Ngăn xếp và Hàng đợi (Stacks and Queues)

Nguyễn Mạnh Hiển
hiennm@tlu.edu.vn

Nội dung

1. Ngăn xếp
2. Hàng đợi

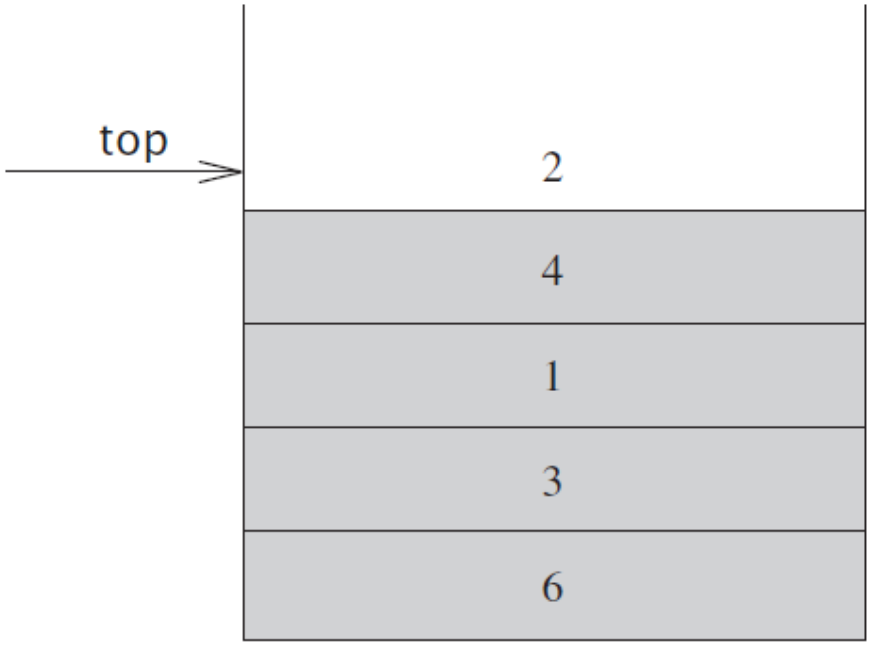
1. Ngăn xếp

Ngăn xếp

- Một danh sách theo kiểu **vào sau ra trước**

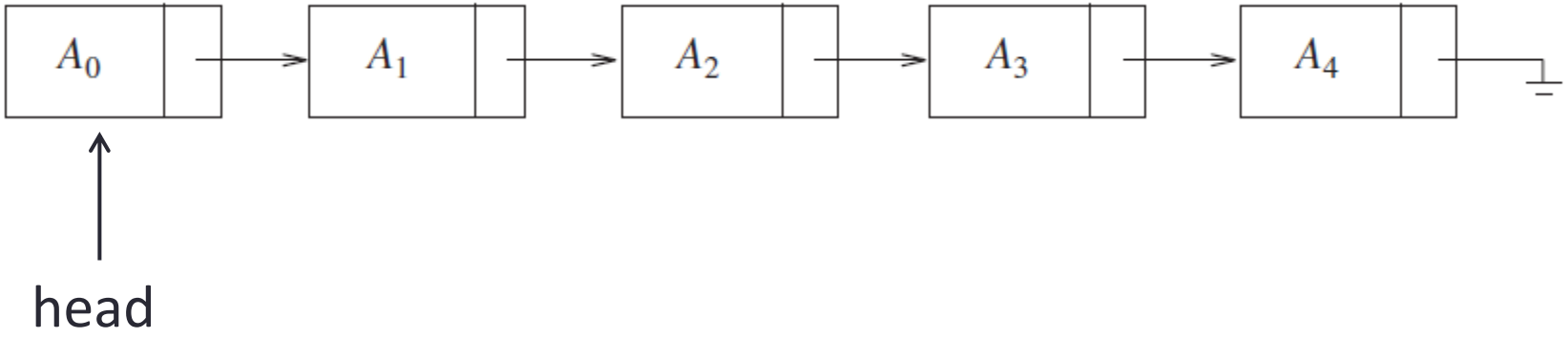
  LIFO (Last In First Out)

- Ba thao tác cơ bản (xảy ra ở đỉnh ngăn xếp):
  - **push**: Thêm phần tử
  - **pop**: Xóa phần tử
  - **top**: Truy nhập phần tử
- Các thao tác khác:
  - Lấy kích thước
  - Kiểm tra rỗng



Cài đặt ngăn xếp – cách 1

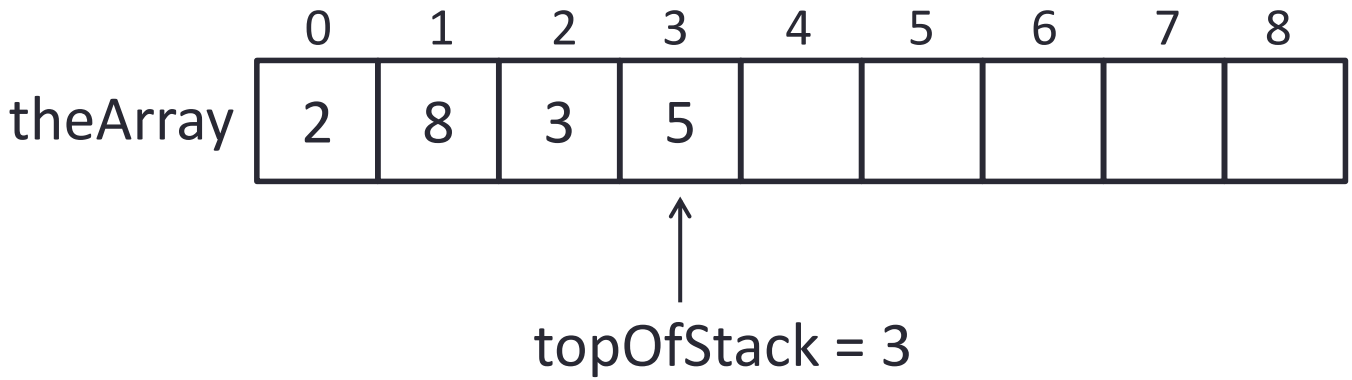
- Cài đặt bằng danh sách liên kết đơn:



- Các thao tác:
  - **push**: gọi thao tác pushFront của DSLK đơn
  - **pop**: gọi thao tác popFront của DSLK đơn
  - **top**: gọi thao tác front của DSLK đơn

Cài đặt ngăn xếp – cách 2

- Cài đặt bằng mảng:



- **push(e)**: topOfStack++, theArray[topOfStack] = e
- **pop**: topOfStack--
- **top**: return theArray[topOfStack]
- Chú ý: Khi ngăn xếp rỗng thì topOfStack = -1

Một số ứng dụng của ngăn xếp

- Cân bằng thẻ (tag) trong một trang HTML
- Định giá biểu thức hậu tố

Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```

Use a Stack, it starts out empty....

Start reading the input (just strings)

Top of Stack →

Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```

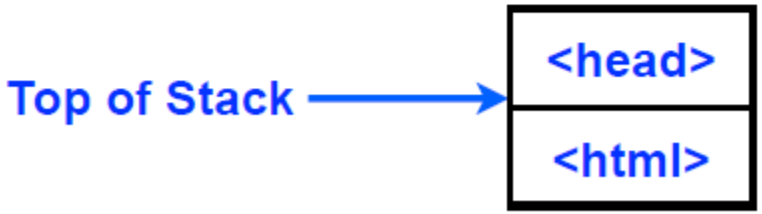
Encounter an open tag, push it on the stack

Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```

Encounter an open tag, push it on the stack

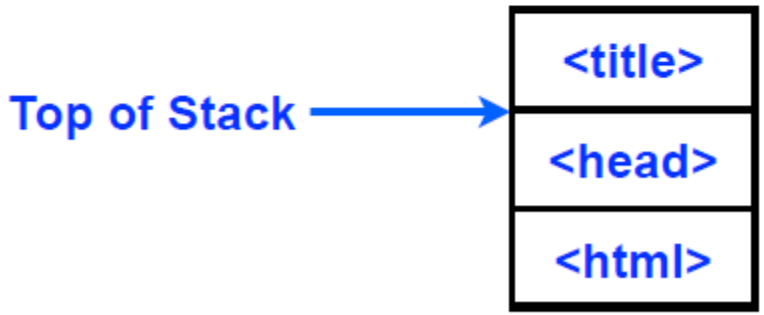


Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```

Encounter an open tag, push it on the stack

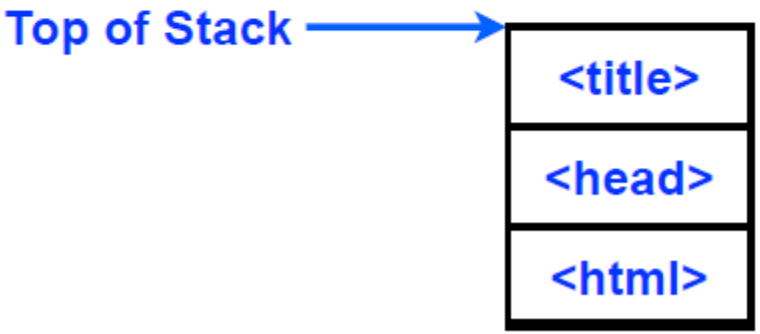


Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```

All the tags on the stack apply to any (non-tag) we encounter

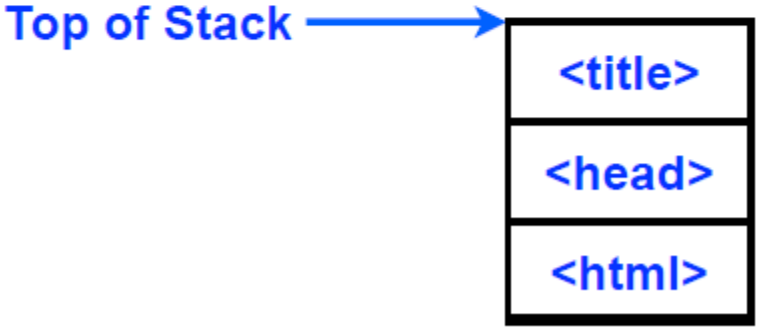


Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```

Encounter a close tag:
pop the stack (remove its top)

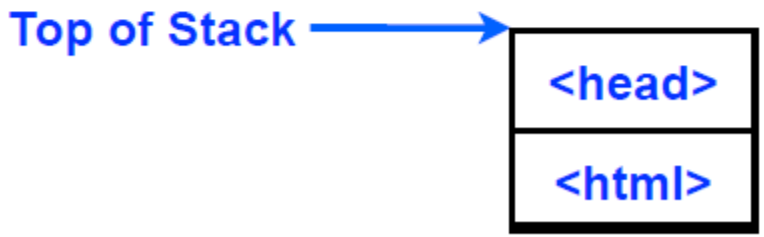


Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```

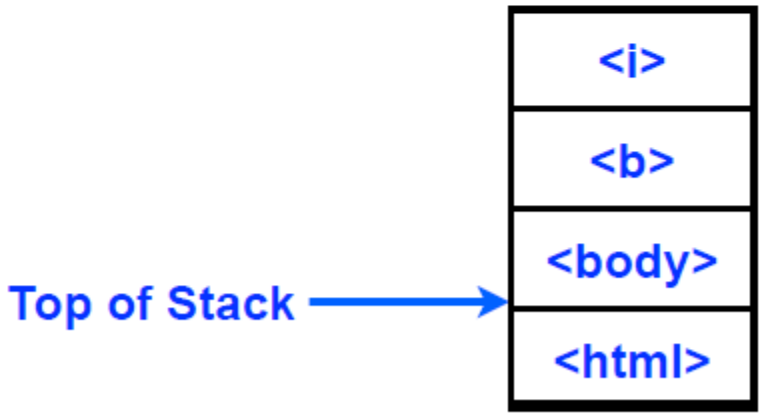
Encounter a close tag:
pop the stack (remove its top)



Matched Tags Example: HTML

- HTML: balanced tags (e.g., <B> for bold, </B>ends bold)

```
<html>
<head>
<title>Example Page</title>
</head>
<body>
Some text
<b>Some bold text
<i>and bold italics
</i> just bold</b>
</body>
</html>
```



Định giá biểu thức hậu tố

- Giả sử phải định giá biểu thức trung tố sau:

  `4,99 * 1,06 + 5,99 + 6,99 * 1,06`

  - Máy tính khoa học sẽ cho kết quả 18,69 → đúng
  - Máy tính giản đơn (tính tuần tự từ trái sang phải) sẽ cho kết quả 19,37 → **sai !**

- Nếu tổ chức biểu thức dưới dạng hậu tố (toán tử viết sau các toán hạng của nó) rồi ứng dụng ngăn xếp, tính tuần tự từ trái sang phải sẽ cho kết quả đúng (tức là không cần quan tâm độ ưu tiên của các toán tử)

  `4,99 1,06 * 5,99 + 6,99 1,06 * +`

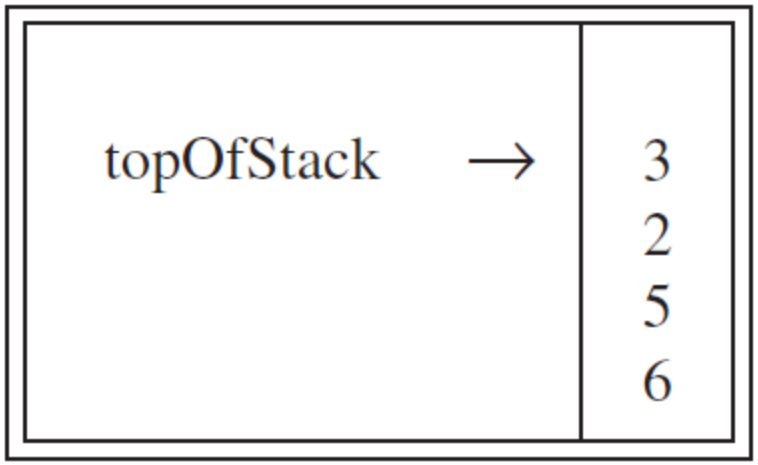
Quy tắc

Duyệt các thành phần (toán hạng/toán tử) trong biểu thức từ trái sang phải:

- Gặp toán hạng:
  - Đặt nó vào ngăn xếp
- Gặp toán tử:
  - Lấy hai toán hạng ra khỏi ngăn xếp và áp dụng toán tử
  - Đặt kết quả vào ngăn xếp

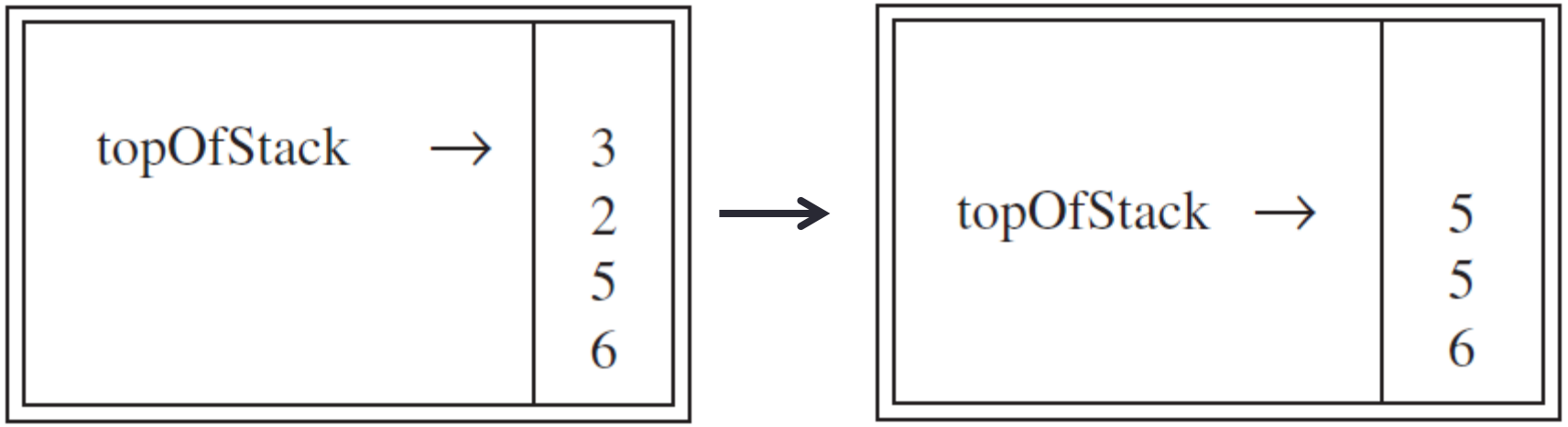
Ví dụ

- Định giá biểu thức **6 5 2 3 + 8 * + 3 + ***
- Đặt bốn toán hạng đầu tiên vào ngăn xếp



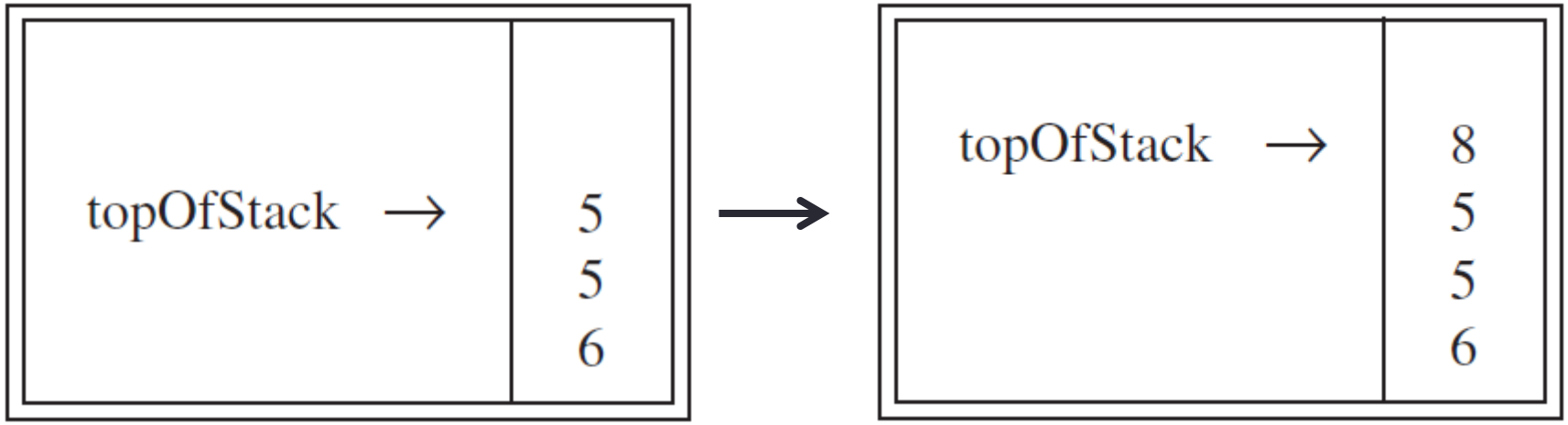
6 5 2 3 + 8 * + 3 + *

- Đọc “+”, lấy 3 và 2 ra, cộng lại được 5, đặt 5 vào ngăn xếp



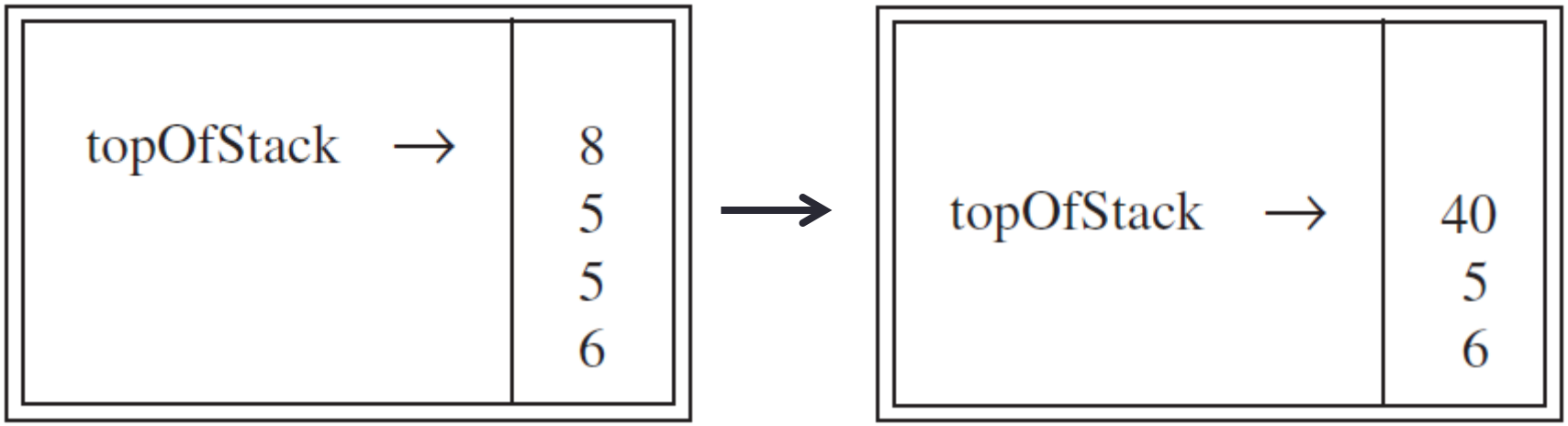
6 5 2 3 + 8 * + 3 + *

- Đặt 8 vào ngăn xếp



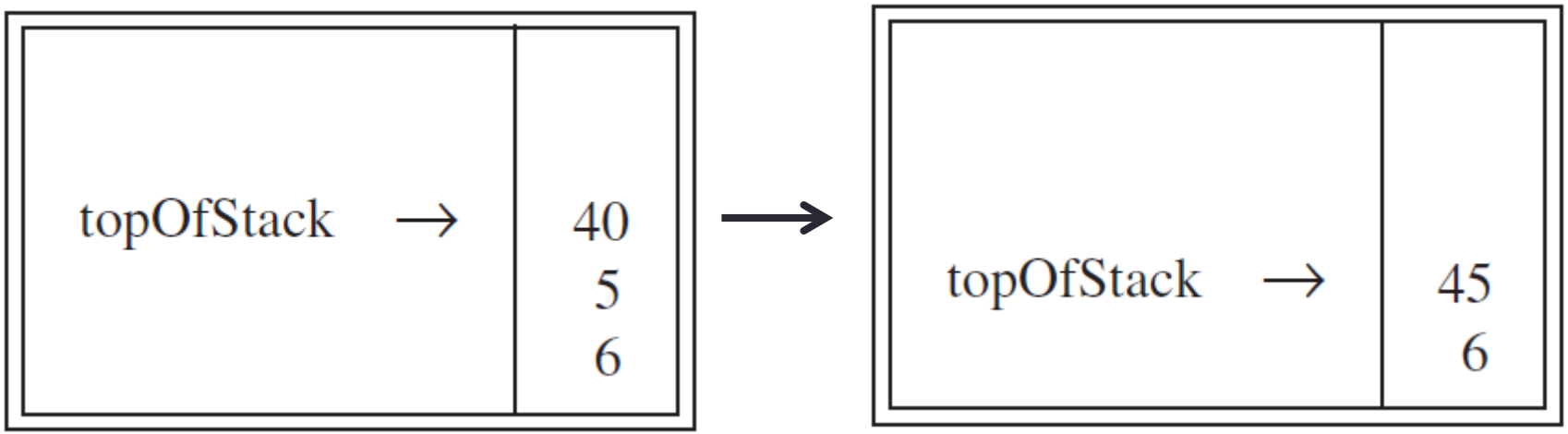
6 5 2 3 + 8 * + 3 + *

- Đọc "*", lấy 8 và 5 ra, nhân vào được 40, đặt 40 vào ngăn xếp



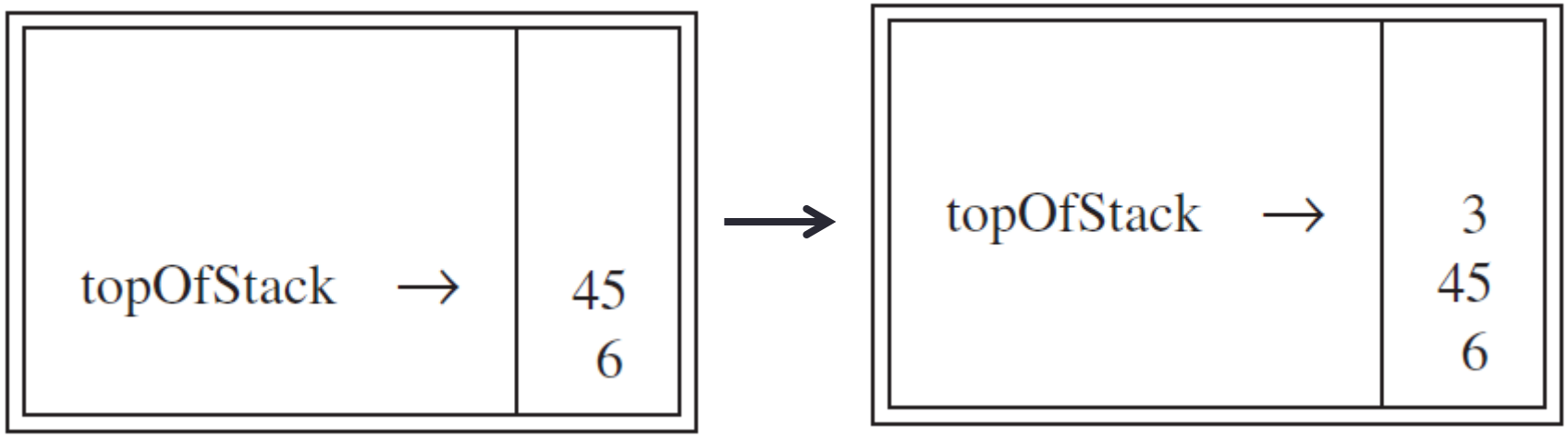
6 5 2 3 + 8 * + 3 + *

- Đọc “+”, lấy 40 và 5 ra, cộng lại được 45, đặt 45 vào ngăn xếp



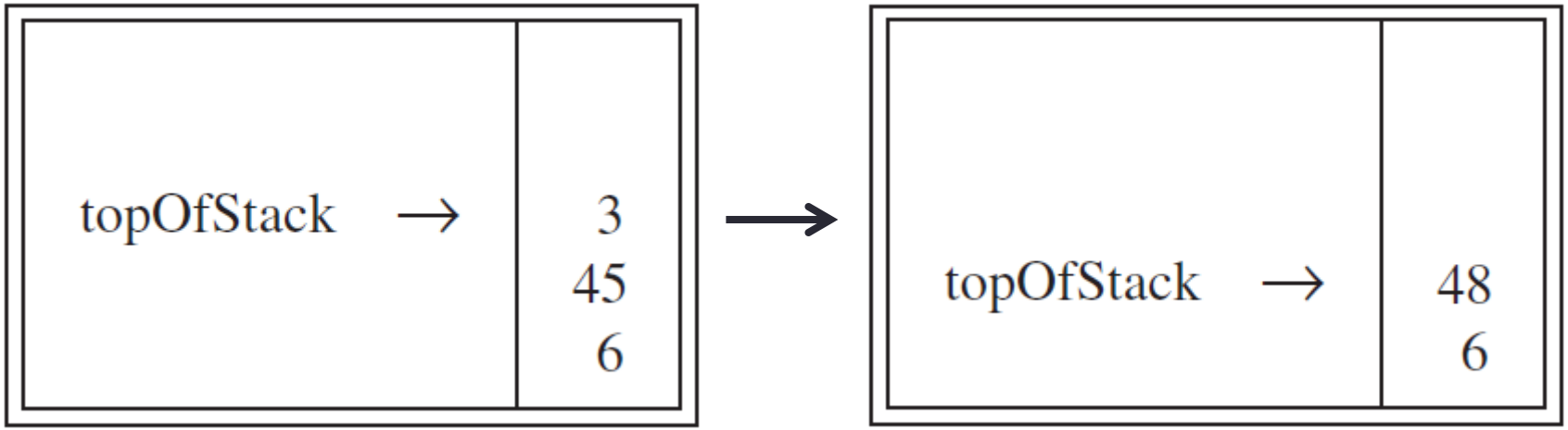
6 5 2 3 + 8 * + 3 + *

- Đặt 3 vào ngăn xếp



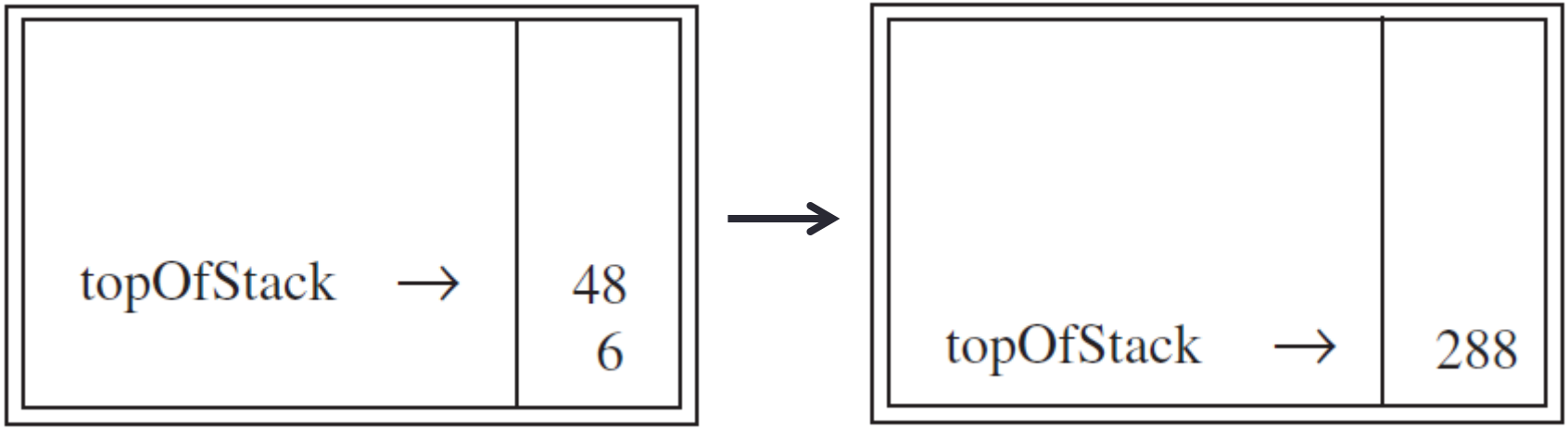
6 5 2 3 + 8 * + 3 + *

- Đọc “+”, lấy 3 và 45 ra, cộng lại được 48, đặt 48 vào ngăn xếp



6 5 2 3 + 8 * + 3 + *

- Đọc "*", lấy 48 và 6 ra, nhân vào được 288, đặt 288 vào ngăn xếp



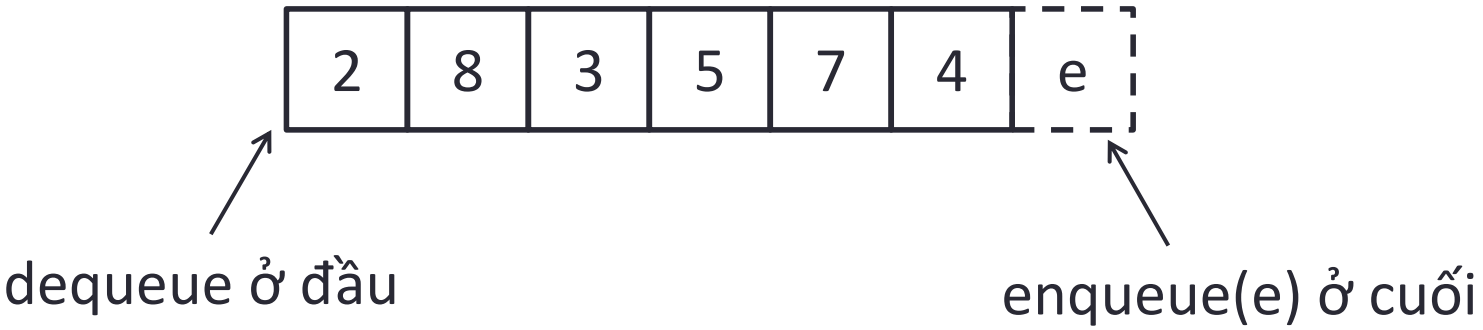
Thời gian định giá biểu thức hậu tố là O(n), với n là số toán tử và toán hạng

2. Hàng đợi

Hàng đợi

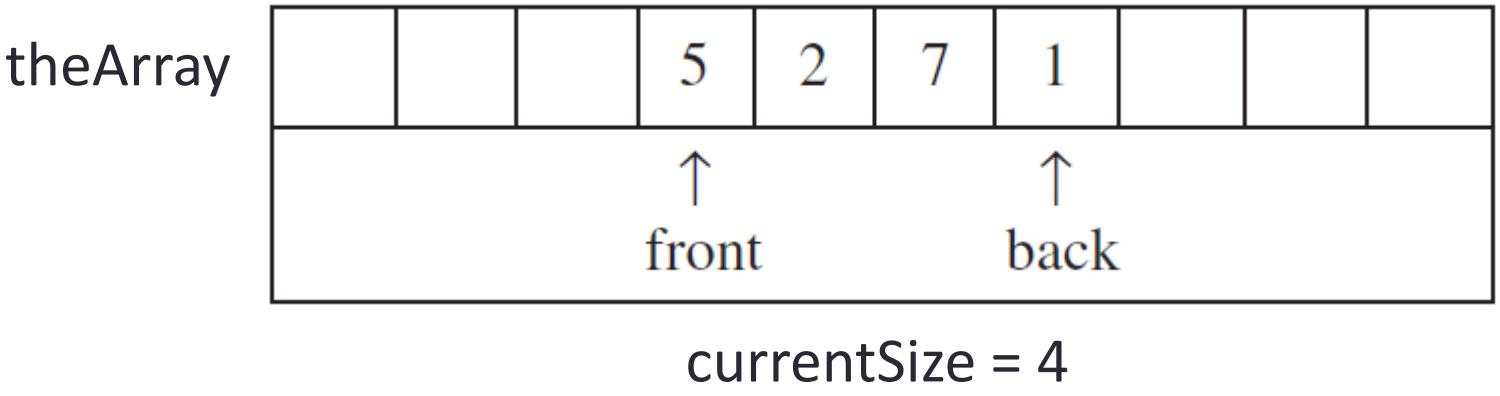
- Một danh sách theo kiểu **vào trước ra trước**

  FIFO (First In First Out)
- Hai thao tác cơ bản:
  - **enqueue**: chèn phần tử vào cuối danh sách
  - **dequeue**: xóa phần tử ở đầu danh sách

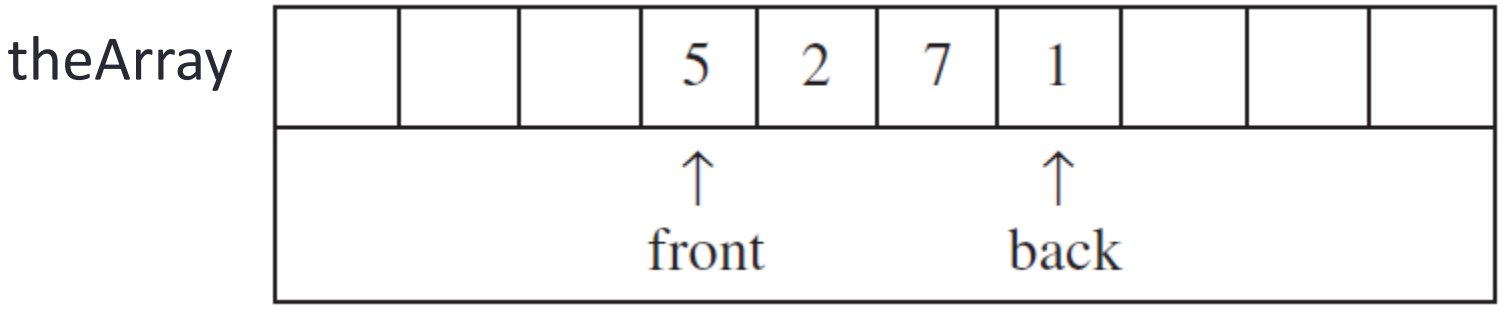


Cài đặt hàng đợi

- Như trường hợp ngăn xếp, có thể dùng mảng hoặc danh sách liên kết để cài đặt hàng đợi
- Các thao tác đều rất nhanh: O(1)
- Ở đây chúng ta chỉ xem xét cài đặt bằng mảng:



Cài đặt hàng đợi bằng mảng

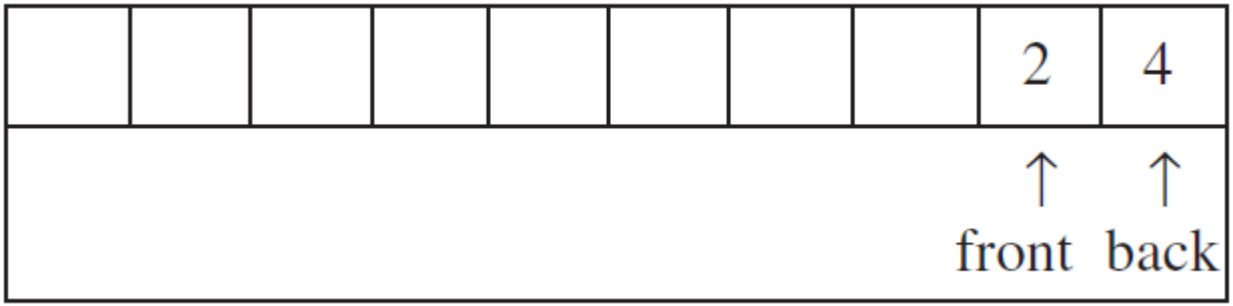


currentSize = 4

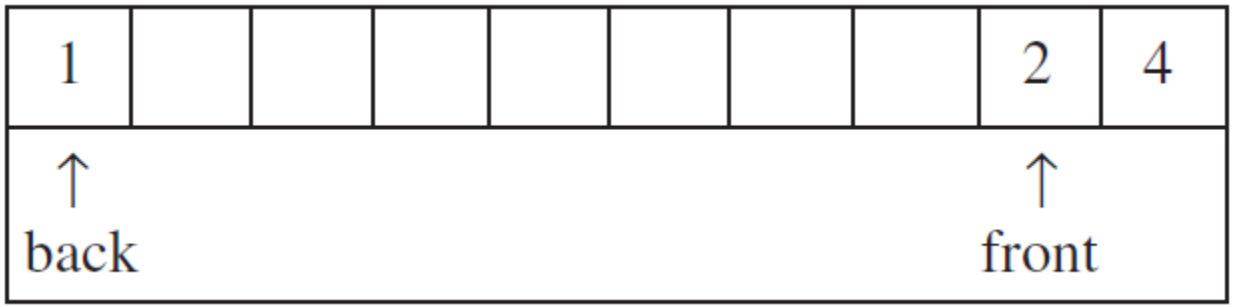
- **enqueue(e)**: back++, theArray[back] = e, currentSize++
- **dequeue**: front++, currentSize--
- Hàng đợi này chỉ chứa được 10 phần tử → nhanh chóng đầy ! → nhưng thực tế hàng đợi thường chỉ cần nhỏ nếu các thao tác enqueue và dequeue xảy ra thường xuyên
- Dù vậy, sau 10 lần enqueue, back ở vị trí cuối cùng → không thể enqueue thêm → giải pháp mảng vòng tròn !

Mảng vòng tròn

Trạng thái ban đầu

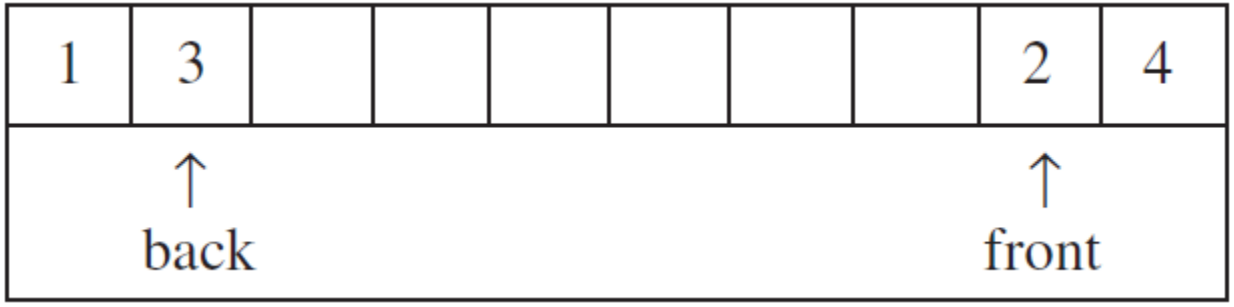


Sau **enqueue(1)**

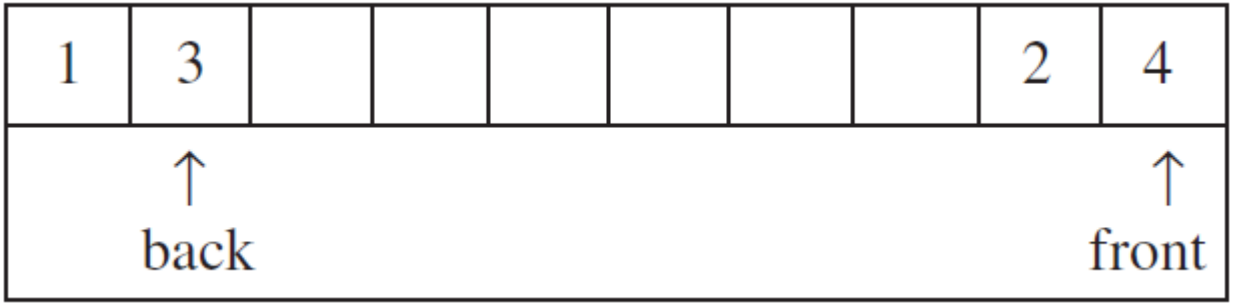


Mảng vòng tròn

Sau **enqueue(3)**

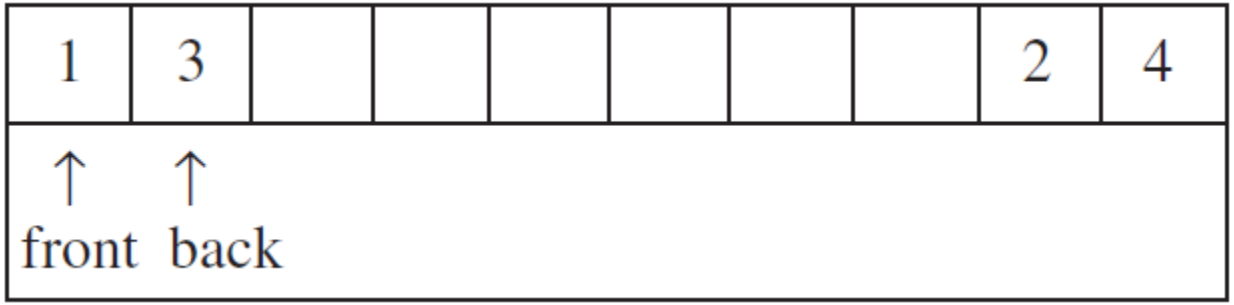


Sau **dequeue** (trả về 2)

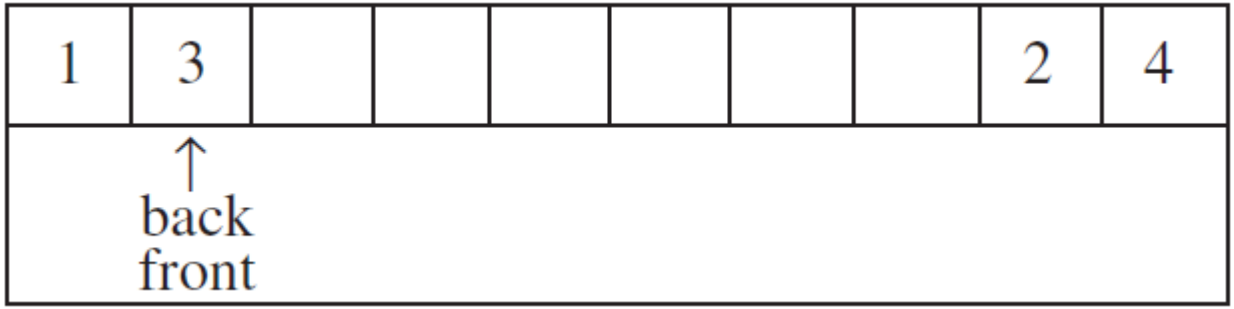


Mảng vòng tròn

Sau **dequeue** (trả về 4)

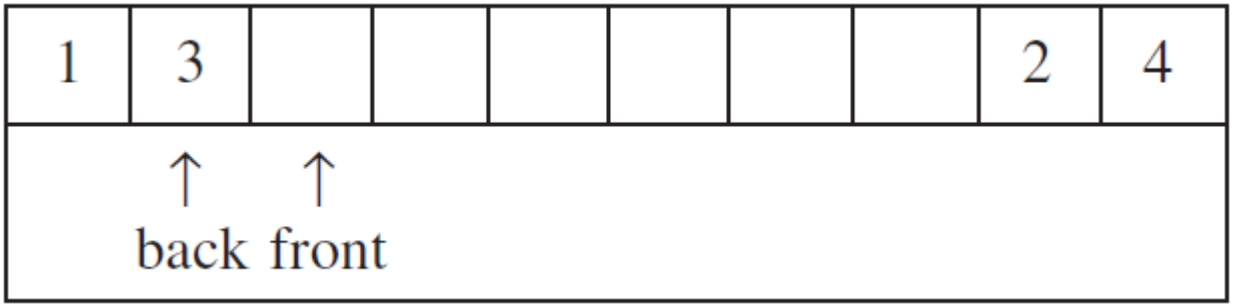


Sau **dequeue** (trả về 1)



Mảng vòng tròn

Sau **dequeue** (trả về 3)



Chú ý: Khi hàng đợi rỗng thì currentSize = 0

Một số ứng dụng của hàng đợi

- Xếp các tác vụ in ấn vào hàng đợi khi chúng được gửi tới cùng một máy in
- Xếp các cuộc gọi tới tổng đài vào hàng đợi khi tất cả các nhân viên trực đều bận
- Xếp các gói tin gửi từ nguồn tới đích (trong mạng máy tính) vào hàng đợi để chờ xử lý