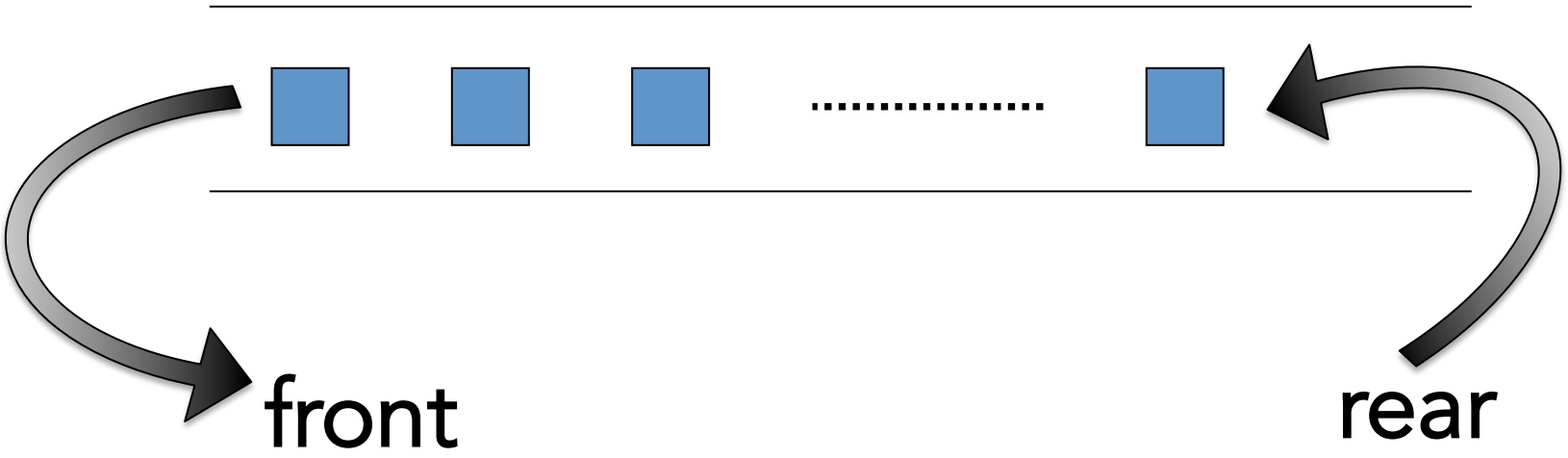
Lecture 8
Queues and Deques

Queues
First-in First-out

Main Operations

1. enqueue (object) at the rear
2. dequeue (object) from front



Auxiliary Operations

- object front():
- integer size():
- boolean empty():

Queue Interface in C++

```
template <typename E>
class Queue {
public:
  int size() const;
  bool empty() const;
  const E& front() const;
  void enqueue (const E& e);
  void dequeue()
};
```

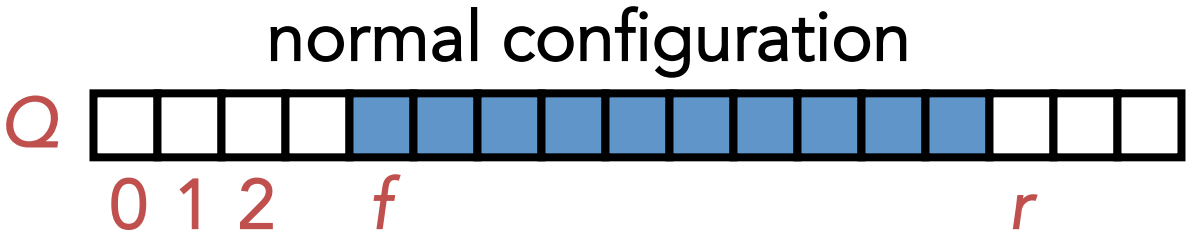
Operation	*Output*	*Queue*
enqueue(5)	–	(5)
enqueue(3)	–	(5, 3)
enqueue(4)	–	(5, 3, 4)
dequeue()	–	(3, 4)
enqueue(7)	–	(3, *4*, *7*)
dequeue()	–	(4, 7)
front()	*4*	(4, 7)
size()	2	(4, 7)
dequeue()	–	(7)
dequeue()	–	()
empty()	*true*	()

Applications

- Direct applications
  - shared resources (e.g., printer)
  - multi-threaded programming
- Indirect applications
  - auxiliary data structure for algorithms

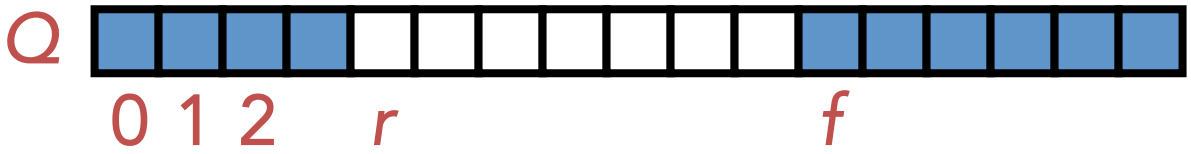
Queue Implementation

Array-based Implementation



- *f* index of the front element
- *r* index immediately past the rear element
- *n* number of items in the queue

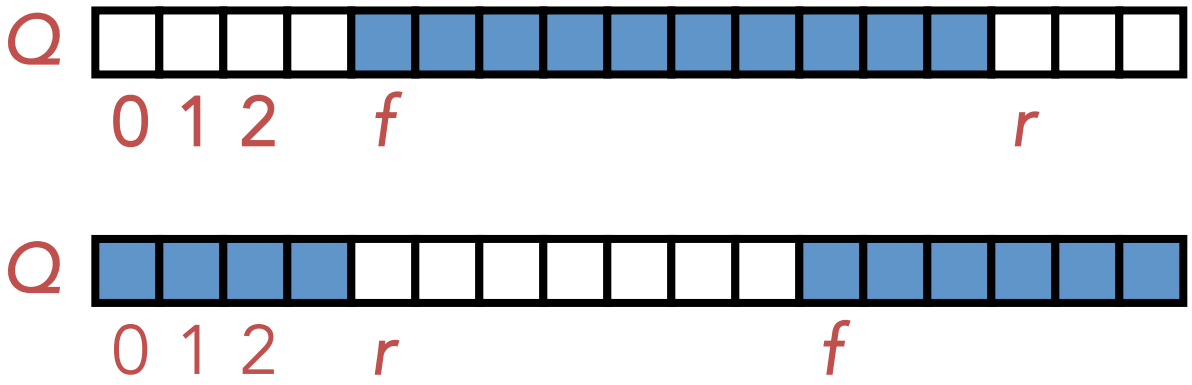
Wrapped-around Configuration



- *f* index of the front element
- *r* index immediately past the rear element
- *n* number of items in the queue

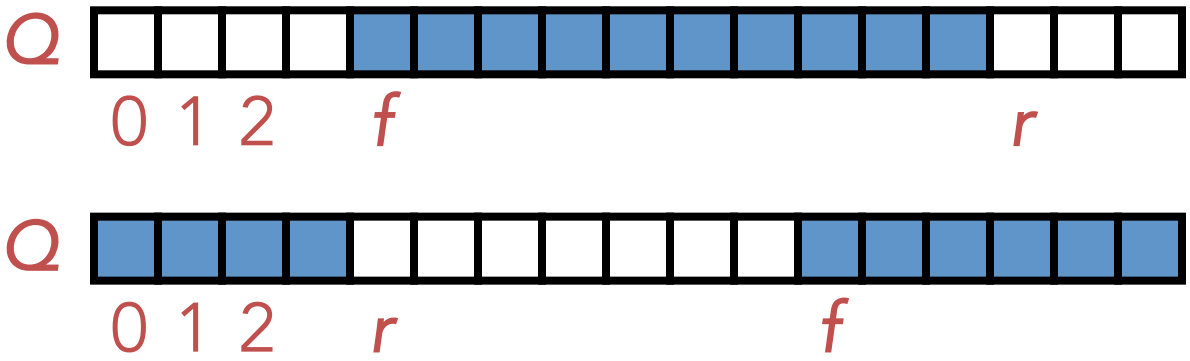
enqueue()

Algorithm *enqueue*(*e*)
    $Q[r] \leftarrow e$
    $r \leftarrow (r + 1) \bmod N$
    $n \leftarrow n + 1$



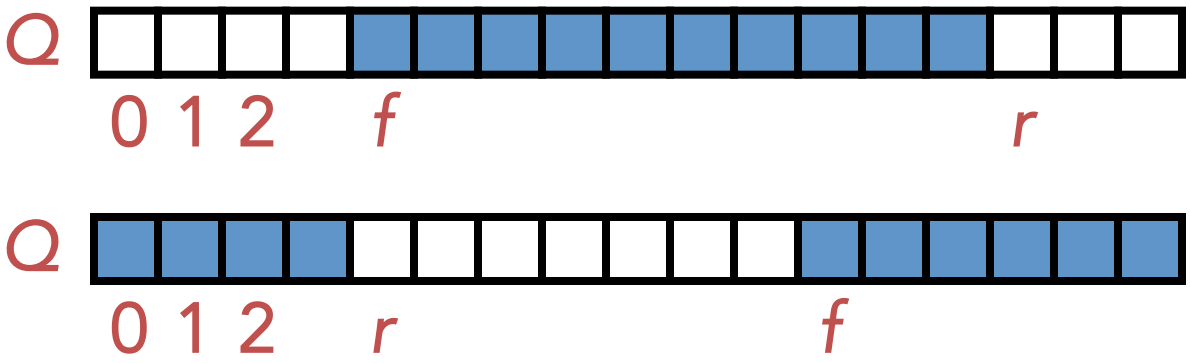
dequeue()

Algorithm *dequeue*()
    $f \leftarrow (f + 1) \bmod N$
    $n \leftarrow n - 1$



size() and empty()

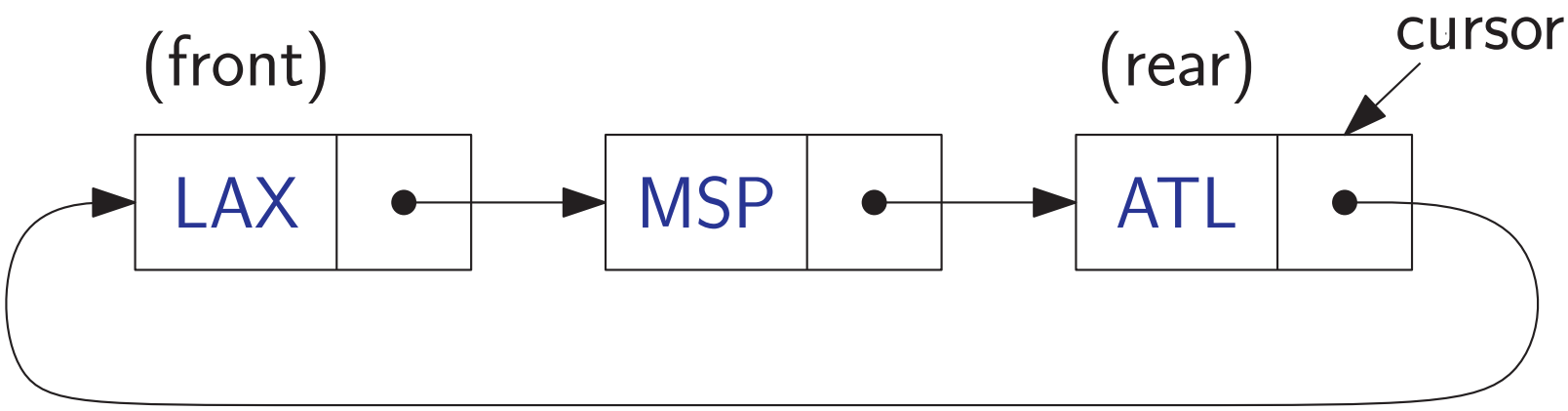
```
Algorithm size()
  return n

Algorithm empty()
  return (n == 0)
```



Array-based Queue Limitation -fixed size!

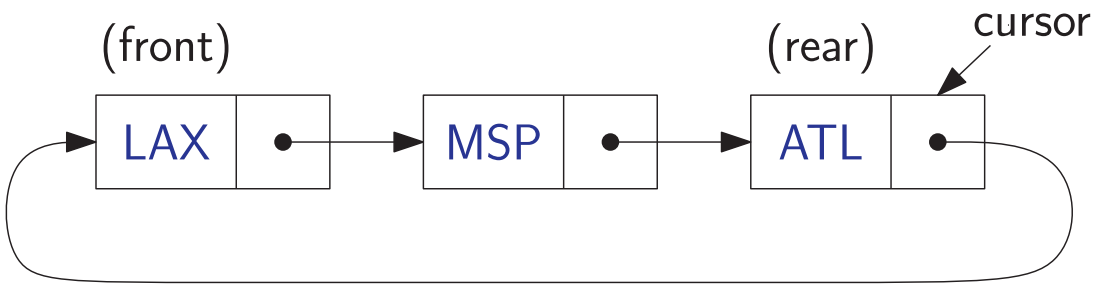
Linked list Implementation of Queues

Circular Linked List for Queue

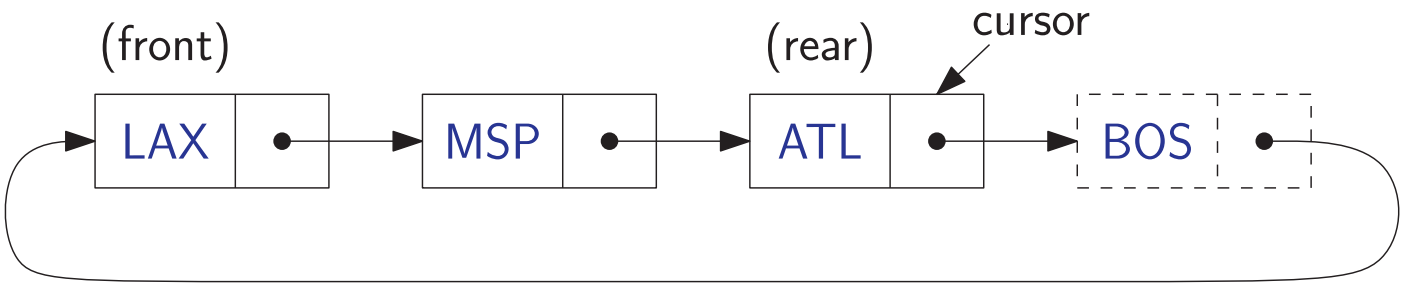


Circular List		Queue
front	➔	front
rear	➔	back

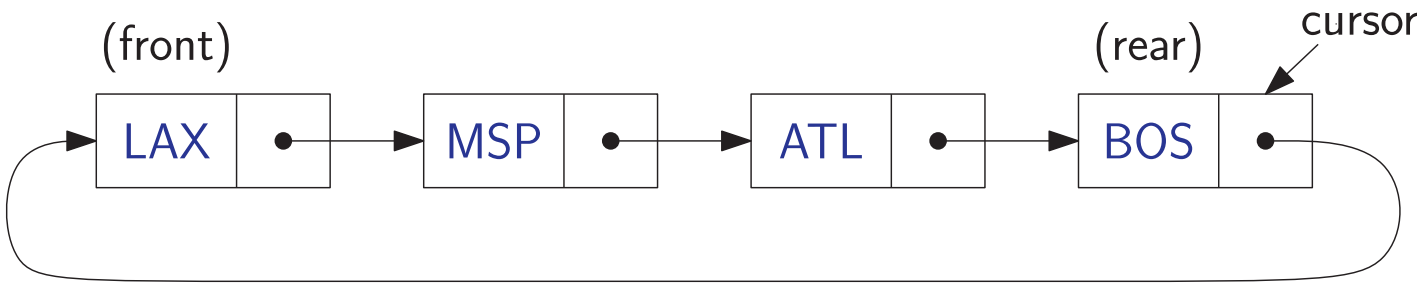
Enqueue


(front)
(rear)
cursor
LAX
MSP
ATL

(a)

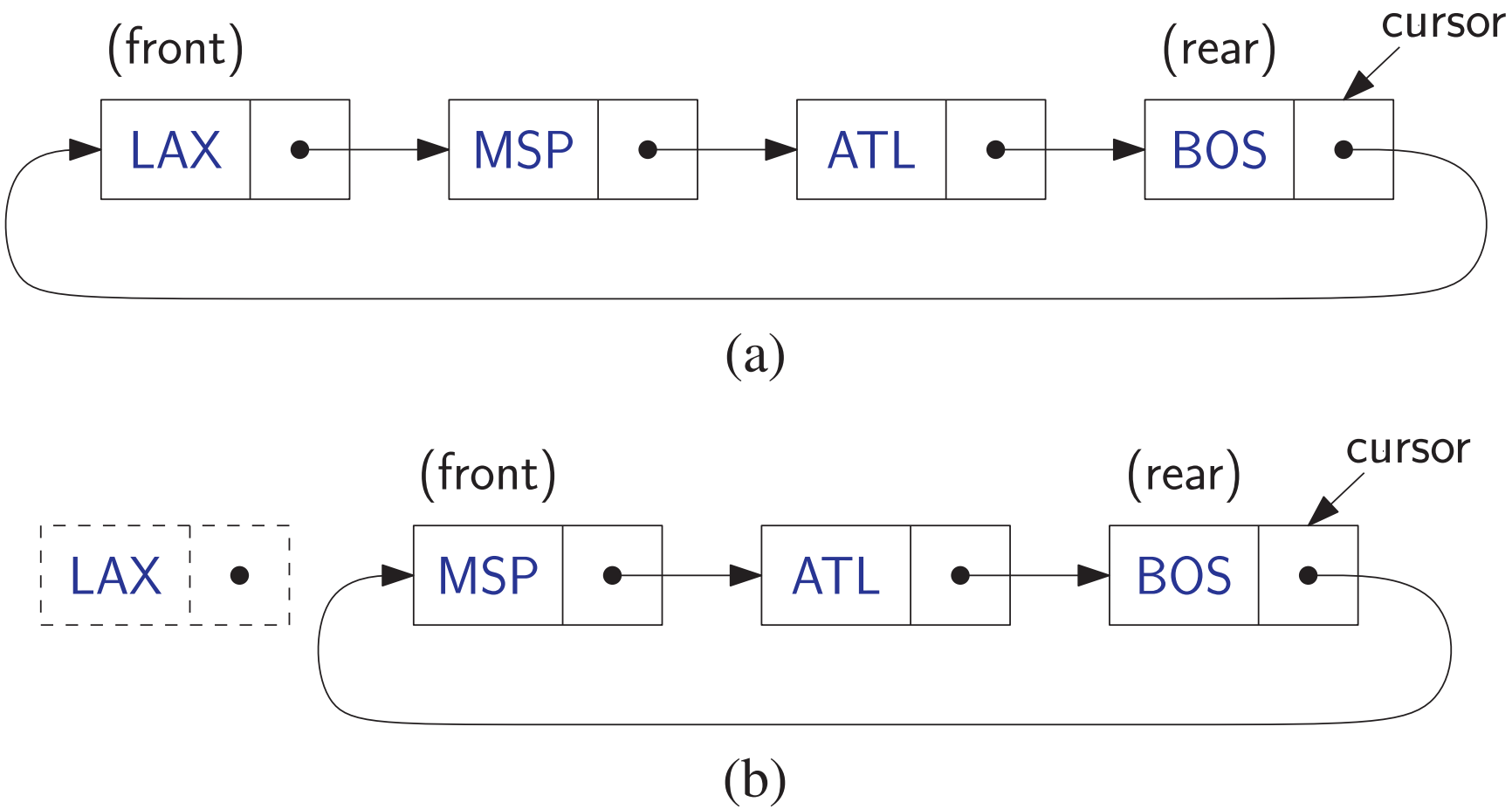

(front)
(rear)
cursor
LAX
MSP
ATL
BOS

(b)


(front)
(rear)
cursor
LAX
MSP
ATL
BOS

(c)

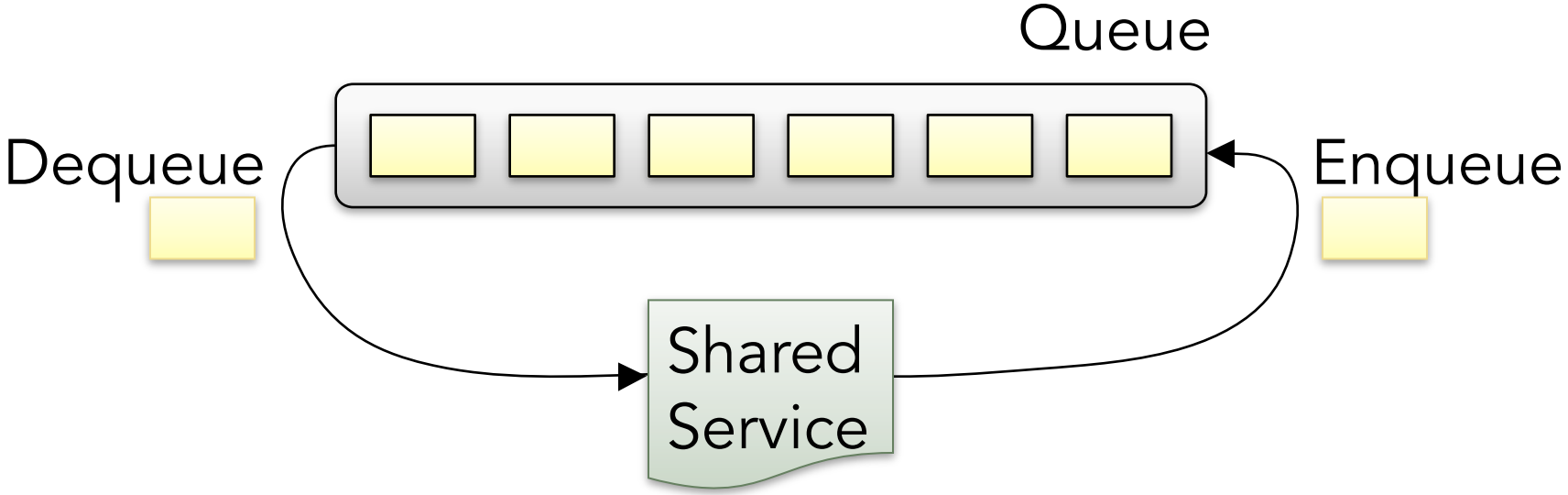
Dequeue


(front)
(rear)
cursor
LAX
MSP
ATL
BOS
(a)
(front)
(rear)
cursor
LAX
MSP
ATL
BOS
(b)

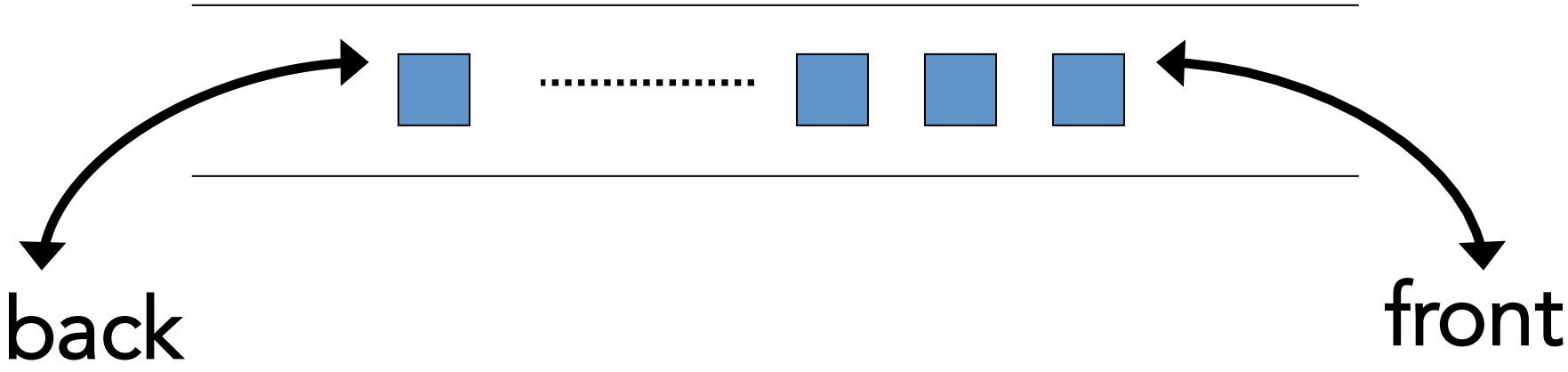
Round Robin Schedulers

Repeat 1-3

1. e = Q.front(); Q.dequeue()
2. Service element e
3. Q.enqueue(e)



Double-Ended Queue : Deque

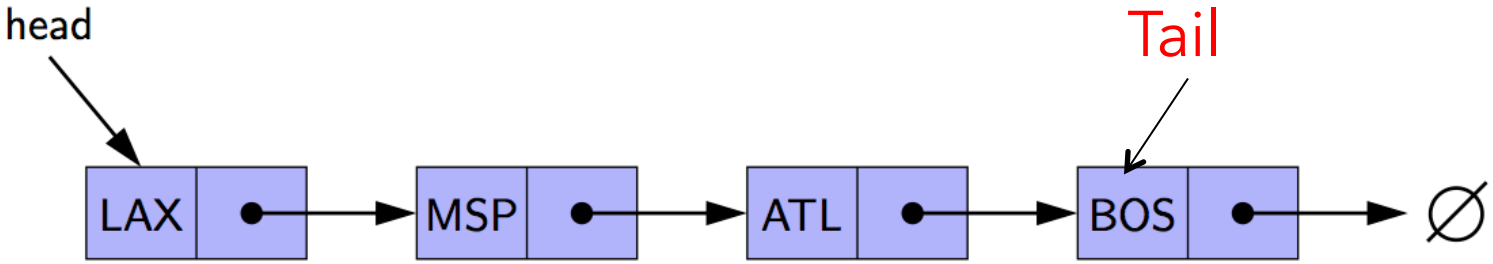


Fundamental operations

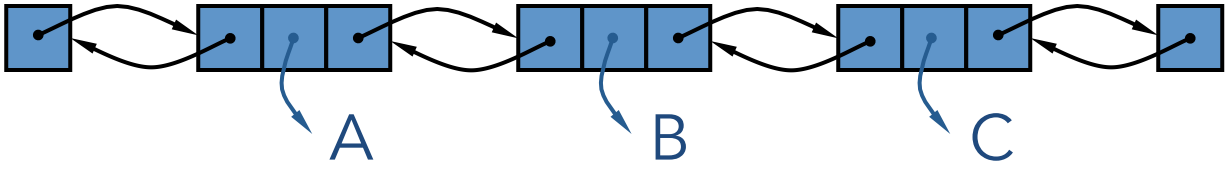
- InsertFirst(e)
- InsertLast(e)
- RemoveFirst()
- RemoveLast()
- First()
- Last()

Deque implementations

- Singly linked list



- Doubly linked list



Maximum of all subarrays of size k

- Input: [9 0 8 1 5 7 19 21 3 64 18]
- Output:[9 8 8 19 21 21 64 64]

D = (1,2,3,4,5,6,7,8)

Q = ()

Change D to

D = (1,2,3,5,4,6,7,8)