Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

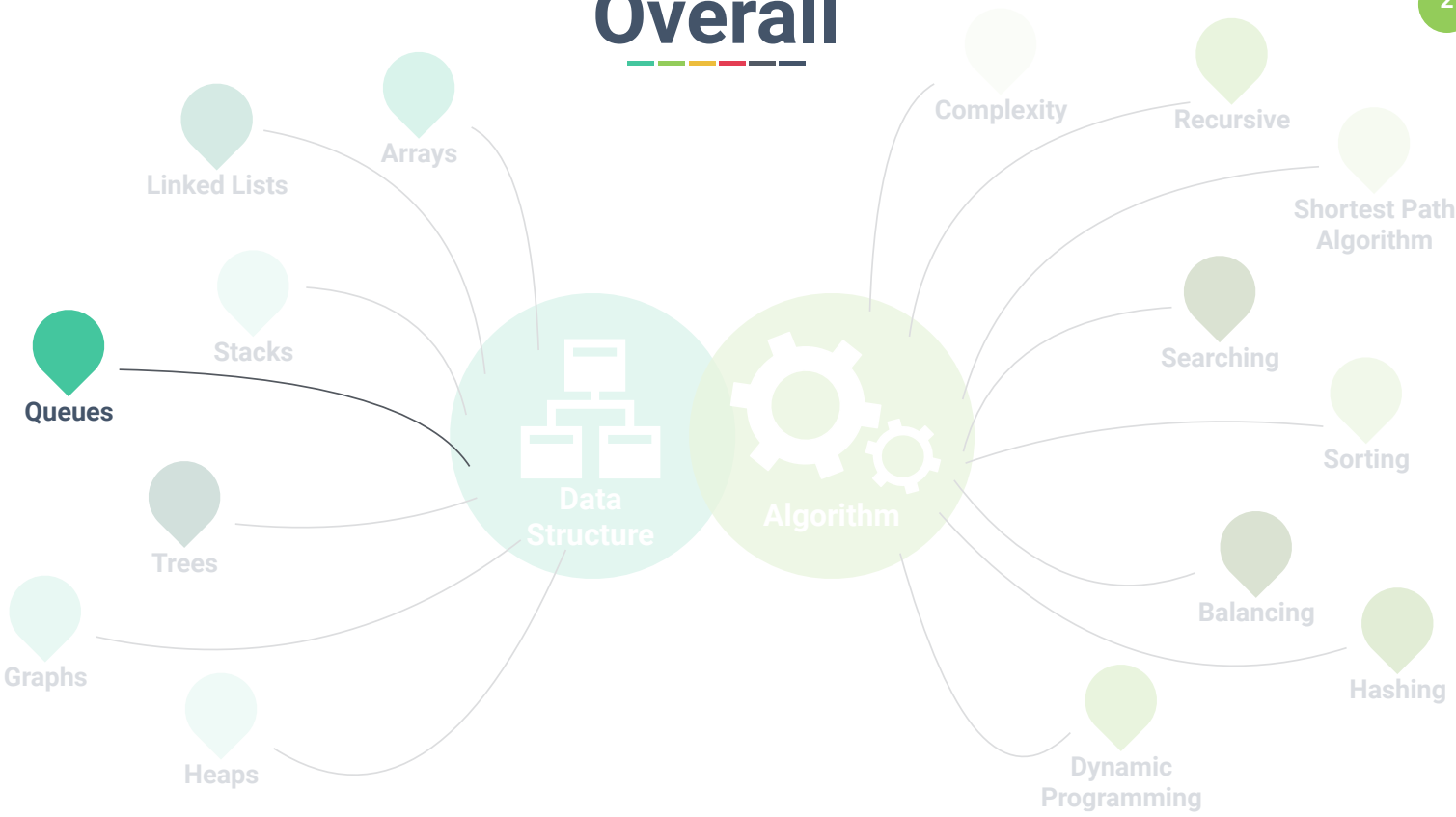
# Chapter 4: Queues

Dr. Sirasit Lochanachit

Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

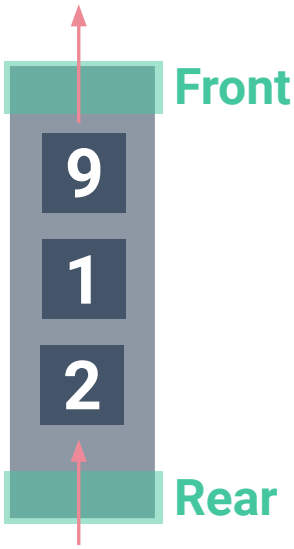


## Overall



# Queues






Queue is a collection, which keeps objects in a sequence, that are inserted and removed according to the **first-in first-out** (FIFO) principle [1].

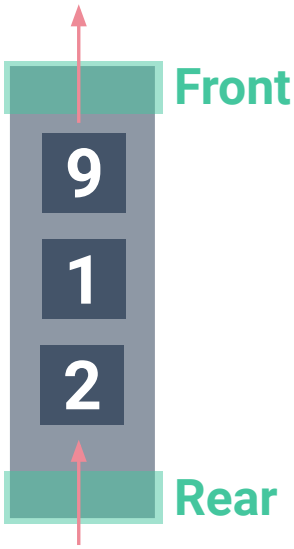
- Queue elements can be inserted at any time, but only the element that has been in the queue the longest can be removed at any time.

[1] Michael T. Goodrich et al., Data Structures and Algorithms in Python, 2013

Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

# Queues






- Close “cousin” of the stack.
- Elements enter a queue at the back.
- Elements are removed from the front.

[1] Michael T. Goodrich et al., Data Structures and Algorithms in Python, 2013

# Queues

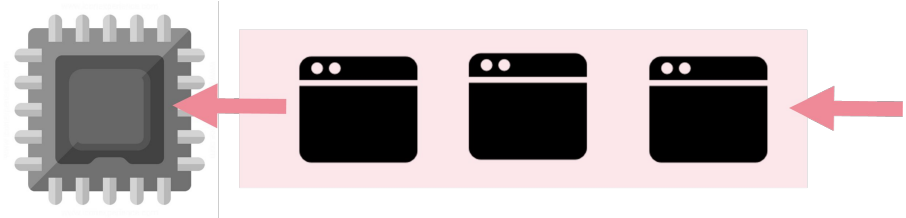
Real-life examples of queue:



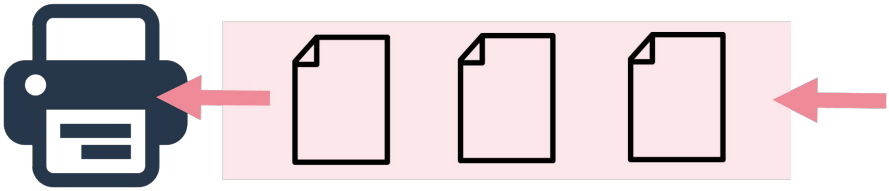


# Queues

Examples of queue:



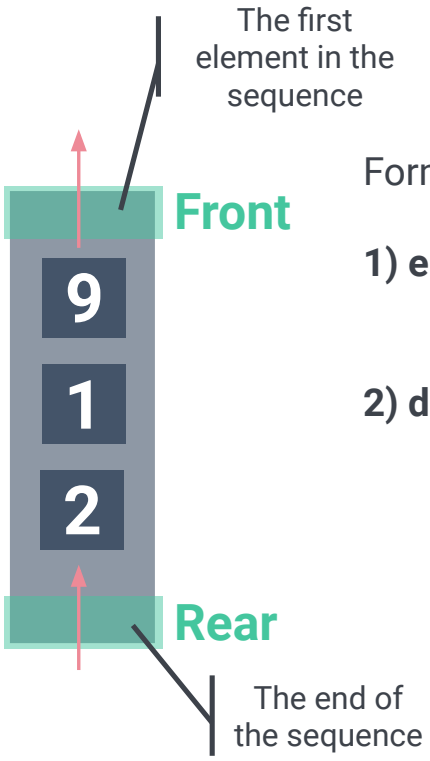
(i) a line of processes waiting to enter the CPU.



(ii) a line of paper waiting to print out.

# Queue Methods






Formally, there are 2 main operations for queues:

**1) enqueue(*e*)**

Insert element *e* at the back of the queue.

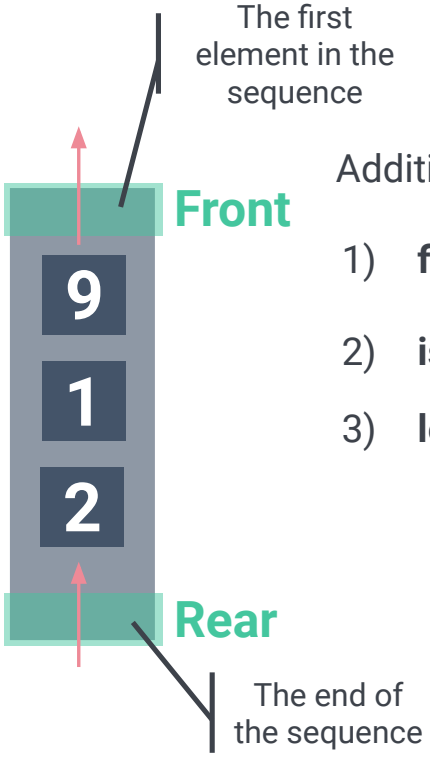
**2) dequeue()**

Remove and return the first element of the queue; return error if the queue is empty.

Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

# Queue Methods

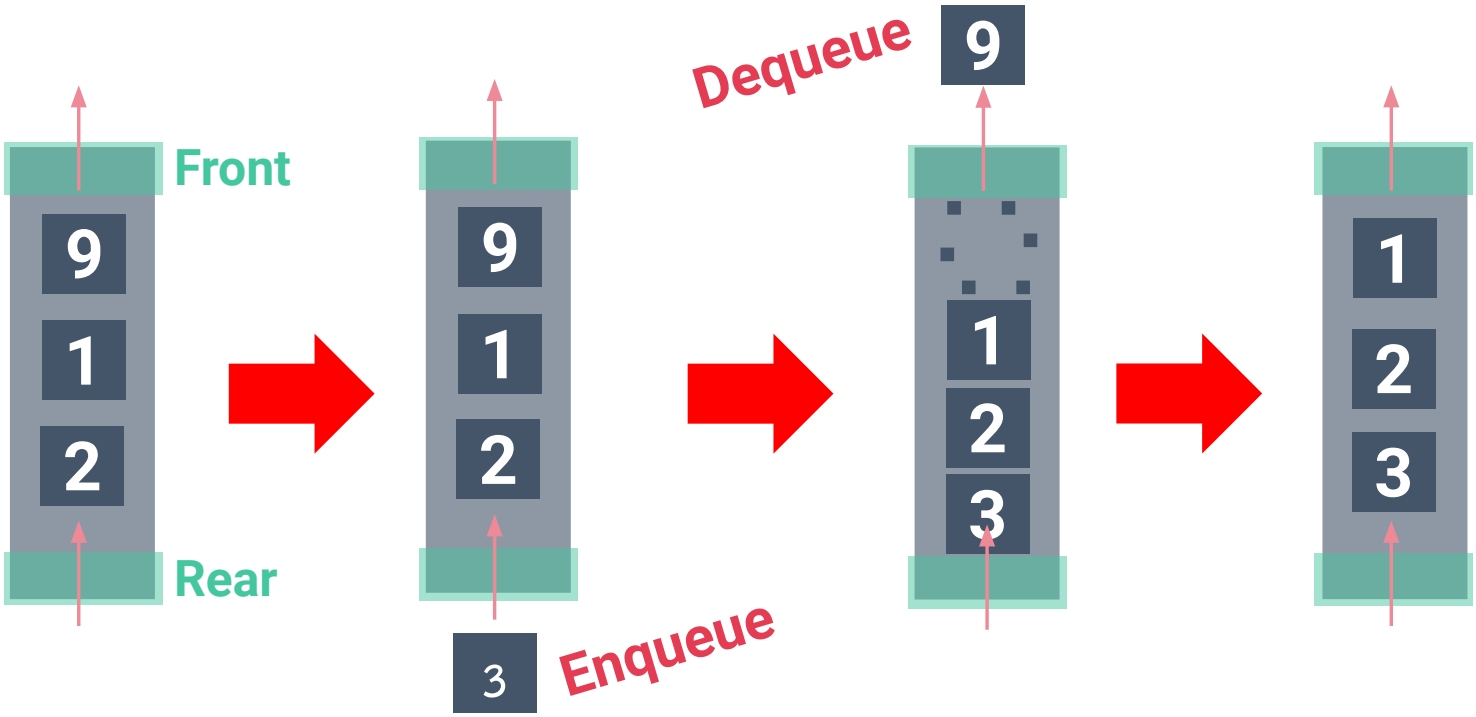





Additional operations for queues:

1) **first()** = Return the first element of the queue without removing it.
2) **is_empty()** = Check whether a queue is empty
3) **len(Q)** = Return the number of elements in a queue Q

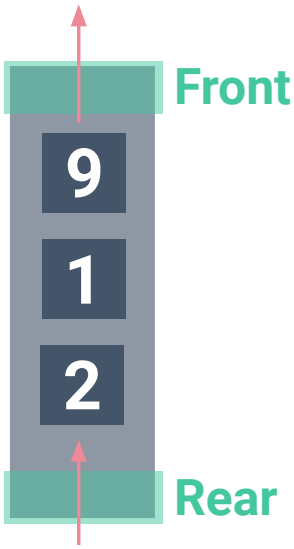
# Queues








# Stacks




**How to Implement a Queue?**

Array!!

# Operation Example

| Operation | Return Value | Queue [first, ..., last] |
|---|---|---|
| Q.enqueue(9) | - | 9 |
| Q.enqueue(5) | - | 9, 5 |
| Q.first() | 9 | 9, 5 |
| Q.enqueue(2) | - | 9, 5, 2 |
| Q.dequeue() | 9 | 5, 2 |
| Q.is_empty() | False | 5, 2 |
| len(Q) | 2 | 5, 2 |

Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang



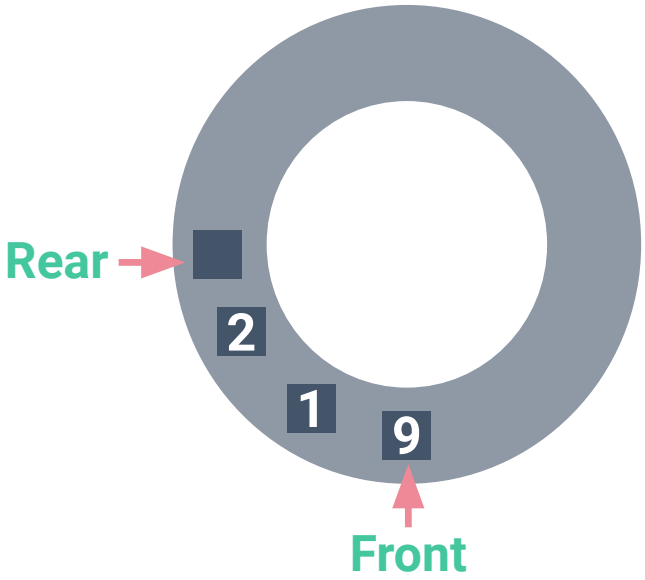
# Asymptotic Performance

| Operation | Running Time |
|---|---|
| Q.enqueue(e) | $O(1)$ |
| Q.dequeue() | $O(n)$ or $O(1)$ |
| Q.first() | $O(1)$ |
| Q.is_empty() | $O(1)$ |
| len(Q) | $O(1)$ |

# Types of Queue

- **Queue**


- **Circular Queue**





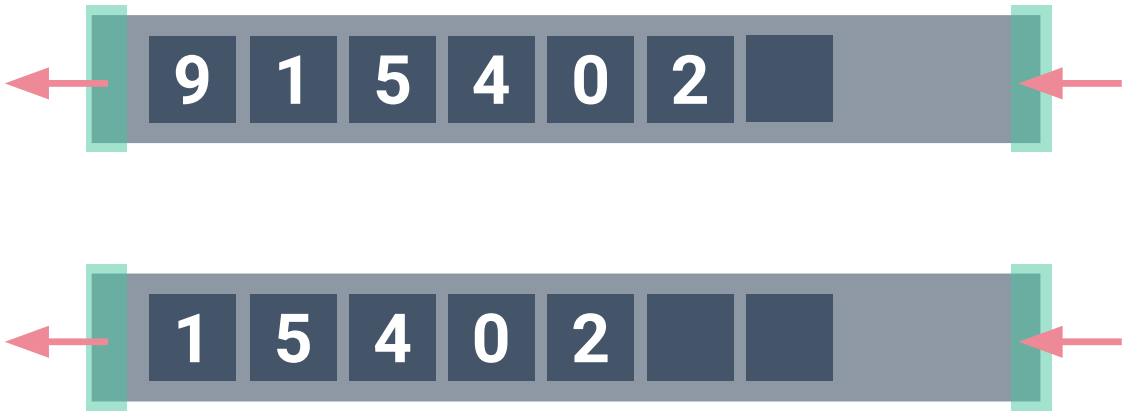



# Simple Queue


Suppose a Python list is created.

- Enqueue can be achieved by calling append(e), adding item at the end of the list.
- Dequeue can be executed by using pop(0) to remove the first element. *O*(?) - inefficient for large number of elements, why?

# Simple Queue

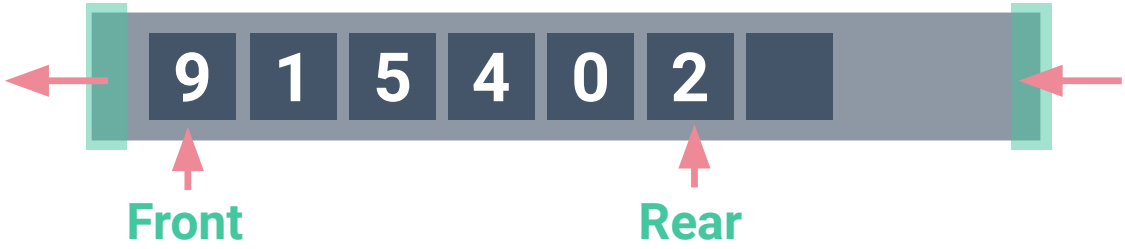



- pop(0) requires a loop to shift all the elements to the left after removing the first element - O(n).





# Revised Queue



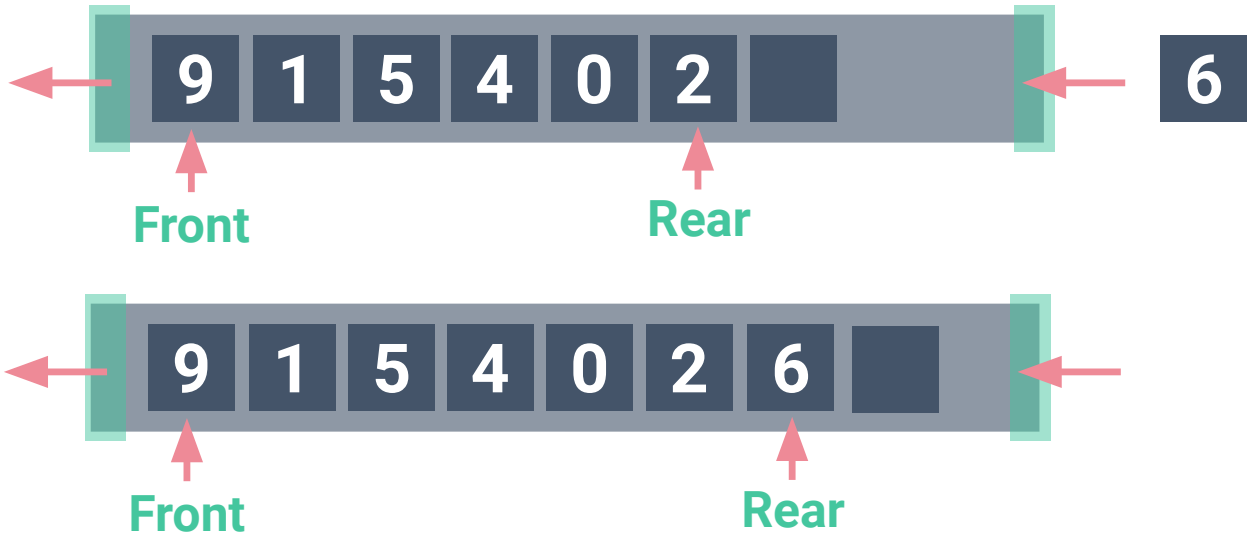

A better approach is to assign two main pointers to store indices that refer to:

the first element of the queue, which is called the ***Front***,

and the last element of the queue, which is known as the ***Rear***.

# Revised Queue

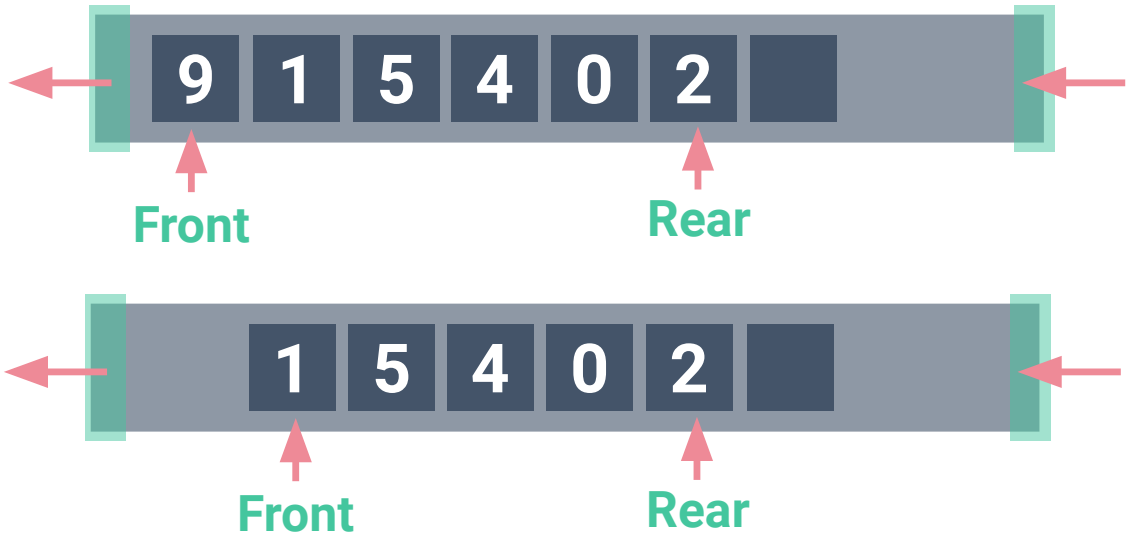
- **Enqueue**



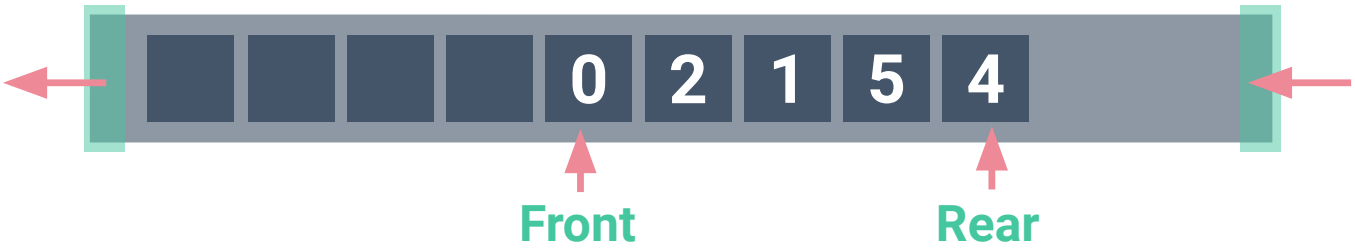



# Revised Queue

- **Dequeue**



# Revised Queue




Drawback:

If enqueue and dequeue is repeatedly executed, then these two pointers will keep shifting rightward.

The size of the list would grow to *m* where *m* is the total number of enqueue operations rather than the current number of elements in the queue.



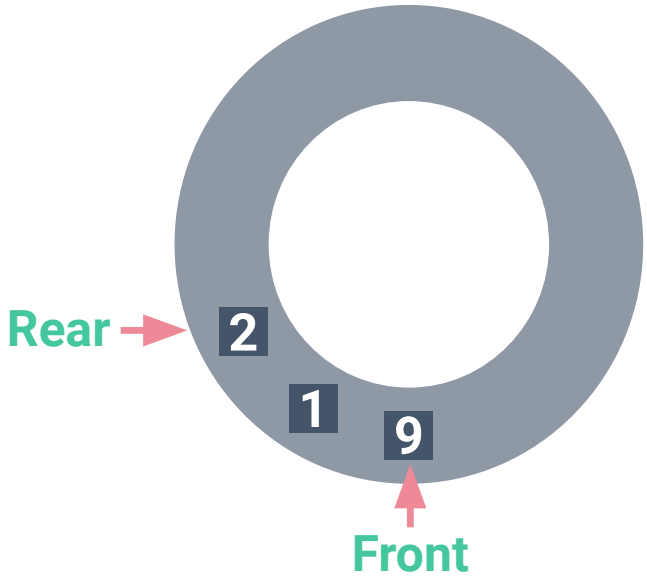


# Types of Queue

- Queues

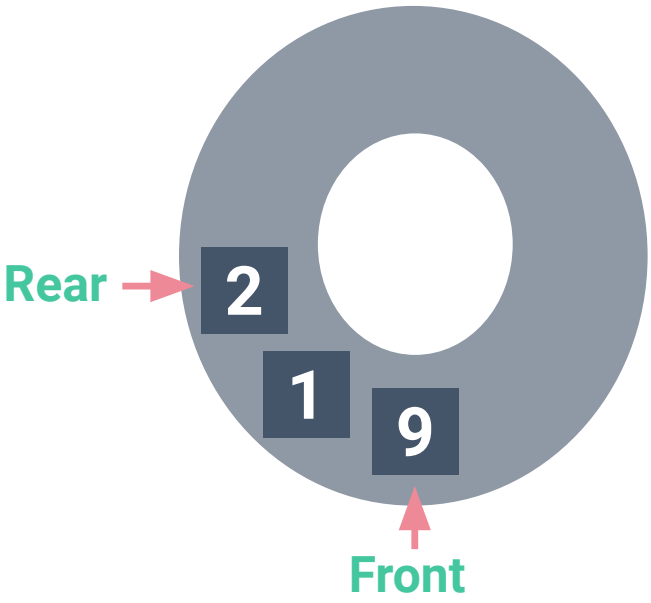



- Circular Queues



# Circular Queue






In addition to allowing the pointers to shift rightward, the elements of the queue are wrap around at the end of an array.

When new elements are enqueued at the end of the current queue, the indices are continued at index 0, 1, and so on.

Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

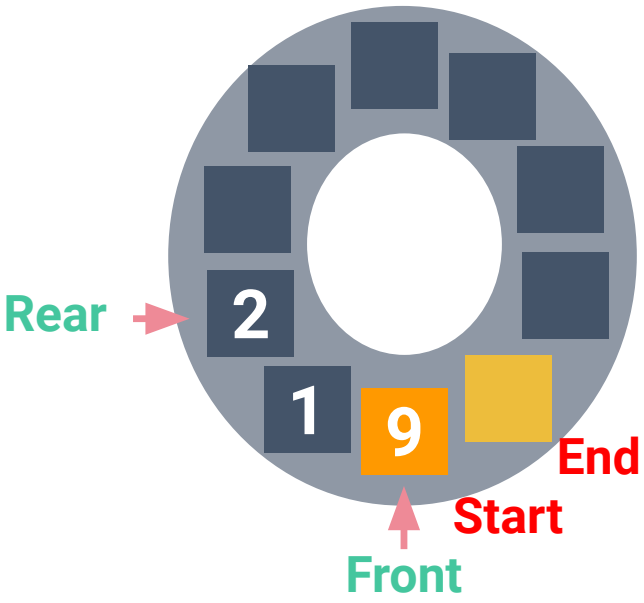
# Circular Queue




Suppose a queue of length 10,

the front index is at 0, the rear index is at 2.

# Circular Queue

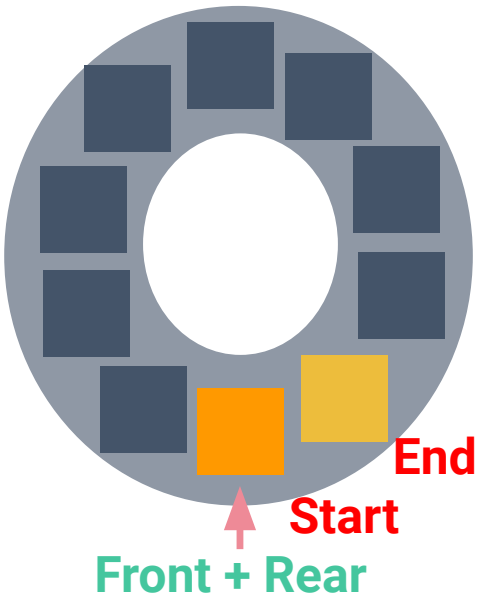




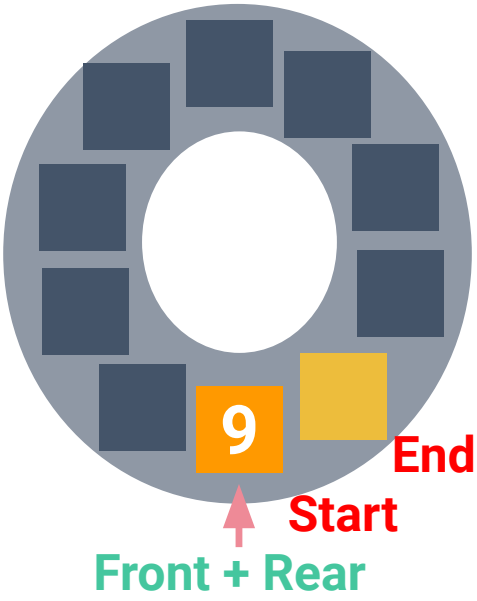



# Circular Queue

Empty Queue



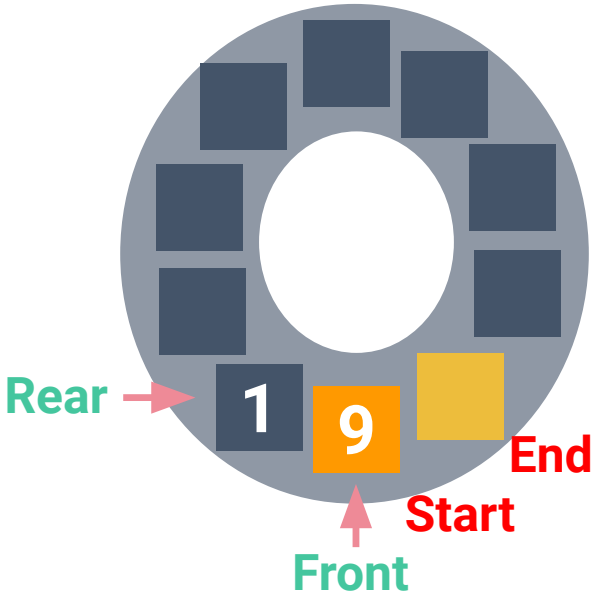
# Circular Queue



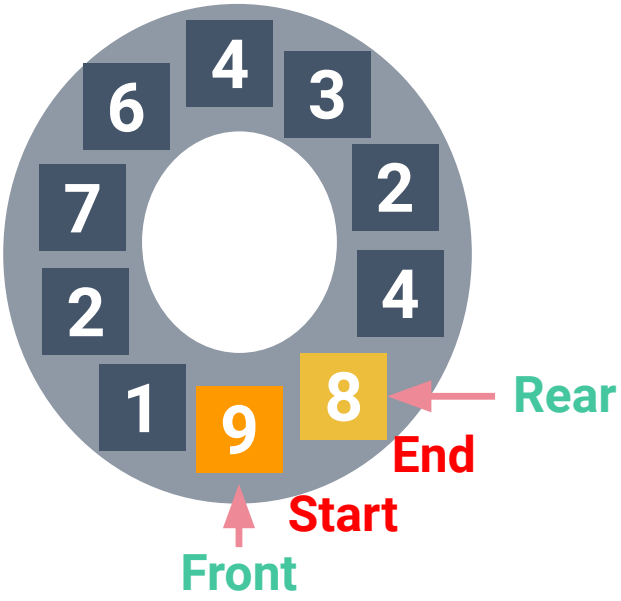

Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang



# Circular Queue



# Circular Queue

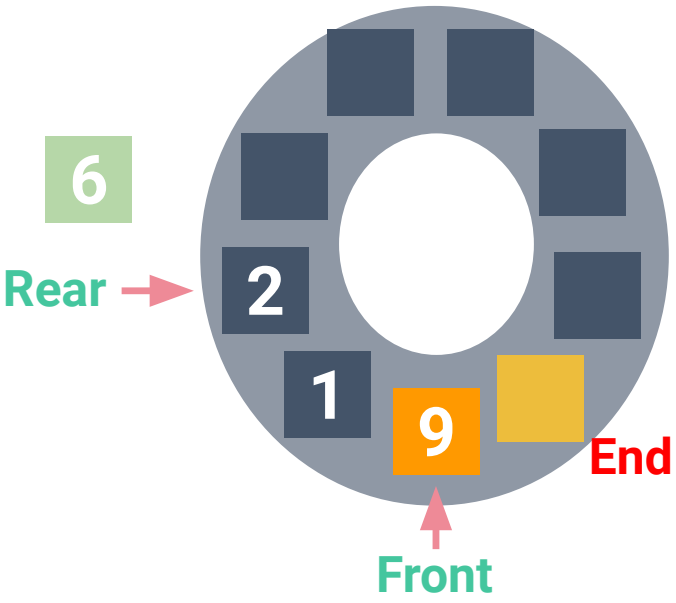






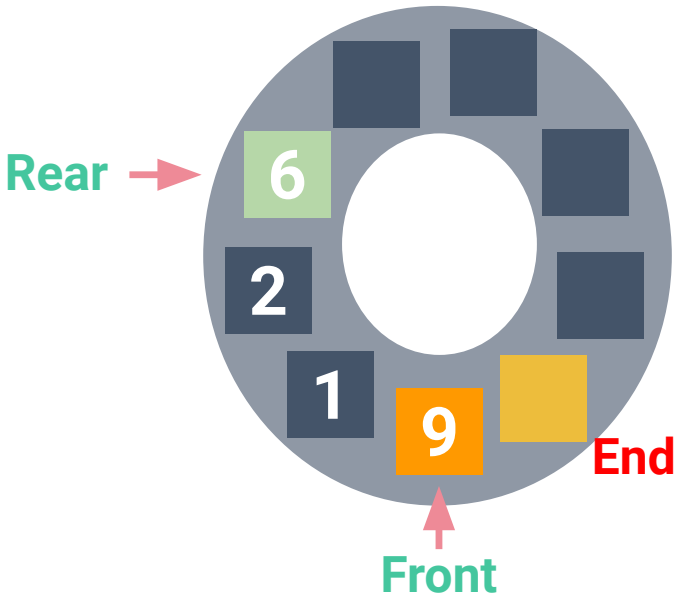

# Circular Queue

**Enqueue**



# Circular Queue

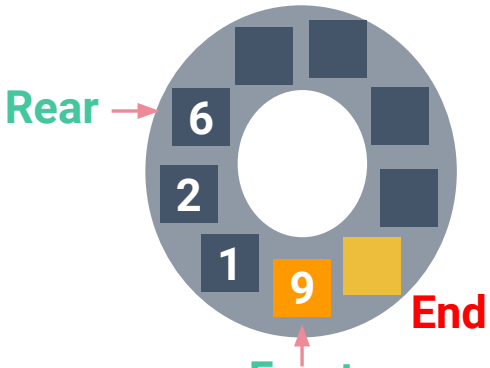
## Enqueue







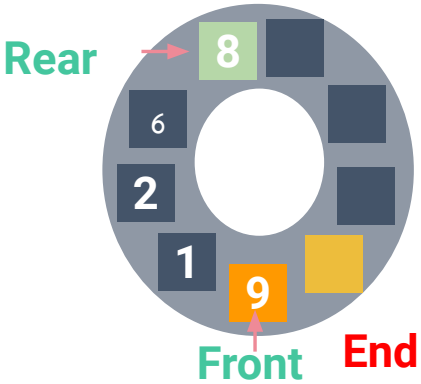

# Circular Queue in Python




When **enqueue**, the rear index is given by

$$r = (r + 1) \% N$$

where N is the array length.



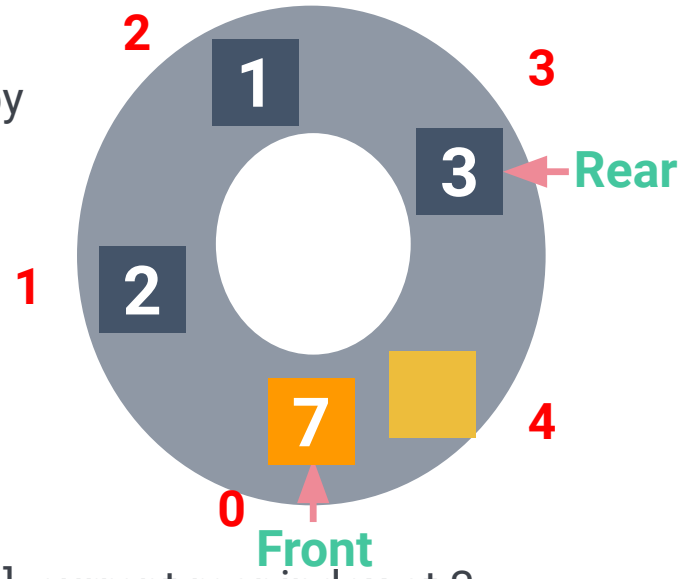

* Note: % denotes ***modulo*** operation in python which provides the remainder after division.

# Circular Queue in Python

When **enqueue**, the rear index is given by

r = (r + 1) % N

where N is the array length.




For example, given a list length of 5 [0-4], current rear index at 3.

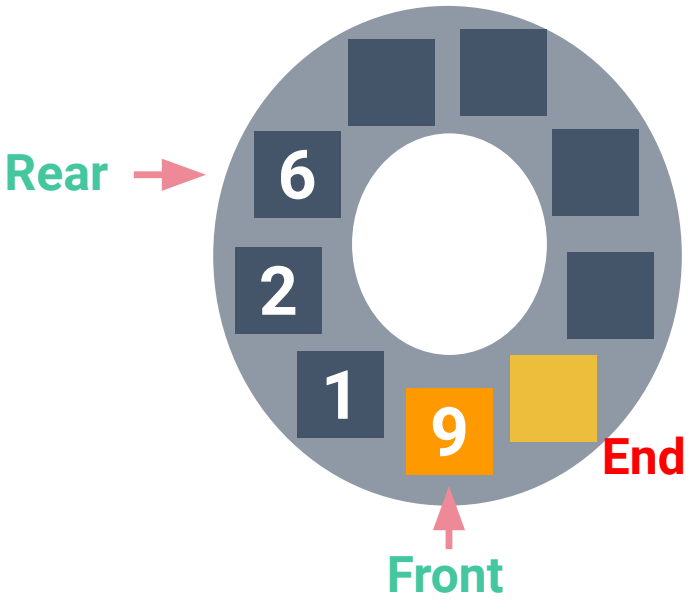
When an enqueue is called, the new rear index is (3+1) % 5 = 4.





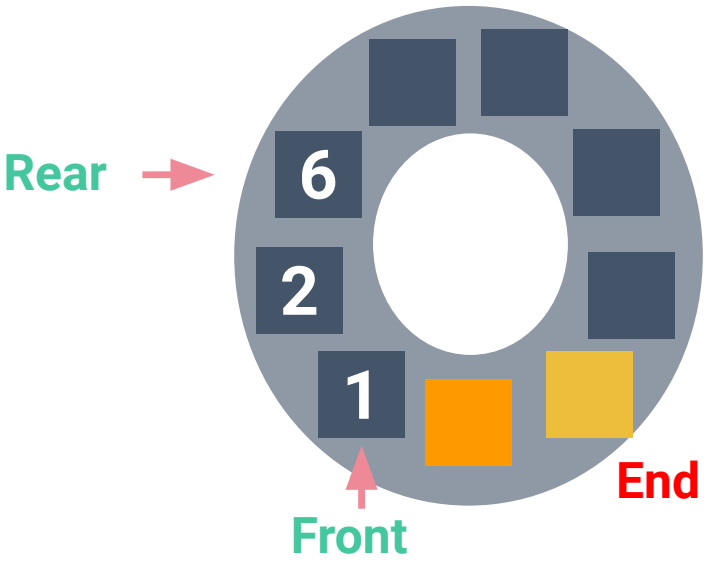
# Circular Queue

**Dequeue**



# Circular Queue

## Dequeue

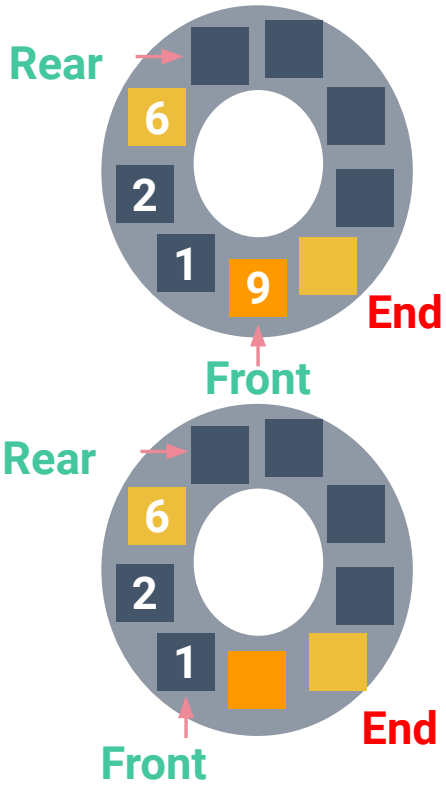



$O(1)$





# Circular Queue in Python




When **dequeue**, the front index is given by

$$f = (f + 1) \% N$$

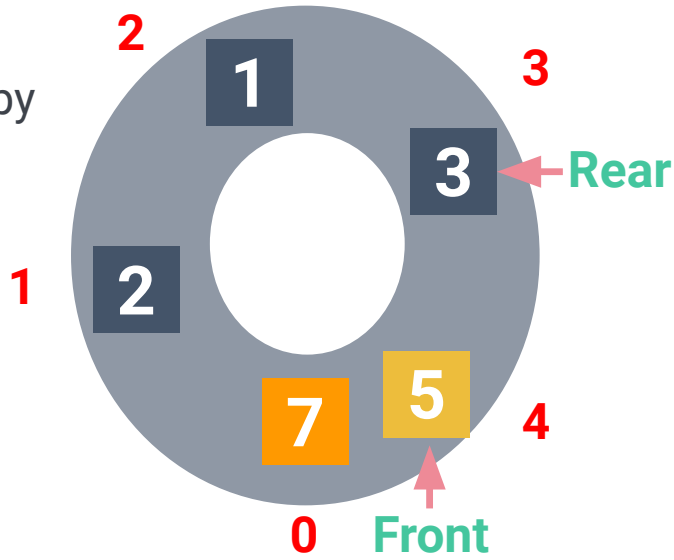
where N is the array length.

# Circular Queue in Python

When **dequeue**, the front index is given by

f = (f + 1) % N

where N is the array length.




For example, given a list length of 5 [0-4], current front index at 4.

When a dequeue is called, the new front index is (4+1) % 5 = 0.





# Extension of Queue ADT

Limitation of queue

- Insertion and deletion of items can only be done at the front and the rear respectively

**Double-ended queue (deque)** additionally allows:

- Add element at the front of the queue.
- Remove element at the back of the queue.

# Double-ended Queue



Main Operations for double-ended queues:

**1) add_first(*e*)**

Insert element *e* at the front of the queue.

**2) add_last(*e*)** - Same as enqueue(e)

Insert element *e* at the back of the queue.

**3) delete_first()** - Same as dequeue()

Remove and return the first element from the queue; return error if the queue is empty.

**4) delete_last()**

Remove and return the last element from the queue; return error if the queue is empty.

Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

# Double-ended Queue



Additional operations for double-ended queues:

1) **first()** = Return the first element of the queue without removing it.

2) **last()** = Return the last element of the queue without removing it.

# Individual Assignment



- Assignment#2: Stacks
- Due 09.00 am 25/08/2020.
- Submission
  - Email: sirasit@it.kmitl.ac.th
  - Paper: in classroom next week
- Can be either written by hand or typing.