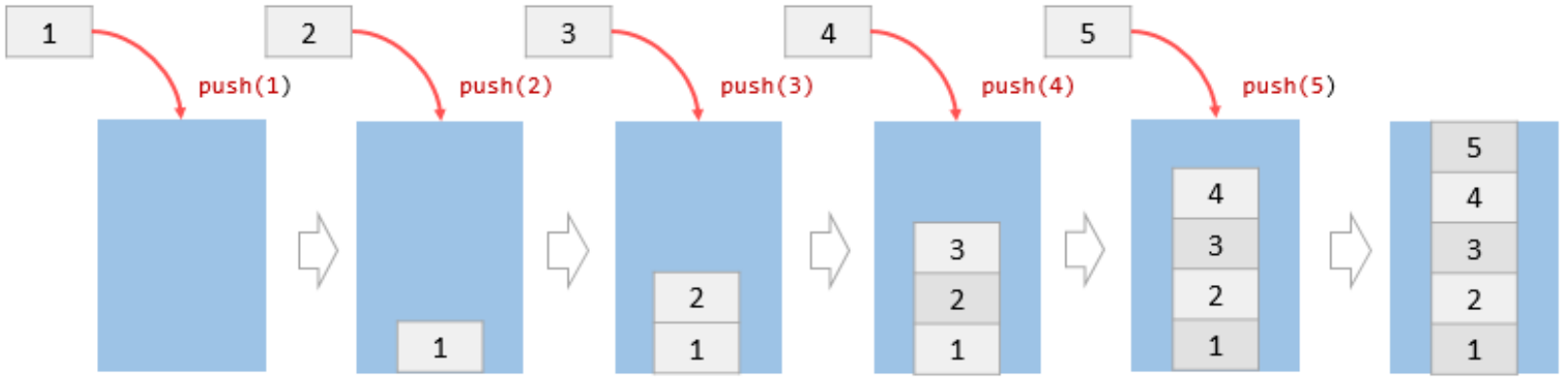
Stack of Books

Stack of Dishes

Stack of Discs

➢ Ví dụ:

```
let stack = [];
stack.push(1);
stack.push(2);
stack.push(3);
stack.push(4);
stack.push(5);
```



Các kỹ thuật thao tác với ngăn xếp

- **Các kỹ thuật thao tác với ngăn xếp (Stack):**
  - Xóa phần tử khỏi ngăn xếp (**pop**)

```
function pop() {
    if (isEmpty() ==  true) {
        console.log("Stack is empty");
    } else {
        stack.pop();
    }
}
```

Stack of books

Stack of Coins

Các kỹ thuật thao tác với ngăn xếp

❑ Các kỹ thuật thao tác với ngăn xếp (Stack):

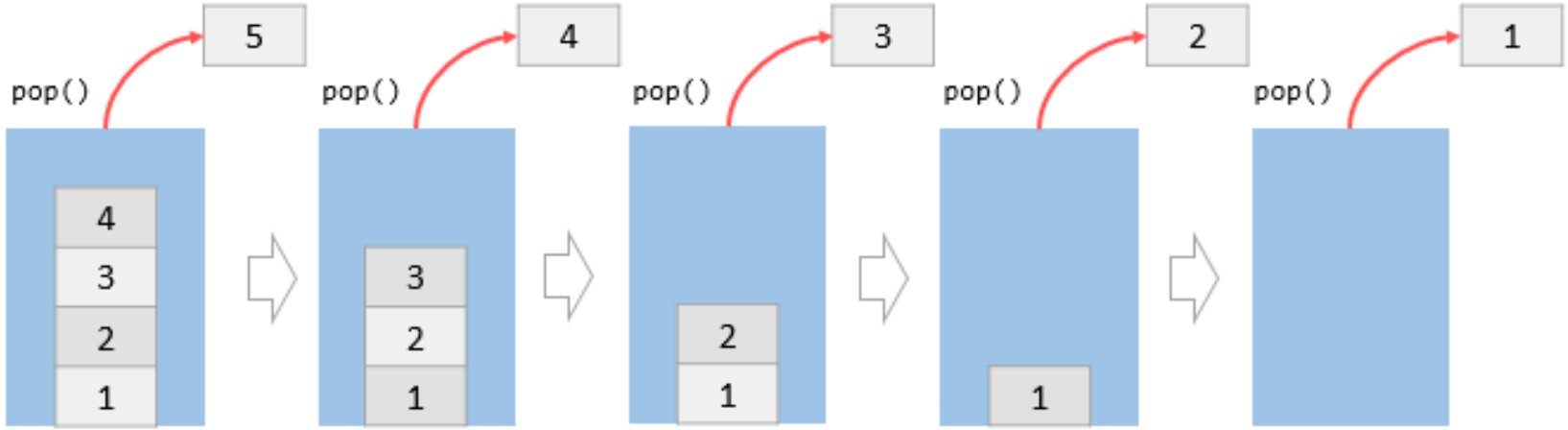
- ➢ Xóa phần tử khỏi ngăn xếp (**pop**)

```
stack.pop();
```

- ➢ Ví dụ:

```
stack.pop();
stack.pop();
stack.pop();
stack.pop();
stack.pop();
```

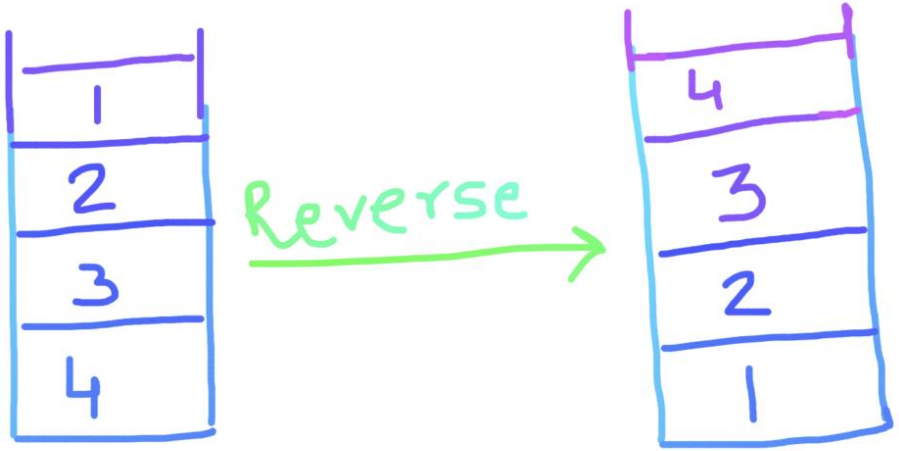
Stack of Books Stack of Dishes Stack of Discs



Các kỹ thuật thao tác với ngăn xếp

- **Các kỹ thuật thao tác với ngăn xếp (Stack):**
  - Nghịch đảo ngăn xếp (**reverse**)

```
function reverse() {
    return stack.reverse();
}
```



Stack of books

Stack of Coins

Tổng quan về cấu trúc dữ liệu hàng đợi

❑ Hàng đợi (Queue) là gì?

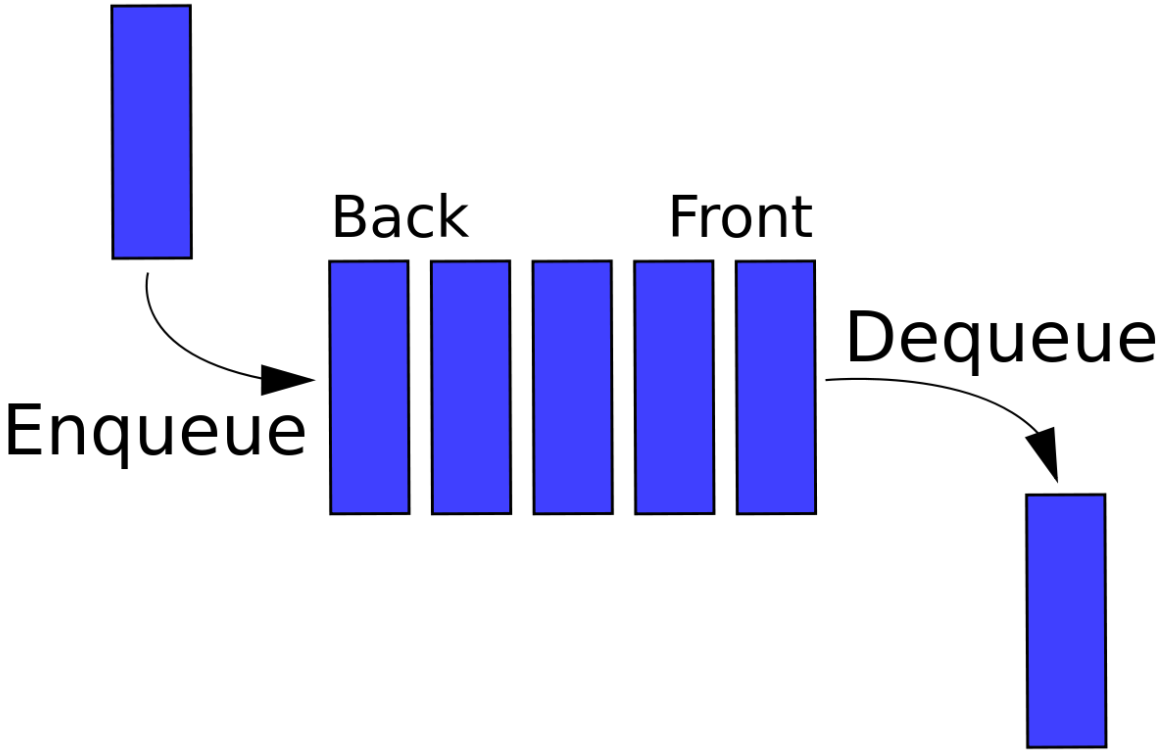
➢ Ngăn xếp (stack) là **một cấu trúc dữ liệu trừu tượng** hoạt động theo nguyên lý "**vào trước ra trước**". (**FIFO**: **F**irst **I**n **F**irst **O**ut)

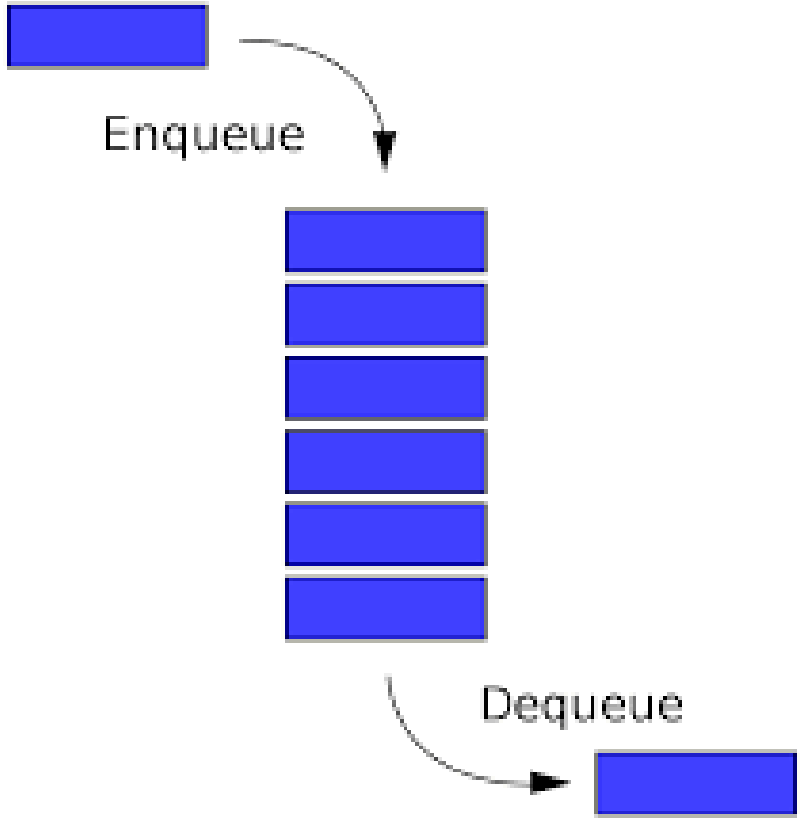
➢ Nguyên lý "**vào trước ra trước**": nghĩa là **phần tử đầu tiên được chèn vào hàng đợi** sẽ **là phần tử đầu tiên được lấy ra** khỏi **hàng đợi**.

Các kỹ thuật thao tác với hàng đợi

❑ **Các kỹ thuật thao tác với hàng đợi (Queue):**

➢ **enqueue**: **thêm 1 phần tử vào cuối (rear) của hàng đợi**, số phần tử của hàng đợi tăng thêm 1.

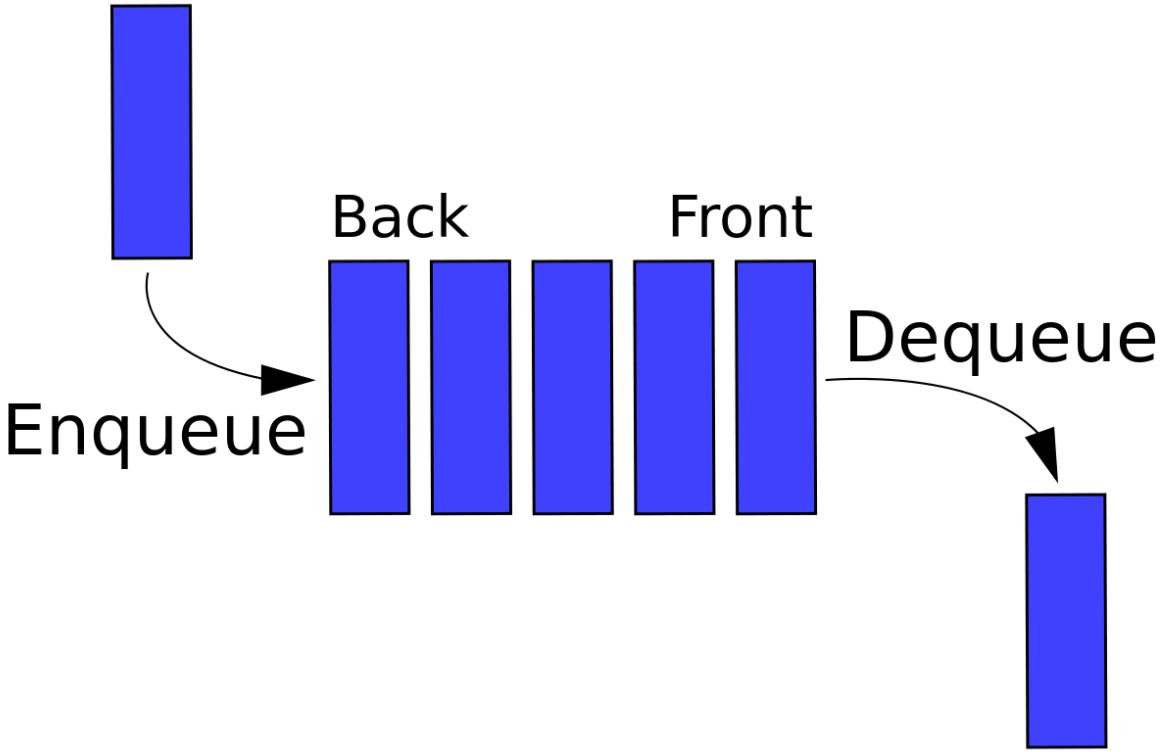


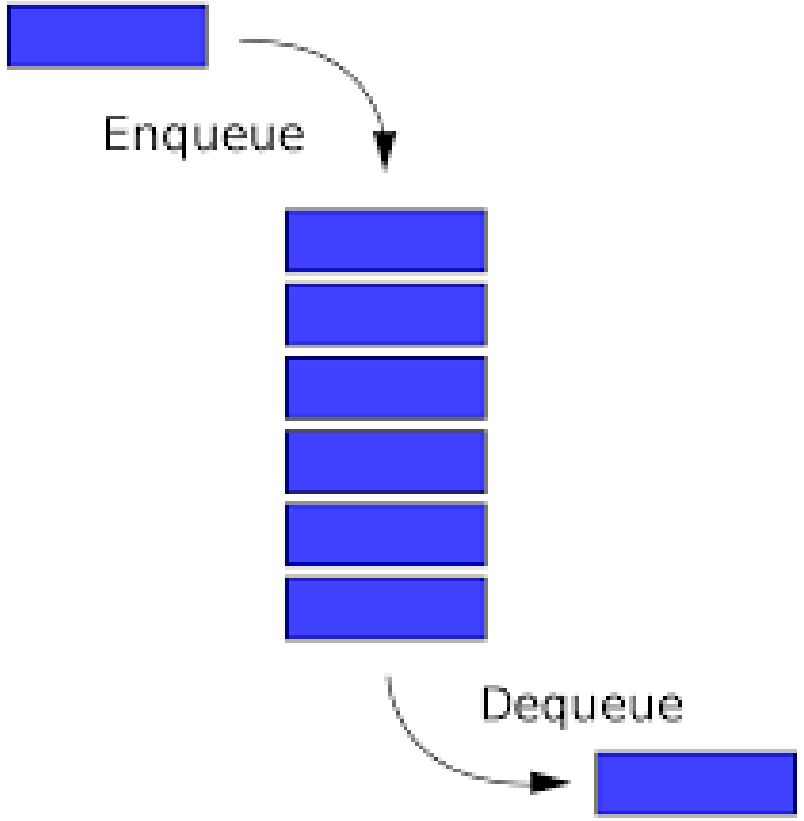


Các kỹ thuật thao tác với hàng đợi

❑ **Các kỹ thuật thao tác với hàng đợi (Queue):**

➢ **dequeue**: **xoá phần tử khỏi đầu (front) của hàng đợi**, số phần tử của ngăn xếp giảm 1.

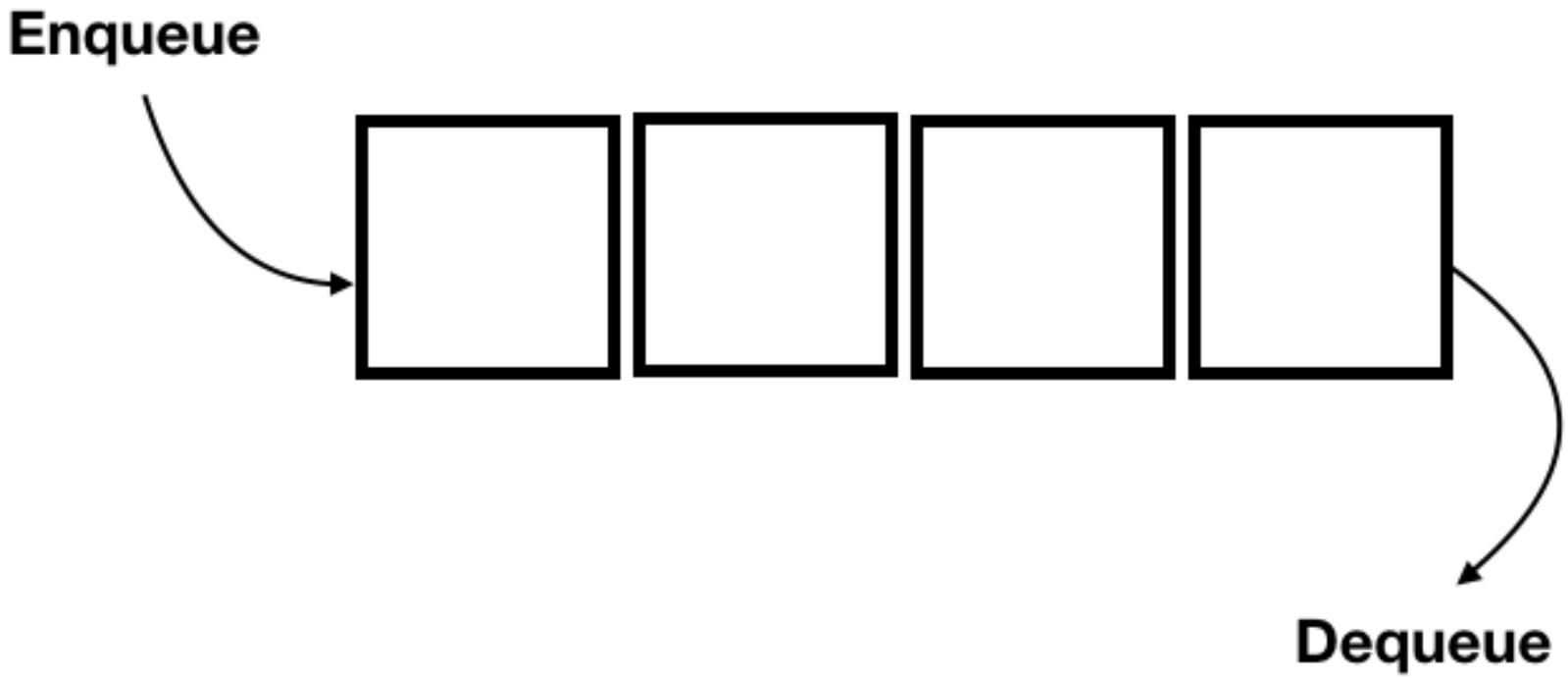




Các kỹ thuật thao tác với hàng đợi

❑ **Các kỹ thuật thao tác với hàng đợi (Queue):**

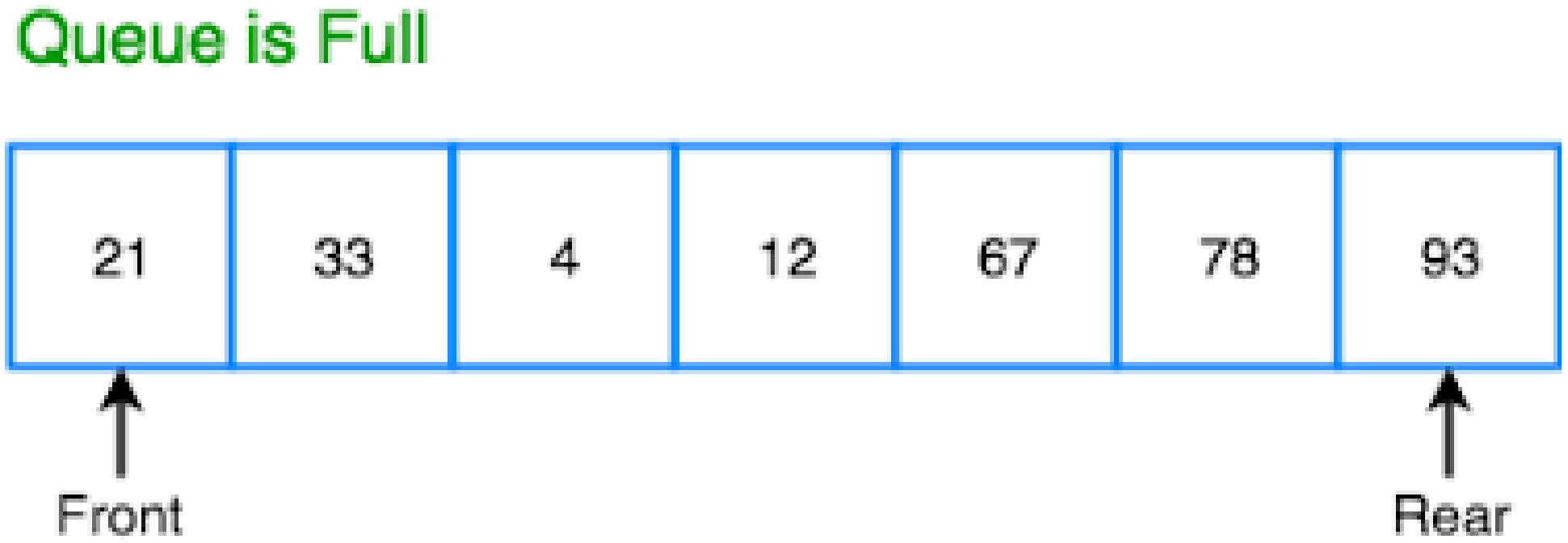
➢ **isEmpty**: **kiểm tra hàng đợi có trống hay không**, hàng đợi trống là hàng đợi không có phần tử nào hết.



Các kỹ thuật thao tác với hàng đợi

❑ **Các kỹ thuật thao tác với hàng đợi (Queue):**

➢ **isFull**: **kiểm tra hàng đợi đã đầy hay chưa**, hàng đợi đầy là hàng đợi đã có số lượng phần tử bằng với giới hạn của mảng (length) đã khởi tạo ban đầu.

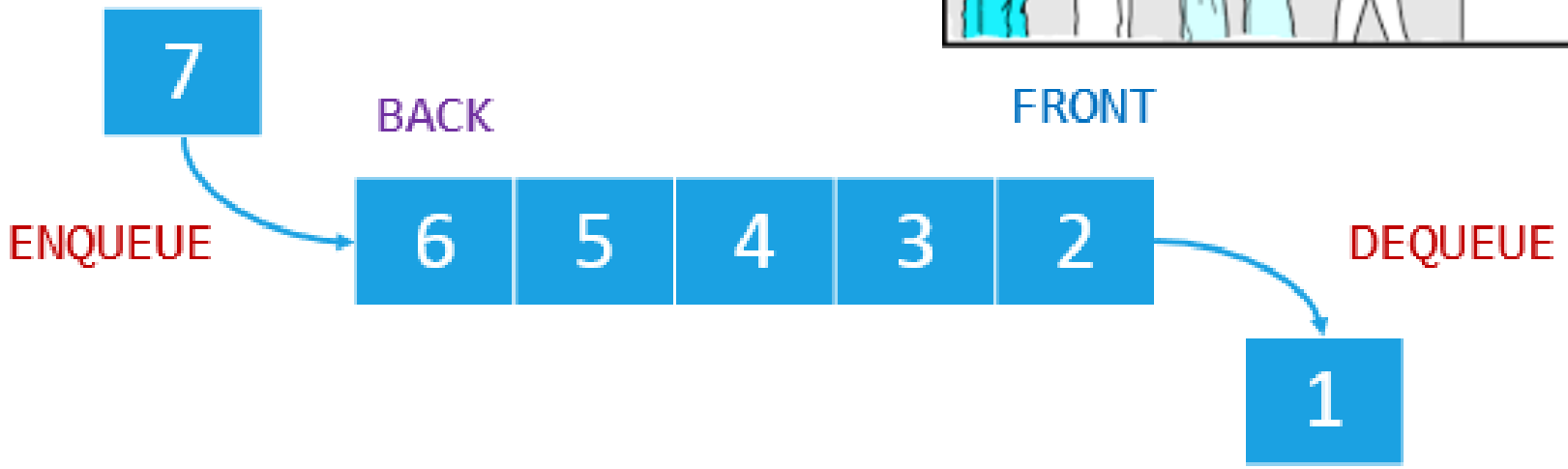


Các kỹ thuật thao tác với hàng đợi

❑ Các kỹ thuật thao tác với hàng đợi (Queue):

➢ Khai báo hàng đợi:

```
let queue = [];
var front; // Đầu hàng đợi
var rear; // Cuối hàng đợi
```

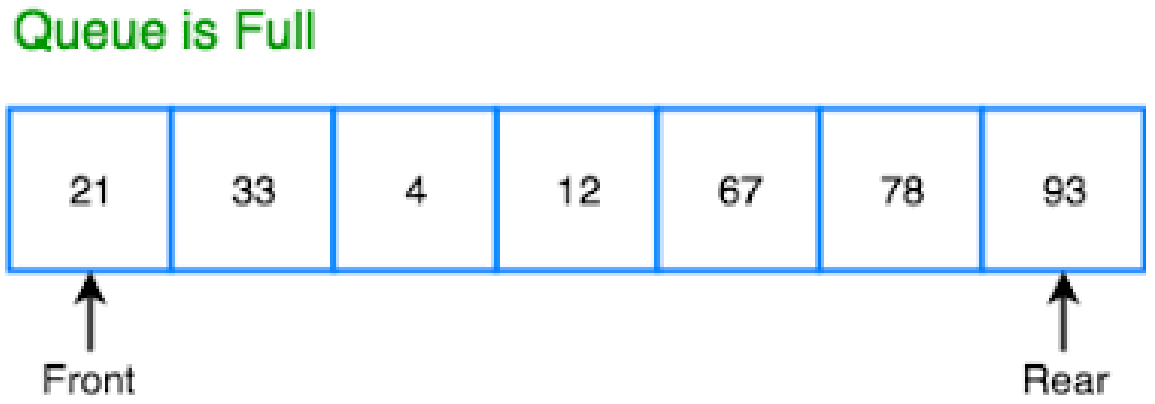


Các kỹ thuật thao tác với hàng đợi

❑ Các kỹ thuật thao tác với hàng đợi (Queue):

➢ Kiểm tra hàng đợi **đã đầy**?

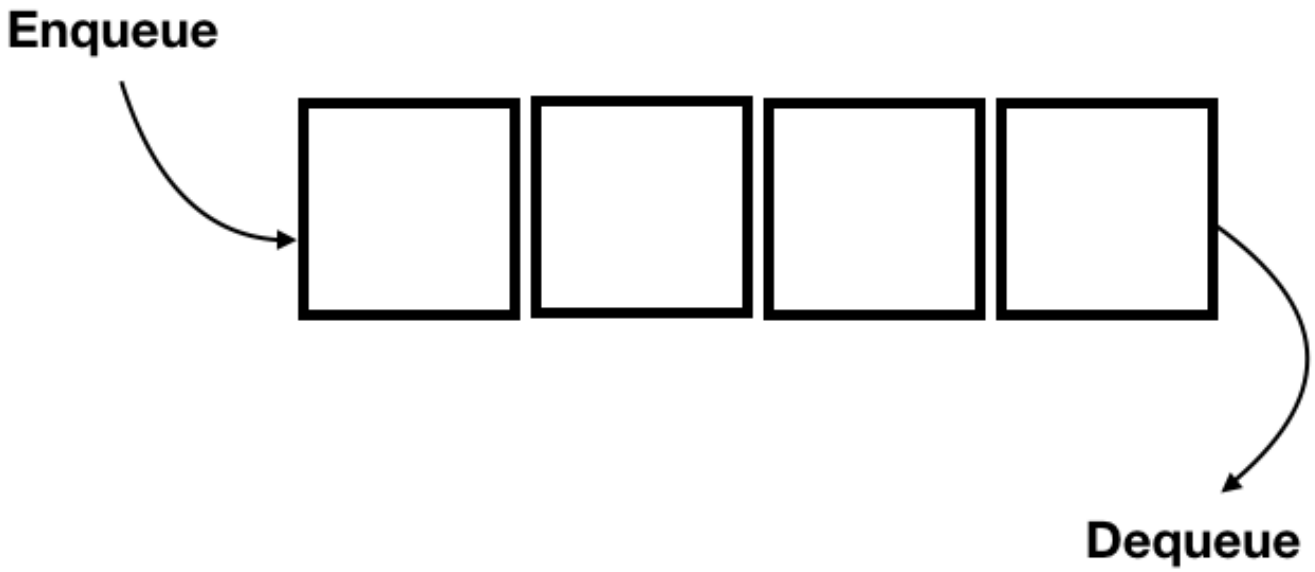
```
function isFull() {
    // số lượng phần tử = kích cỡ
    if (queue.length == queue.size) {
        return true;
    } else {
        return false;
    }
}
```



Các kỹ thuật thao tác với hàng đợi

- **Các kỹ thuật thao tác với hàng đợi (Queue):**
  - Kiểm tra hàng đợi **đang rỗng**?

```
function isEmpty() {
    // số lượng phần tử = 0
    if (queue.length == 0) {
        return true;
    } else {
        return false;
    }
}
```

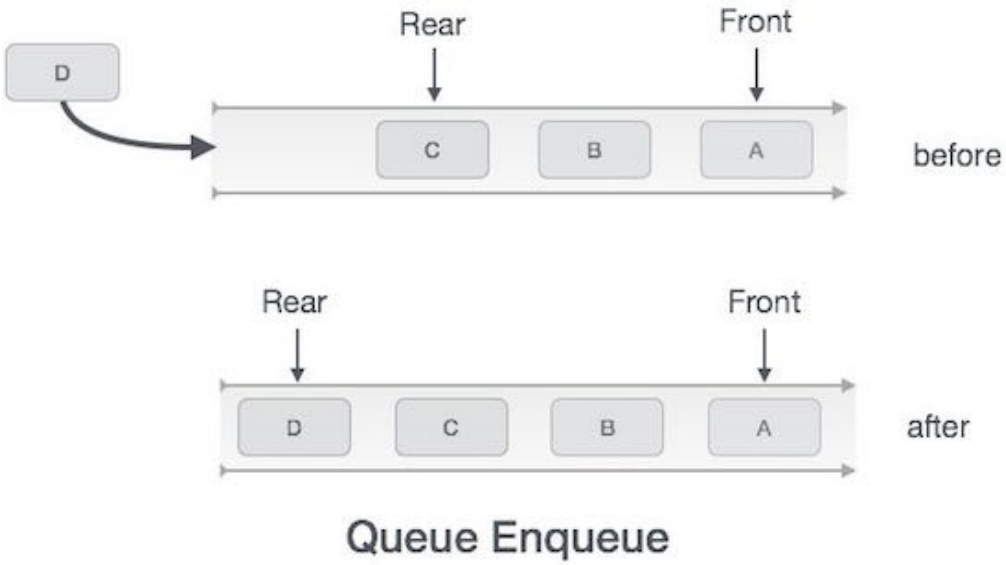


Các kỹ thuật thao tác với hàng đợi

❑ Các kỹ thuật thao tác với hàng đợi (Queue):

➢ Thêm phần tử vào hàng đợi (**enqueue**):

```
function enqueue(item) {
    return queue.push(item);
   // return queue.unshift(item);
}
```



Queue Enqueue

➢ <u>Lưu ý khi dùng</u>:

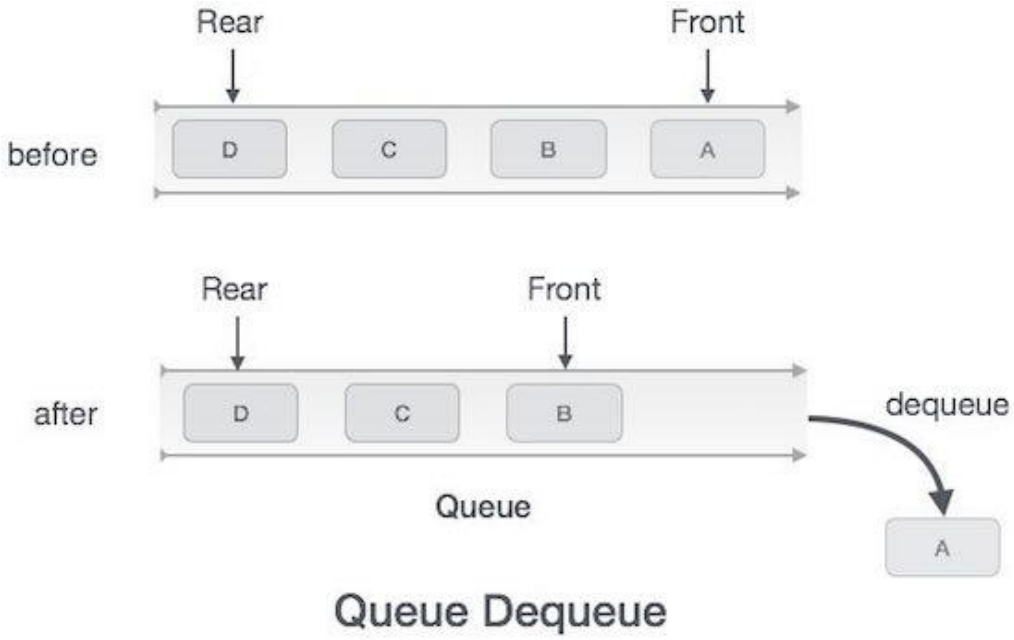
- push (enqueue) -> shift (dequeue)
- unshift (enqueue) -> pop (dequeue)

Các kỹ thuật thao tác với hàng đợi

❑ Các kỹ thuật thao tác với hàng đợi (Queue):

➢ Xóa phần tử khỏi hàng đợi (**dequeue**):

```
function dequeue(item) {
    return queue.shift(item);
   // return queue.pop(item);
}
```



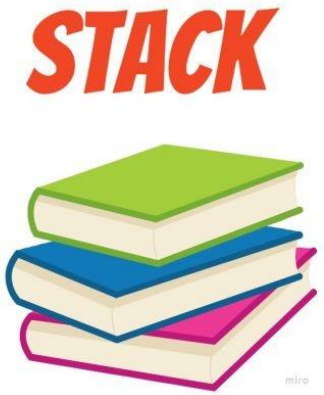
Queue Dequeue

➢ <u>Lưu ý khi dùng</u>:

- push (enqueue) -> shift (dequeue)
- unshift (enqueue) -> pop (dequeue)

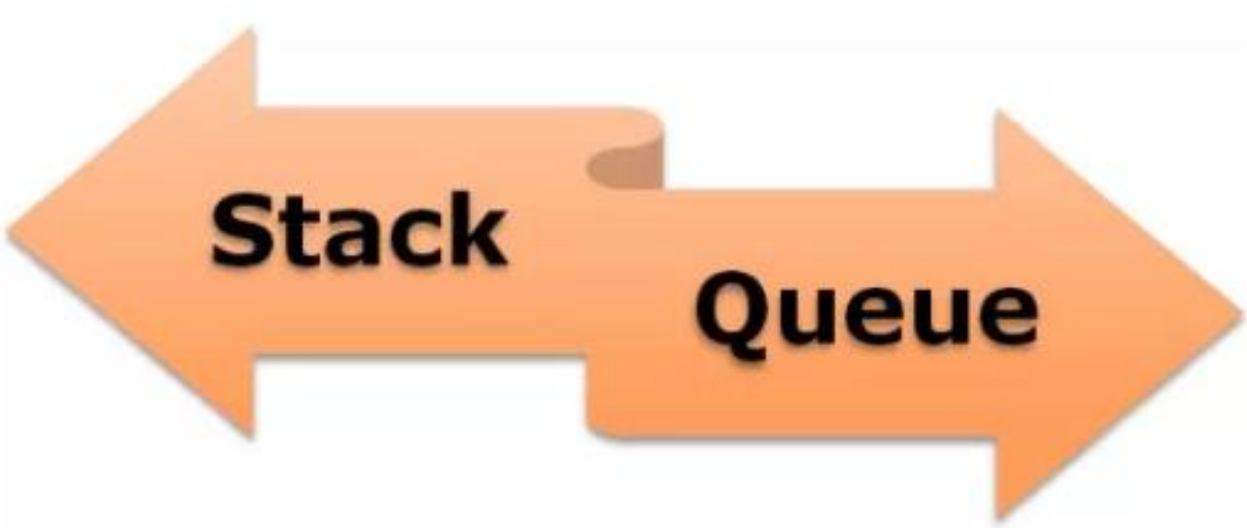
So sánh ngăn xếp và hàng đợi

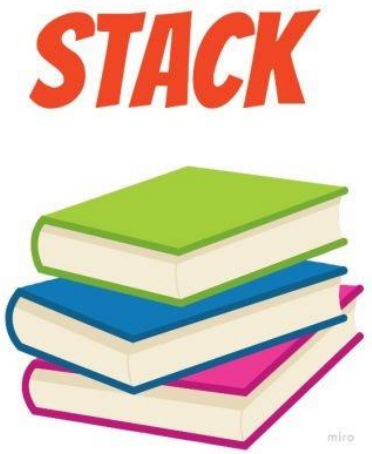
- **Stack** và **Queue** đều là các cấu trúc dữ liệu không nguyên thủy (non-primitive).
- Sự khác biệt lớn nhất giữa **Stack** và **Queue** là:
  - **Stack** sử dụng phương thức **LIFO** (Last In First Out) để truy cập và thêm các phần tử dữ liệu
  - **Queue** sử dụng phương thức **FIFO** (First In First Out) để truy cập và thêm các phần tử dữ liệu.



So sánh ngăn xếp và hàng đợi

- **Stack** chỉ có một đầu mở để **pushing** và **popping** các phần tử dữ liệu, còn **Queue** có cả hai đầu mở để **enqueuing** và **dequeuing** các phần tử dữ liệu.
- **Stack** và **Queue** là các cấu trúc dữ liệu được sử dụng để lưu trữ các yếu tố dữ liệu và nó dựa trên một số các ví dụ có thực trong cuộc sống hàng ngày của chúng ta.





So sánh ngăn xếp và hàng đợi

- Cơ chế hoạt động:
  - **Stack** chỉ có một đầu mở để **pushing** và **popping** các phần tử dữ liệu.
  - **Queue** có cả 2 đầu mở để **enqueuing** và **dequeuing** các phần tử dữ liệu.

Stack of Books

Stack of Dishes

So sánh ngăn xếp và hàng đợi

- Ví dụ, **Stack** là một chồng đĩa CD, nơi bạn có thể lấy ra và đưa vào đĩa CD thông qua đỉnh của ngăn xếp đĩa CD.

Stack of Books

Stack of Dishes

Stack of Discs

So sánh ngăn xếp và hàng đợi

- Tương tự, **Queue** là hàng đợi cho mua vé của Nhà hát nơi người đứng ở vị trí đầu tiên sẽ được phục vụ trước, người đến sau sẽ ở phía sau hàng đợi.

So sánh ngăn xếp và hàng đợi

- Sau khi học tìm hiểu, ta có **bảng so sánh một số sự khác nhau** như sau**:**

CƠ CHẾ SO SÁNH	STACK	QUEUE
Nguyên tắc làm việc	LIFO (Last In First Out)	FIFO (First In First Out)
Cấu trúc	Dùng một đầu để chèn và xóa các phần tử	Có 2 đầu để xử lý dữ liệu, một đầu chèn, một đầu xóa
Số con trỏ được sử dụng	Một	Hai (hoặc hơn)
Hoạt động được thực hiện	Push và Pop	Enqueue và Dequeue

So sánh ngăn xếp và hàng đợi

- Sau khi học tìm hiểu, ta có **bảng so sánh một số sự khác nhau** như sau**:**

CƠ CHẾ SO SÁNH	STACK	QUEUE
Kiểm tra empty condition	Top == -1	Front == -1
Kiểm tra full condition	Top == Max -1	Rear == Max – 1
Biến thể	Không có biến thể	Có nhiều biến thể
Thực hiện	Đơn giản	Tương đối phức tạp

Tổng kết:

- ❑ Tổng quan về cấu trúc dữ liệu ngăn xếp
- ❑ Các kỹ thuật để thao tác với ngăn xếp
- ❑ Tổng quan về cấu trúc dữ liệu hàng đợi
- ❑ Các kỹ thuật để thao tác với hàng đợi
- ❑ So sánh ngăn xếp và hàng đợi