Ngăn xếp và Hàng đợi (Stacks and Queues)

Bài giảng môn Cấu trúc dữ liệu và giải thuật

Khoa Công nghệ thông tin
Trường Đại học Thủy Lợi

Nội dung

1. Ngăn xếp
2. Hàng đợi

1. Ngăn xếp

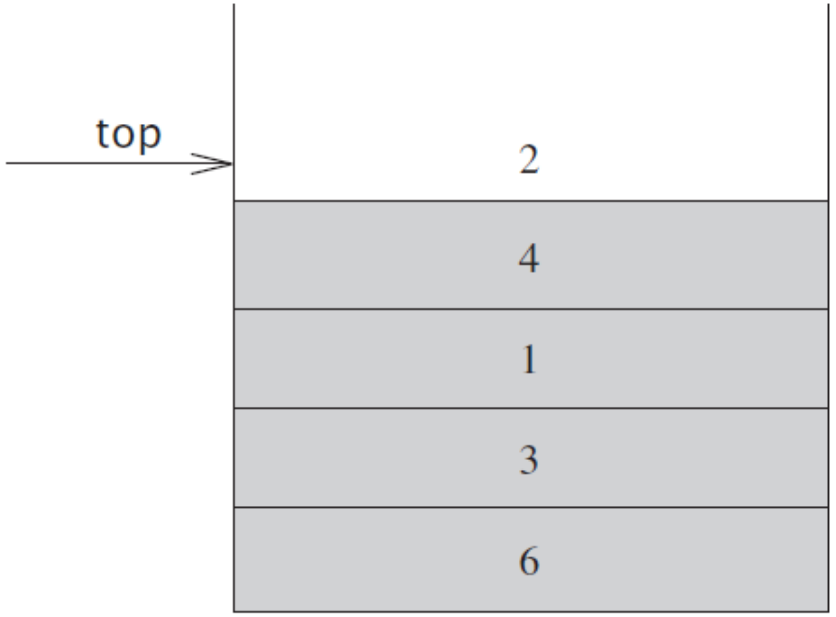
Ngăn xếp

- Một danh sách các phần tử theo kiểu **vào sau ra trước**:

  LIFO (Last In First Out)

- Ba thao tác chính, xảy ra ở đỉnh ngăn xếp và chỉ mất thời gian O(1):
  - **push**: Thêm phần tử.
  - **pop**: Xóa phần tử.
  - **top**: Trả về phần tử.

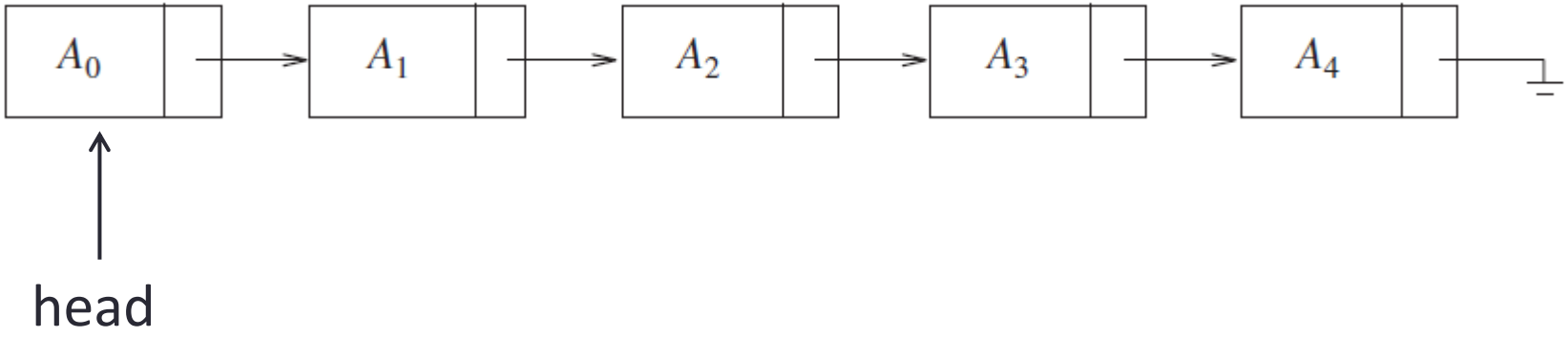
    (*pop và top có thể kết hợp thành một thao tác*)



- Các thao tác khác:
  - Lấy kích thước
  - Kiểm tra rỗng

Cài đặt ngăn xếp – cách 1

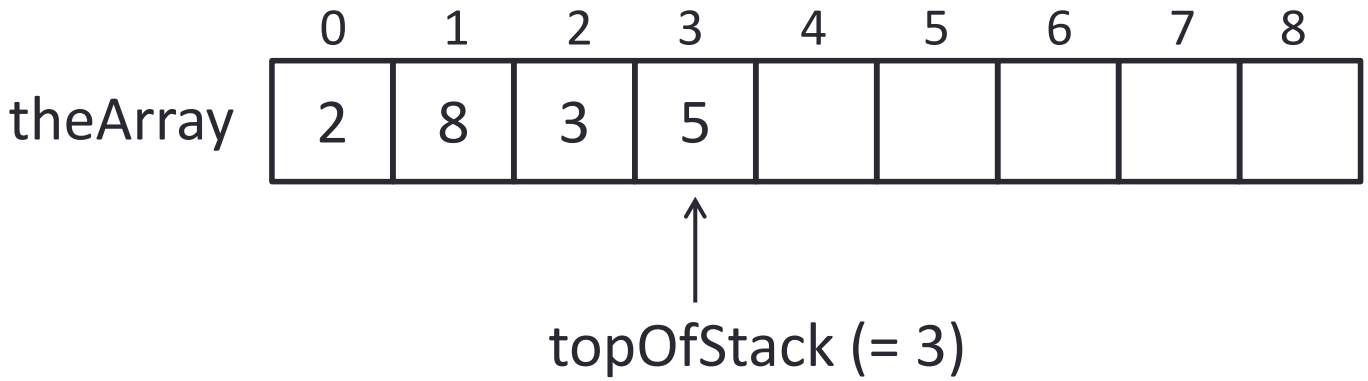
- Cài đặt bằng danh sách liên kết đơn:



- Cài đặt các thao tác:
  - **push(e)**: Gọi thao tác **pushFront(e)** của DSLK đơn.
  - **pop**: Gọi thao tác **popFront** của DSLK đơn.
  - **top**: Gọi thao tác **front** của DSLK đơn. lay gia tri

Cài đặt ngăn xếp – cách 2

- Cài đặt bằng mảng:



- Cài đặt các thao tác:
  - **push(e)**: topOfStack++, theArray[topOfStack] = e
  - **pop**: topOfStack--
  - **top**: Trả về theArray[topOfStack]

Một số ứng dụng của ngăn xếp

- Cân bằng thẻ (tag) trong trang HTML
- Định giá biểu thức hậu tố
- Ngăn xếp lời gọi hàm (xem trong sách)

Cân bằng thẻ trong trang HTML

Kiểm tra hai yêu cầu sau đây:

1. Mỗi thẻ mở phải có thẻ đóng tương ứng:

    something

2. Hai cặp thẻ khác nhau có thể lồng nhau nhưng không được bắt chéo nhau:

    something → OK

    something → Lỗi

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>
```

Ban đầu ngăn xếp rỗng

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>
```

Gặp thẻ mở, thêm nó vào ngăn xếp

Encounter an open tag, push it on the stack

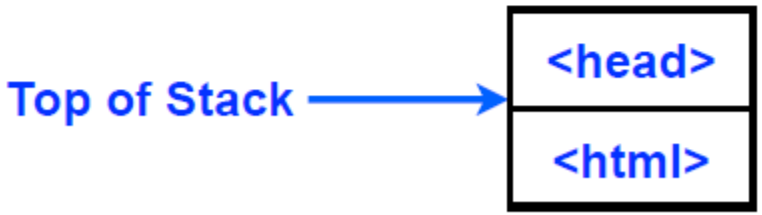
Matched Tags Example: HTML

- 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>
```

Gặp thẻ mở, thêm nó vào ngăn xếp

Encounter an open tag, push it on the stack



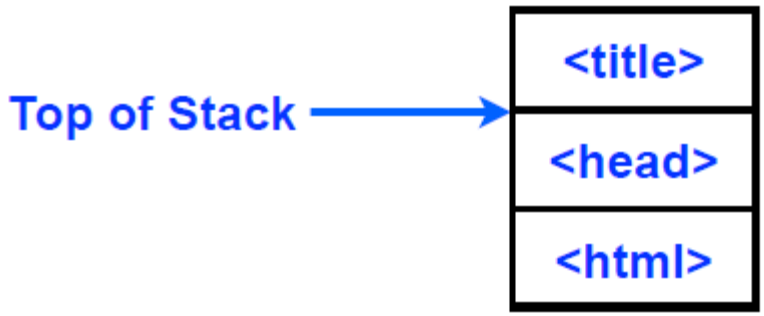
Matched Tags Example: HTML

- 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>
```

Gặp thẻ mở, thêm nó vào ngăn xếp

Encounter an open tag, push it on the stack



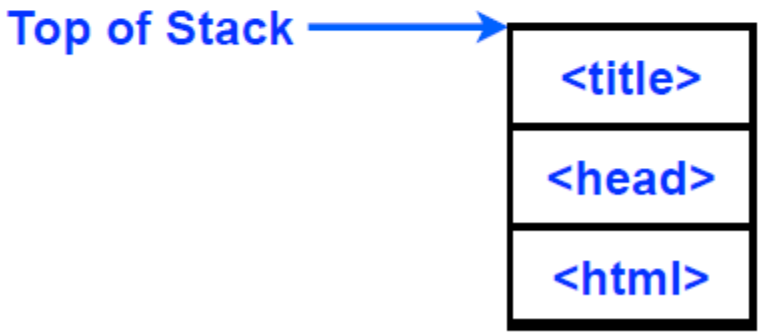
Matched Tags Example: HTML

- 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>
```

Gặp nội dung, không làm gì cả

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



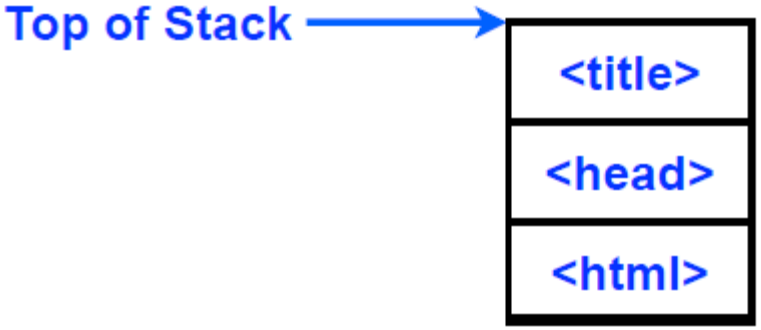
Matched Tags Example: HTML

- 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>
```

Gặp thẻ đóng, lấy một thẻ ra khỏi ngăn xếp, xem có khớp nhau không

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>
```

Gặp thẻ đóng, lấy một thẻ ra khỏi ngăn xếp, xem có khớp nhau không

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

Khi đã quét hết trang HTML và ngăn xếp rỗng, trang HTML là chuẩn!

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.

a+b=ab+

- Nếu viết lại biểu thức trên dưới dạng hậu tố (toán tử nằm sau các toán hạng của nó) rồi áp dụng ngăn xếp, ta chỉ cần tính tuần tự từ trái sang phải mà không cần quan tâm độ ưu tiên của các toán tử:

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

(*Có thể dùng ngăn xếp để chuyển đổi biểu thức trung tố thành biểu thức hậu tố → xem thêm trong sách*)

a*b+c+d*e=>ab*c+de*+

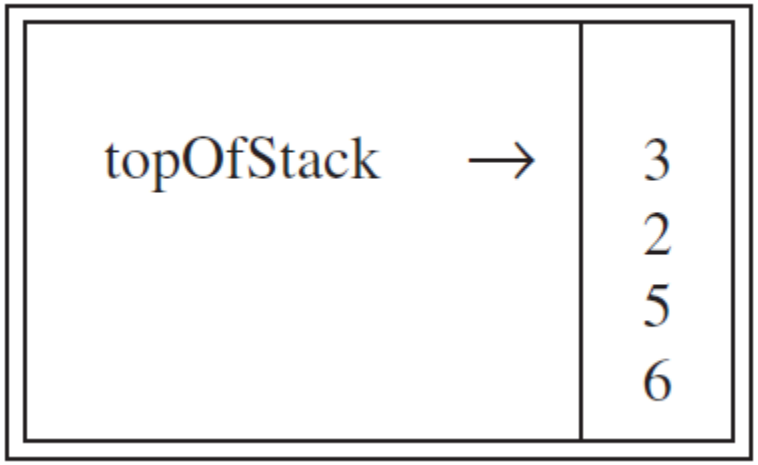
Định giá biểu thức hậu tố (tiếp)

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

- Gặp toán hạng:
  - Đặt nó vào ngăn xếp.
- Gặp toán tử:
  1. Lấy hai toán hạng ra khỏi ngăn xếp và áp dụng toán tử lên hai toán hạng đó.
  2. Đặt kết quả tính toán vào ngăn xếp.

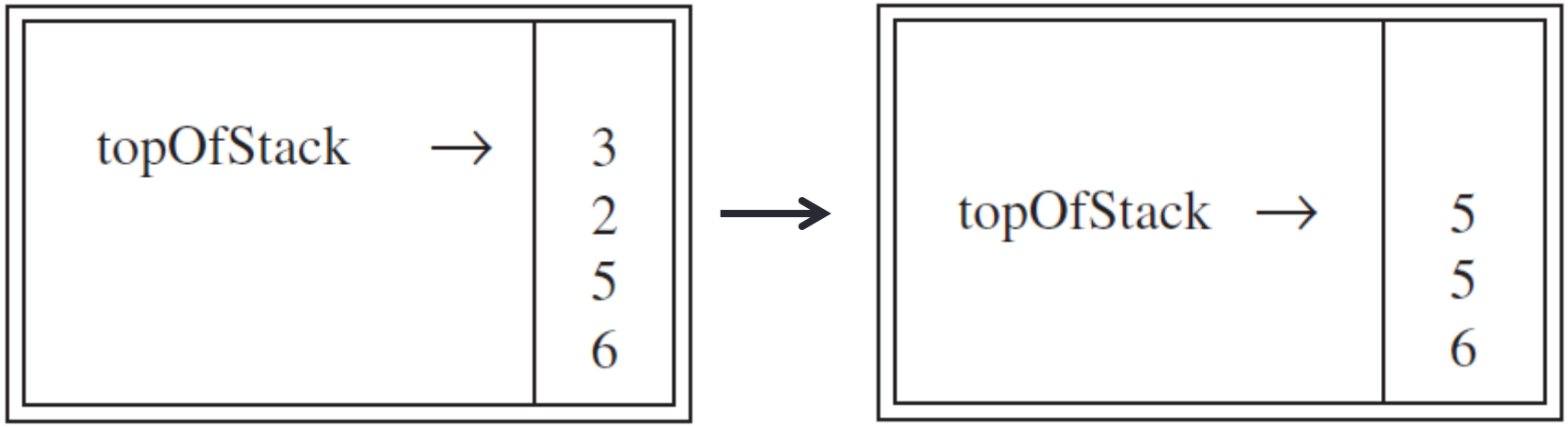
Ví dụ định giá biểu thức hậu tố

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



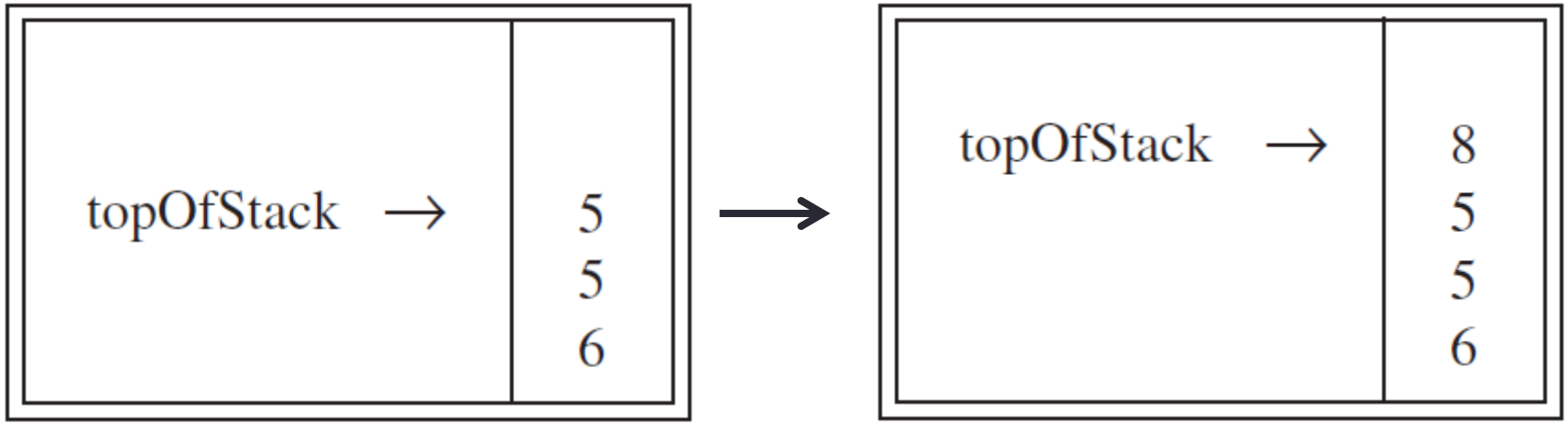
6 5 2 3 + 8 * + 3 + *

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



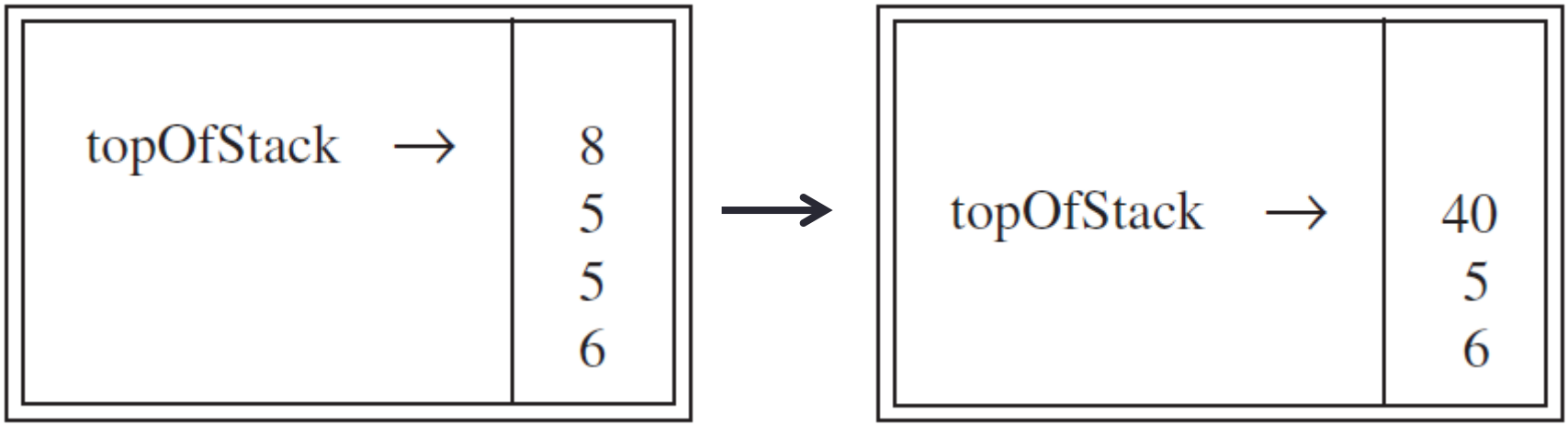
6 5 2 3 + 8 * + 3 + *

- Đặt 8 vào ngăn xếp.



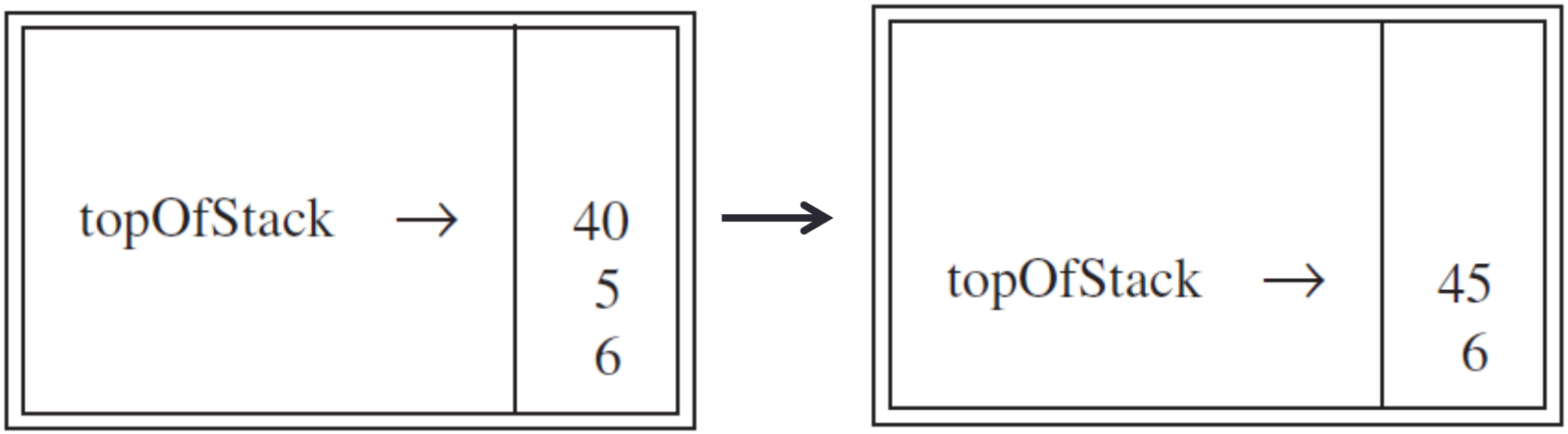
6 5 2 3 + 8 * + 3 + *

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



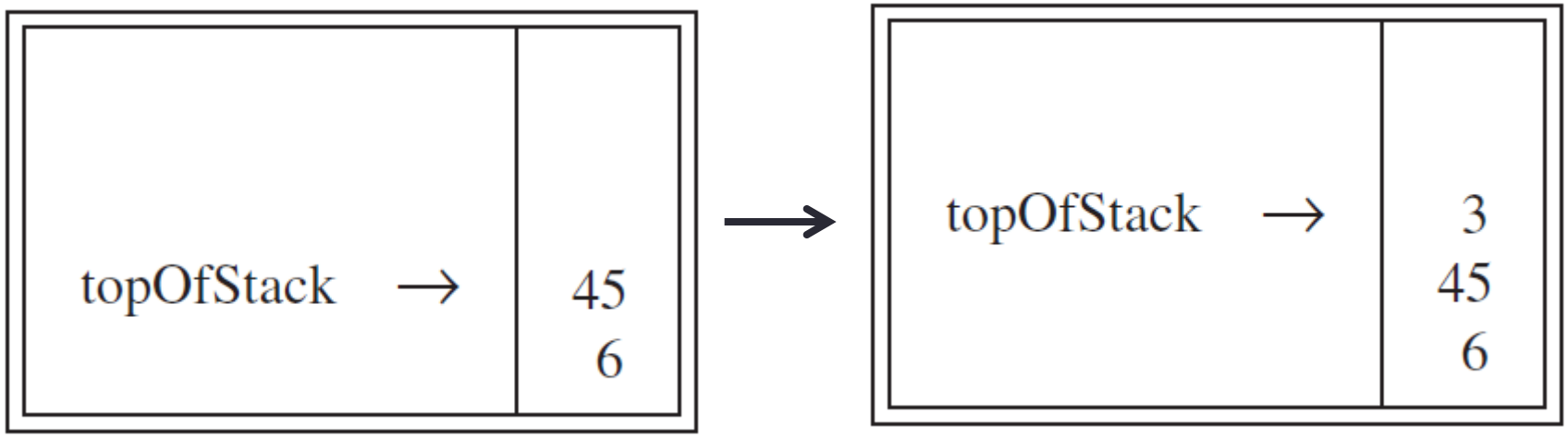
6 5 2 3 + 8 * + 3 + *

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



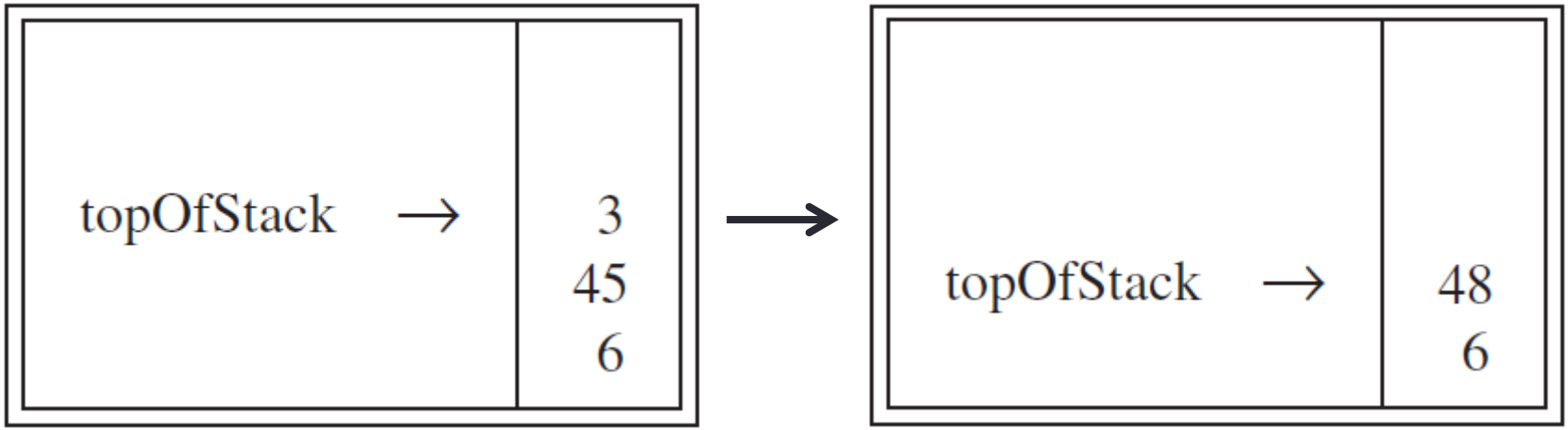
6 5 2 3 + 8 * + 3 + *

- Đặt 3 vào ngăn xếp.



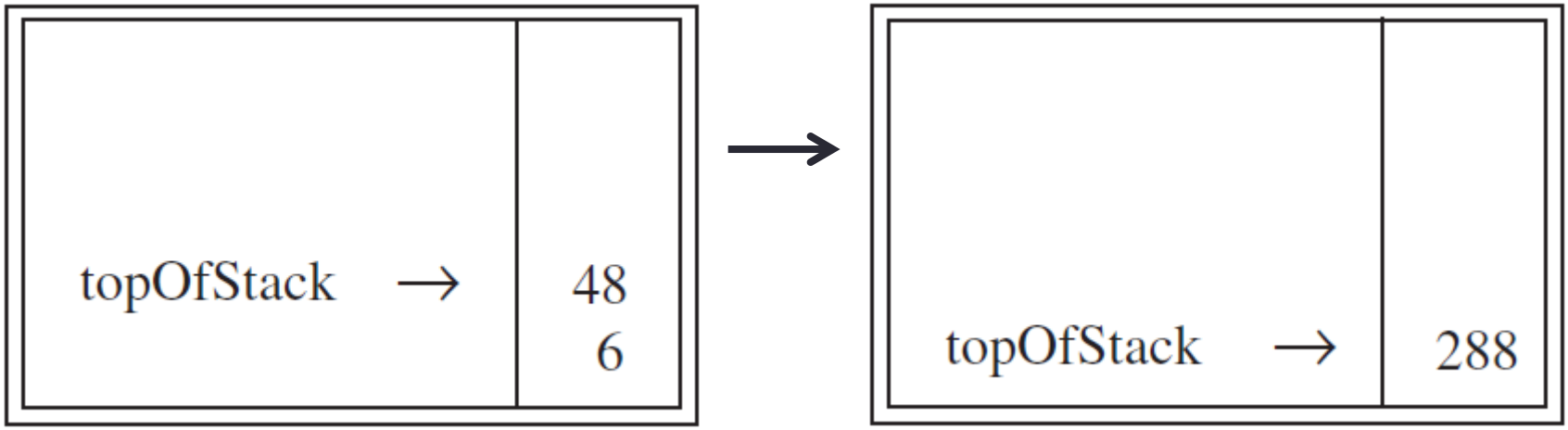
6 5 2 3 + 8 * + 3 + *

- 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.
- Khi đã quét hết biểu thức, giá trị duy nhất còn lại trong ngăn xếp (288) chính là giá trị tính được của biểu thức.



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

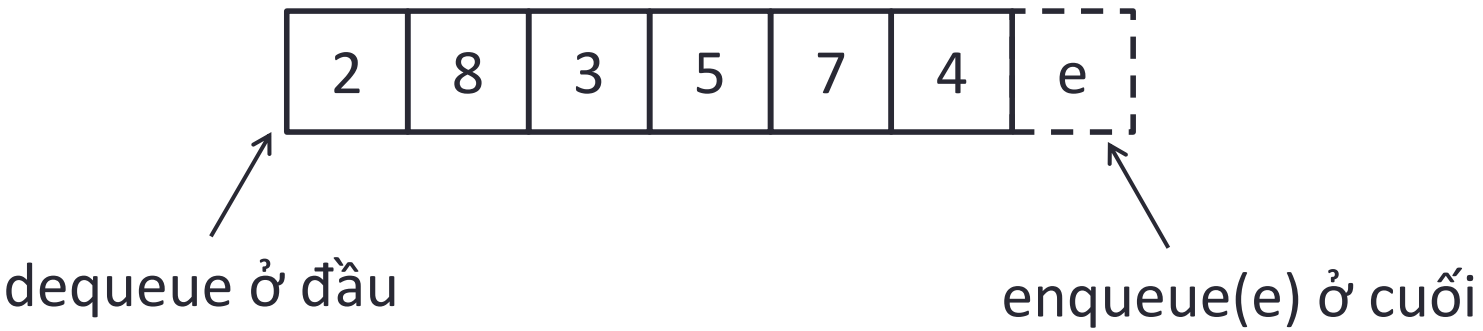
2. Hàng đợi

Hàng đợi

- Một danh sách các phần tử theo kiểu **vào trước ra trước**:

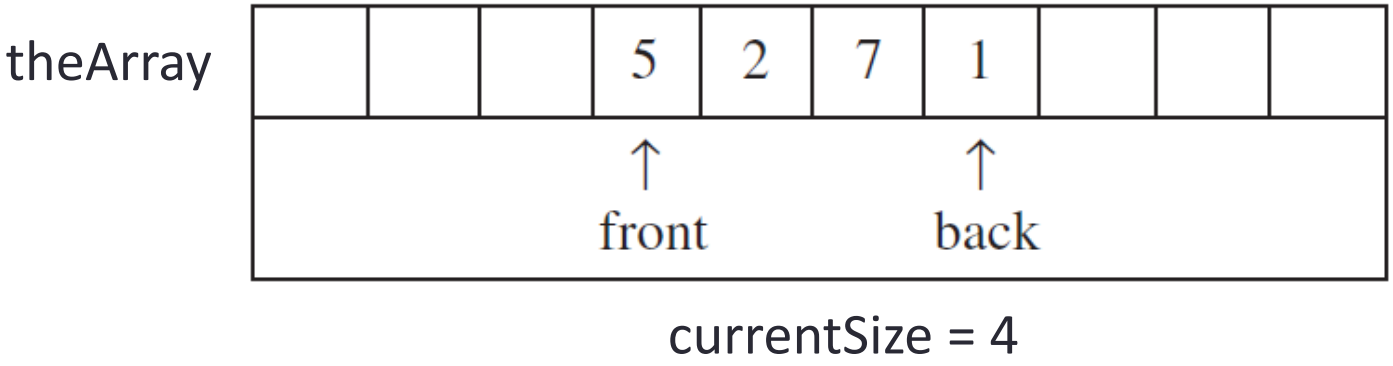
  FIFO (First In First Out)

- Hai thao tác chính, chỉ mất thời gian O(1):
  - **enqueue**: Chèn phần tử vào cuối danh sách.
  - **dequeue**: Xóa phần tử ở đầu danh sách.

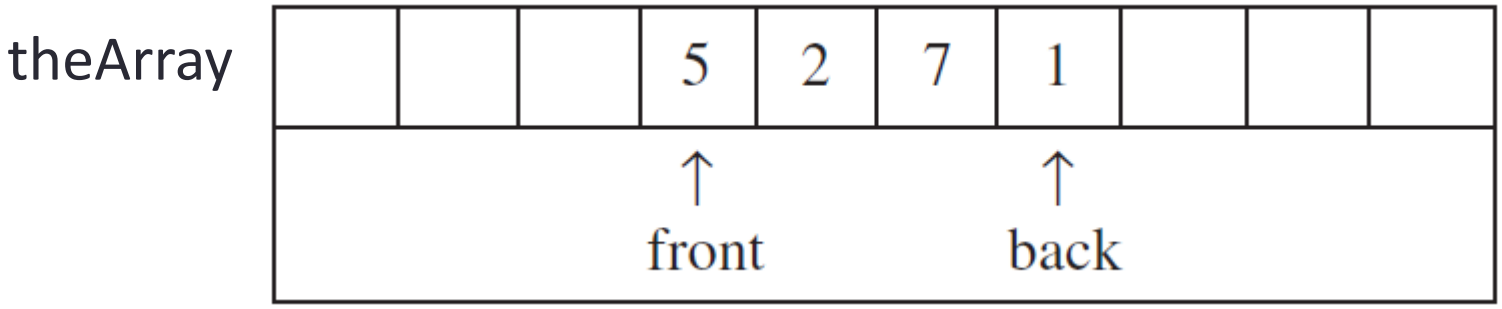


Cài đặt hàng đợi

- Như trường hợp ngăn xếp, ta có thể dùng mảng hoặc danh sách liên kết đơn để cài đặt hàng đợi.
- Ở đây, ta chỉ xem xét cách cài đặt dùng mảng:
  - **theArray** là mảng chứa các phần tử;
  - **currentSize** là số phần tử hiện có trong hàng đợi;
  - **front** và **back** giữ vị trí của phần tử đầu và cuối hàng đợi; vị trí nằm trong khoảng từ 0 đến chiều dài của mảng trừ đi 1.



Cài đặt hàng đợi dùng mảng

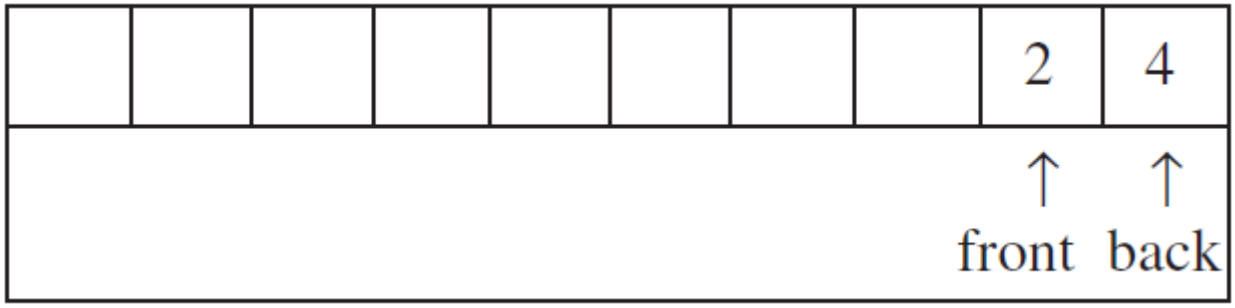


currentSize = 4

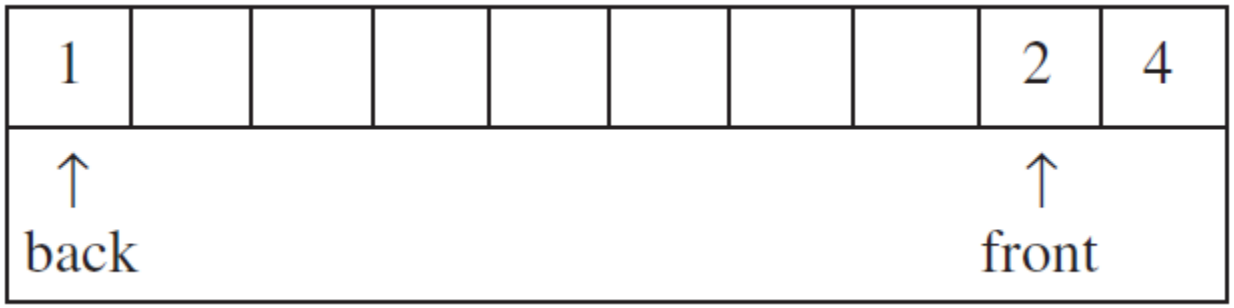
- **enqueue(e)**: back++, theArray[back] = e, currentSize++
- **dequeue**: front++, currentSize--
- Hàng đợi này chỉ chứa được 10 phần tử → nhanh chóng đầy?
  - Trên thực tế, hàng đợi thường chỉ cần nhỏ vì dequeue sẽ xảy ra đều đặn nhằm lấy từng phần tử ra khỏi hàng đợi để xử lý.
- Dù vậy, trong ví dụ trên, sau 10 lần enqueue, back ở vị trí cuối cùng thì sẽ không thể enqueue thêm → giải pháp **mảng vòng tròn**!

Mảng vòng tròn

Trạng thái hiện tại

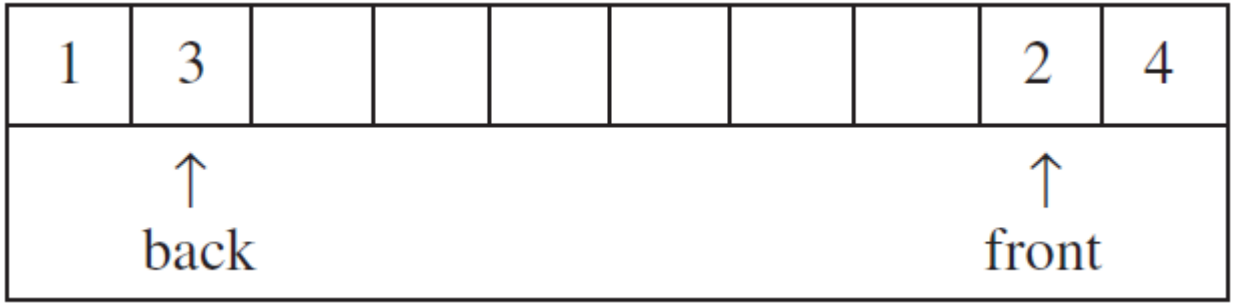


Sau **enqueue(1)**

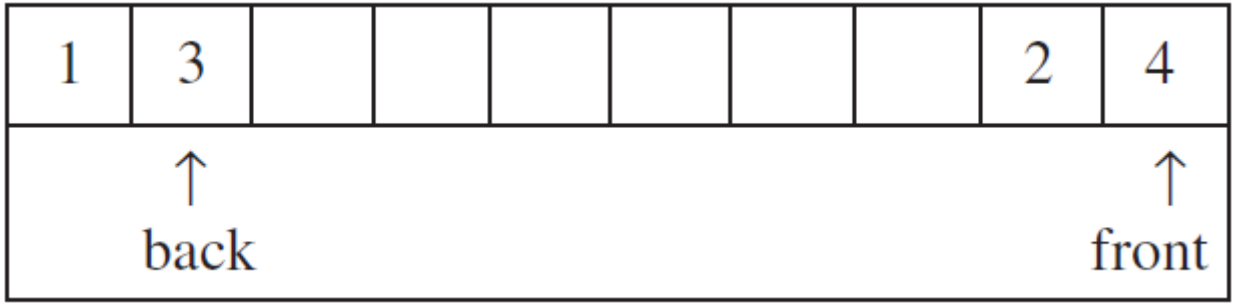


Mảng vòng tròn

Sau **enqueue(3)**

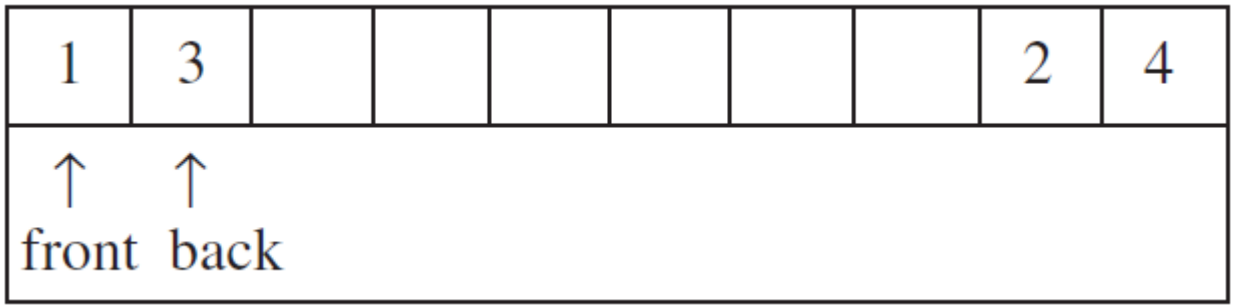


Sau **dequeue** (trả về 2)

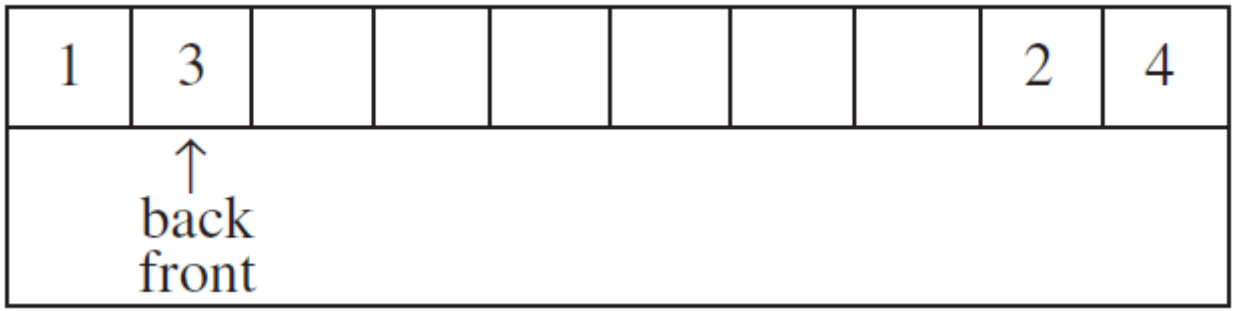


Mảng vòng tròn

Sau **dequeue** (trả về 4)

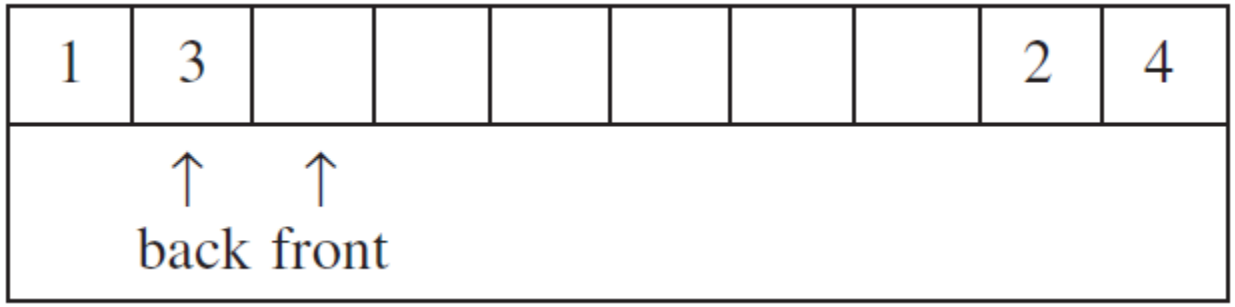


Sau **dequeue** (trả về 1)



Mảng vòng tròn

Sau **dequeue** (trả về 3)



Chú ý: Vì front có thể nằm trước hoặc sau back, nên không thể biết khi nào ***hàng đợi rỗng*** nếu chỉ căn cứ vào front và back → phải kiểm tra currentSize!

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.
  - Trên thực tế, ta có thể hủy một tác vụ in nằm giữa hàng đợi $\rightarrow$ không phải hàng đợi đúng nghĩa!
- Xếp các cuộc gọi tới tổng đài chăm sóc khách hàng vào hàng đợi khi tất cả các nhân viên tổng đài đều đang bận.
- Xếp các gói tin gửi từ nguồn tới đích (trong mạng máy tính) vào hàng đợi (ở đích) để chờ xử lý.

Bài tập

1. Nêu cách thực hiện các thao tác lấy kích thước và kiểm tra rỗng của ngăn xếp trong các trường hợp sau:

(a) Cài đặt ngăn xếp bằng danh sách liên kết đơn.

(b) Cài đặt ngăn xếp bằng mảng.

2. Đề xuất cách thực hiện hai ngăn xếp dùng chung một mảng.

3. Đề xuất cấu trúc dữ liệu hỗ trợ hai thao tác push và pop, như trong trường hợp ngăn xếp, nhưng hỗ trợ thêm thao tác **findMin** trả về phần tử nhỏ nhất trong thời gian O(1).

Bài tập

4. Xét một hàng đợi kiểu mảng vòng tròn. Hãy chỉ ra dãy thao tác enqueue và dequeue phải thực hiện để chuyển hàng đợi từ trạng thái 1 sang trạng thái 2:

5. Đề xuất cách thực hiện hàng đợi dùng danh sách liên kết đơn.