ساختمان داده و الگوریتم ها

مبحث نهم:
صف

سجاد شیرعلی شهرضا

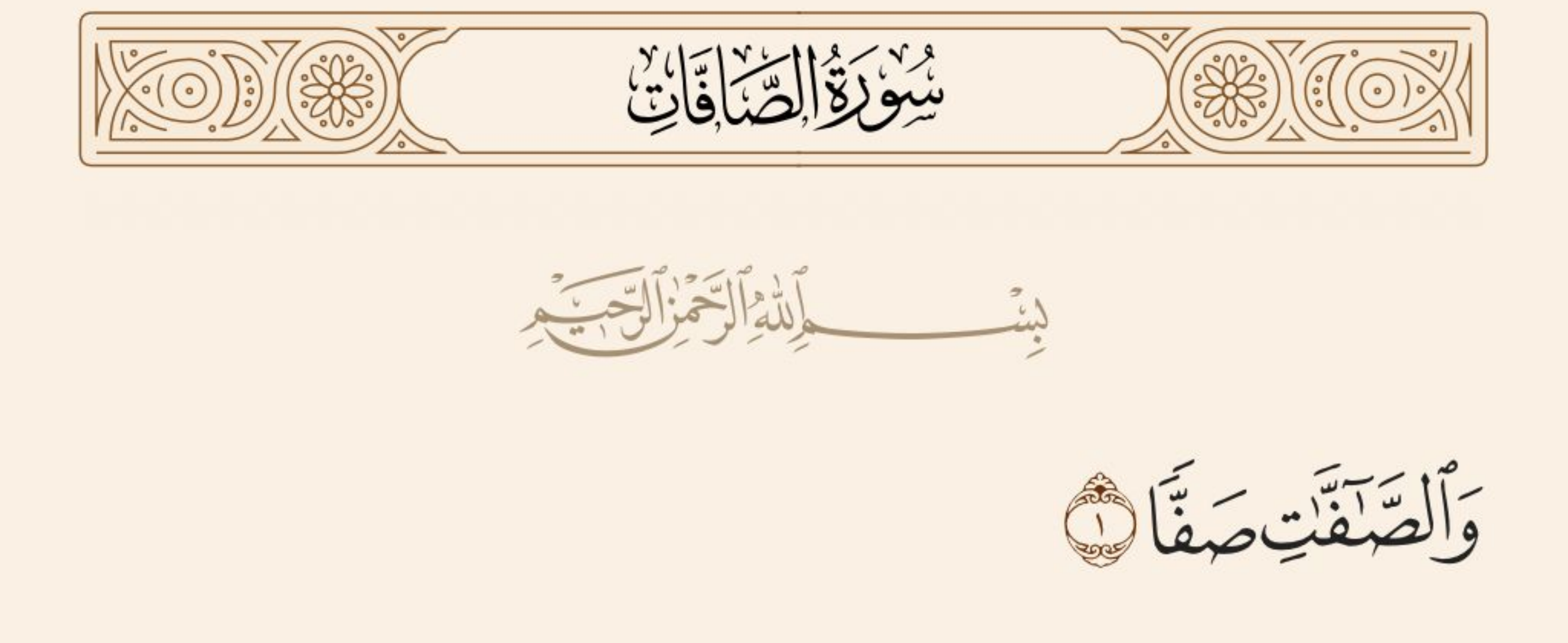
بهار 1402

شنبه، 6 آبان 1402

Based on Washington University CSE 373 (Autumn 2020 by Hunter Schafer), University of North Florida COP2220 (Chip Klostermeyer), and University of Illinois Chicago CS 201 (Spring 2014 by Bing Liu) slides

اطلاع رسانی

- بخش مرتبط کتاب برای این جلسه: 10
- امتحانک دوم
  - دوشنبه هفته آینده (15 آبان 1402)
  - در ساعت کلاس
  - در محل کلاس



سُورَةُ الصَّافَّاتِ

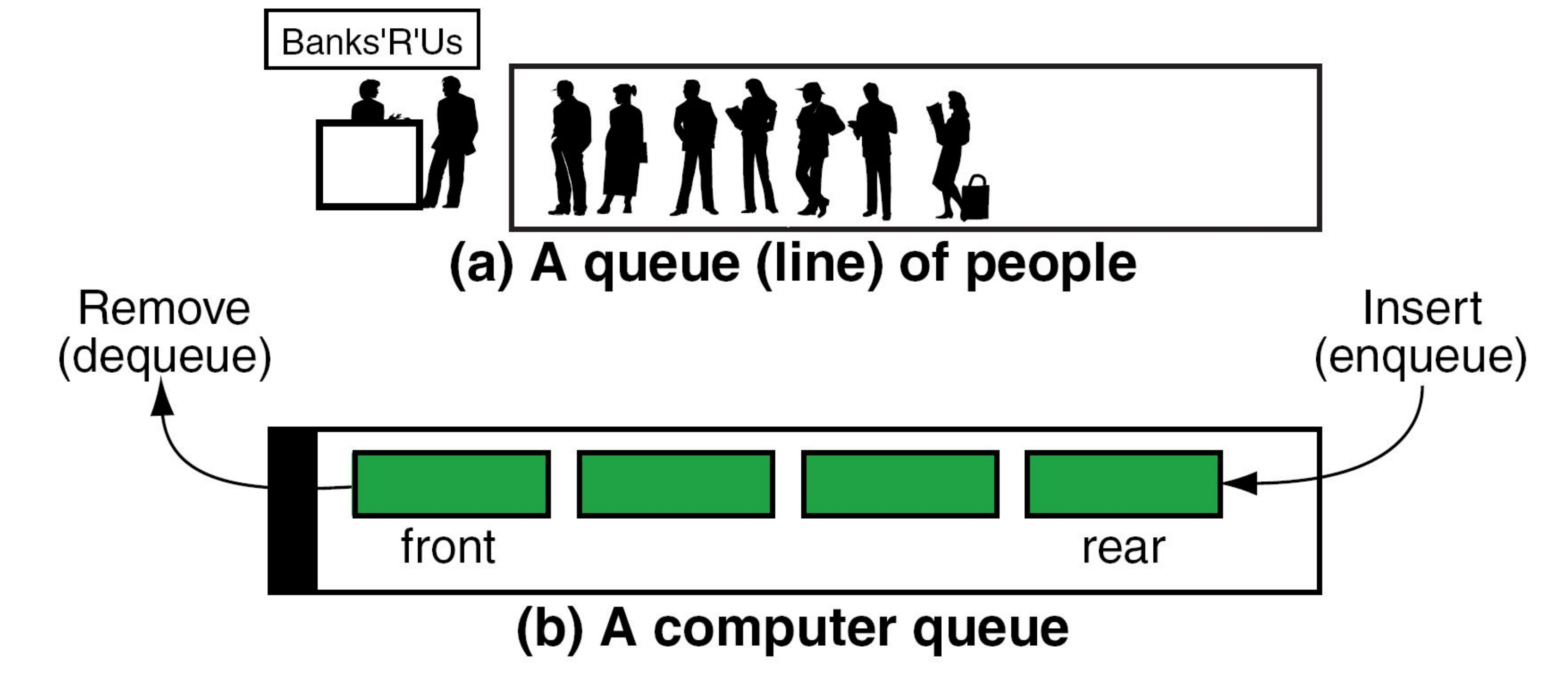
بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

وَٱلصَّٰٓفَّٰتِ صَفّٗا ١

صف

مجموعه ای از اشیاء پشت سر هم قرار گرفته

Idea

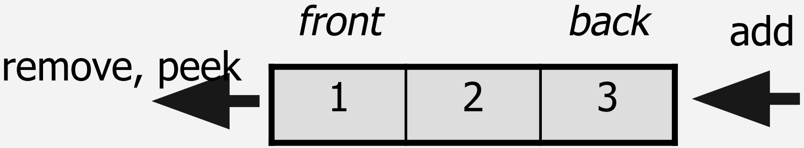


(a) A queue (line) of people

(b) A computer queue

Queue ADT

- Represents an ordered sequence of elements
- Elements can only be added from one end and removed from the other
- First-In, First-Out (FIFO)
- Elements stored in order of insertion



QUEUE ADT

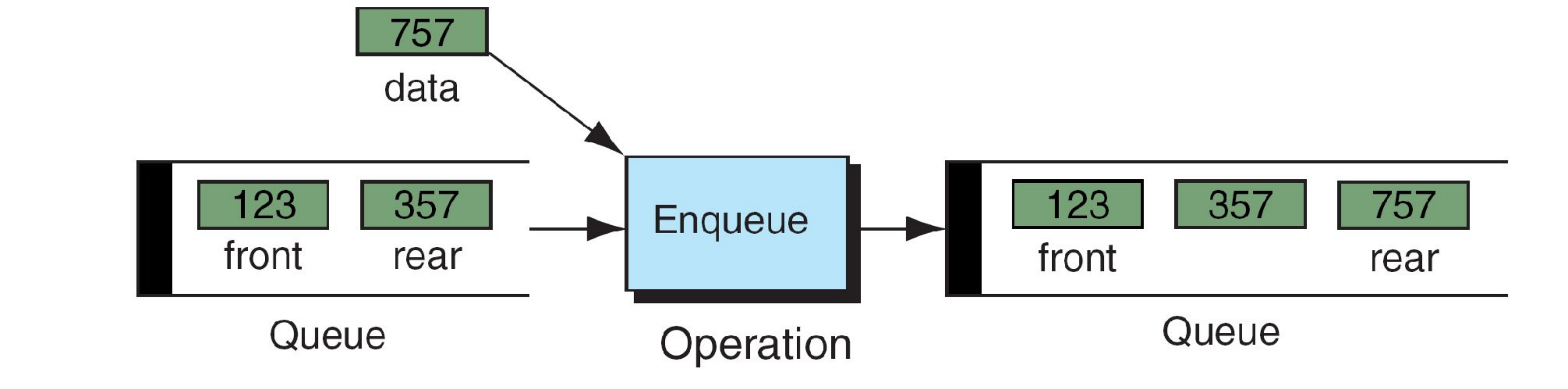
State

Collection of ordered items
Count of items

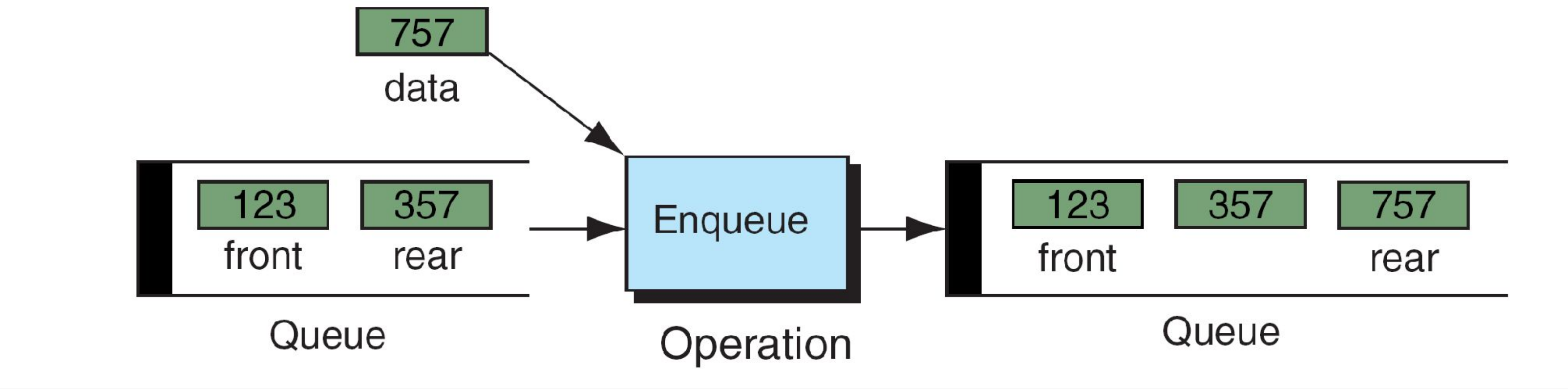
Behavior

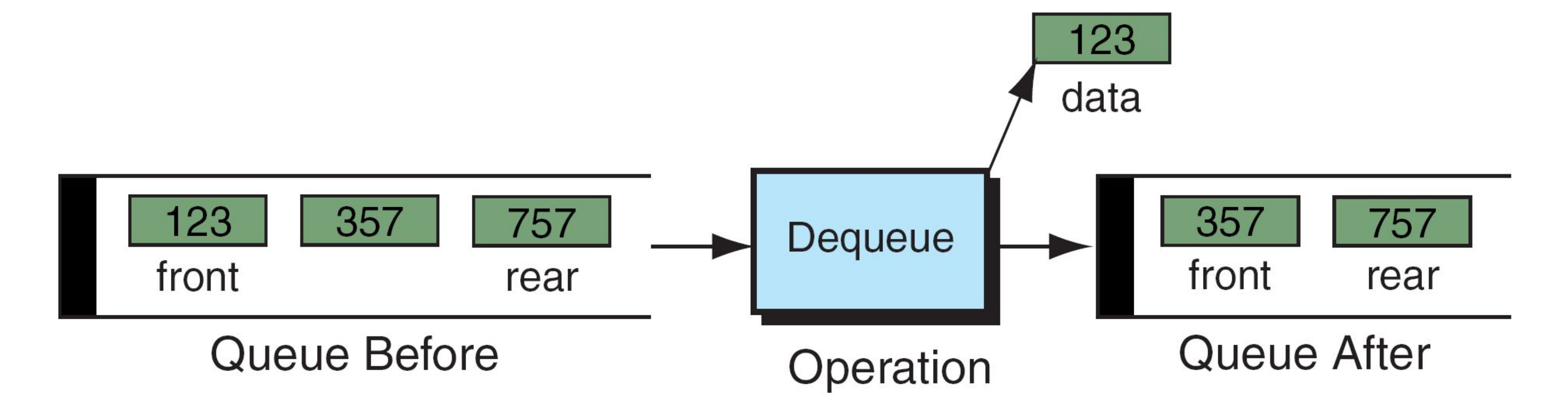
```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

Operations


757
data
123
front
357
rear
Queue
Enqueue
Operation
123
front
357
757
rear
Queue

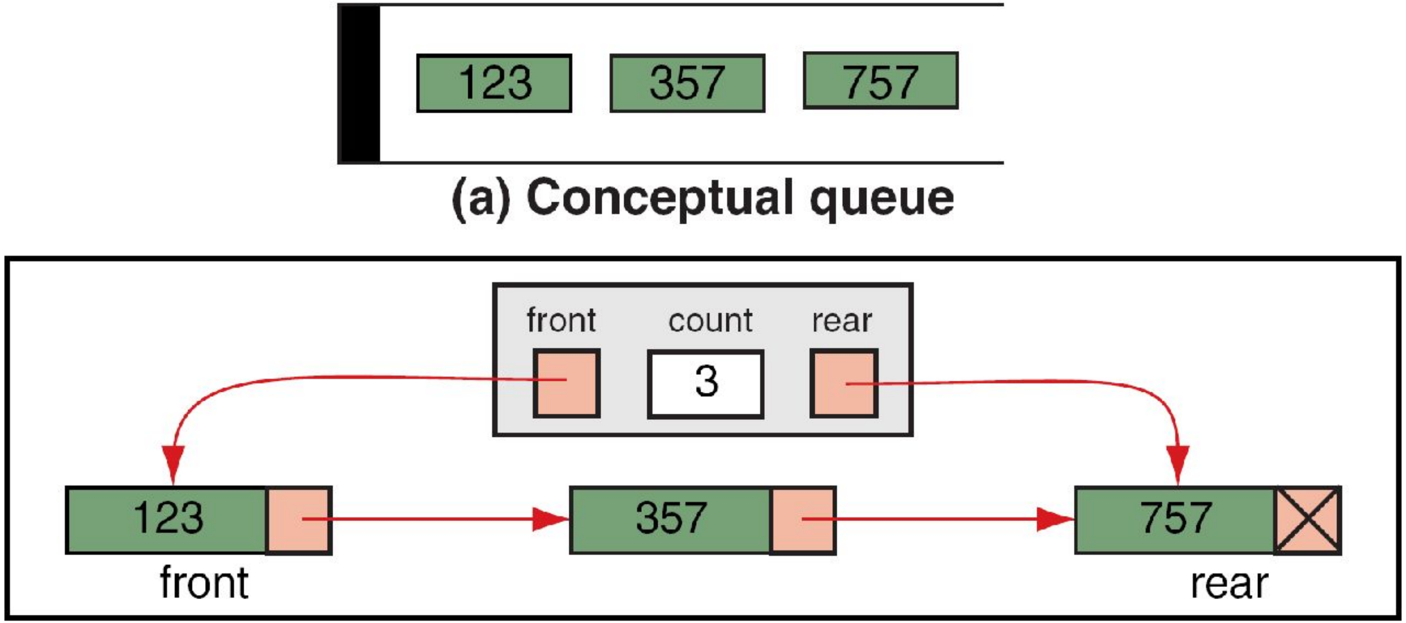
Operations


757
data
123
front
357
rear
Queue
Enqueue
Operation
123
front
357
757
rear
Queue


123
data
123
front
357
757
rear
Queue Before
Dequeue
Operation
357
front
757
rear
Queue After

پیاده سازی صف با لیست پیوندی

Linked-list Implementation



(a) Conceptual queue

(b) Physical queue

Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

State

```
Node front
Node back
size
```

Behavior

```
add – add node to back
remove – return and remove
node at front
peek – return node at front
size – return size
isEmpty – return size == 0
```

Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

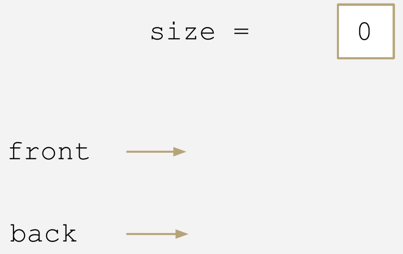
LinkedQueue<E>

State

Node front
Node back
size

Behavior

add – add node to back
remove – return and remove node at front
peek – return node at front
size – return size
isEmpty – return size == 0



size = 0

front

back

Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

LinkedQueue<E>

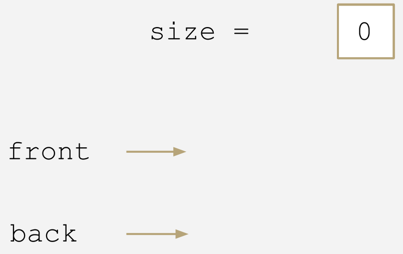
State

Node front
Node back
size

Behavior

add - add node to back
remove - return and remove node at front
peek - return node at front
size - return size
isEmpty - return size == 0

add(5)



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

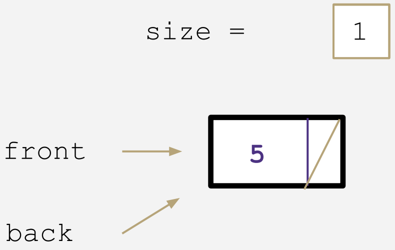
State

```
Node front
Node back
size
```

Behavior

```
add - add node to back
remove - return and remove
node at front
peek - return node at front
size - return size
isEmpty - return size == 0
```

```
add(5)
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

LinkedQueue<E>

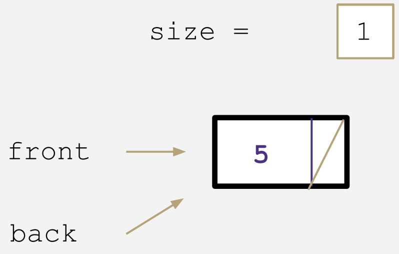
State

Node front
Node back
size

Behavior

add – add node to back
remove – return and remove node at front
peek – return node at front
size – return size
isEmpty – return size == 0

```
add(5)
add(8)
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

LinkedQueue<E>

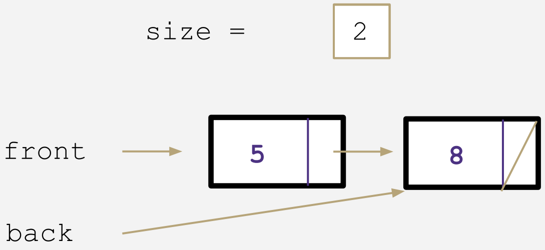
State

```
Node front
Node back
size
```

Behavior

add - add node to back
remove - return and remove node at front
peek - return node at front
size - return size
isEmpty - return size == 0

```
add(5)
add(8)
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

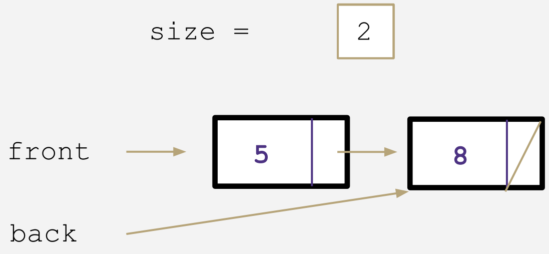
State

```
Node front
Node back
size
```

Behavior

```
add – add node to back
remove – return and remove
node at front
peek – return node at front
size – return size
isEmpty – return size == 0
```

```
add(5)
add(8)
remove()
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

`add(item)` add item to back
`remove()` remove and return item at front
`peek()` return item at front
`size()` count of items
`isEmpty()` count is 0?

LinkedQueue<E>

State

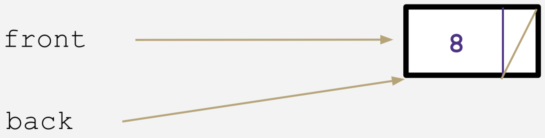
```
Node front
Node back
size
```

Behavior

`add` – add node to back
`remove` – return and remove node at front
`peek` – return node at front
`size` – return size
`isEmpty` – return size == 0

```
size =  1
```

```
add(5)
add(8)
remove()
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

State

```
Node front
Node back
size
```

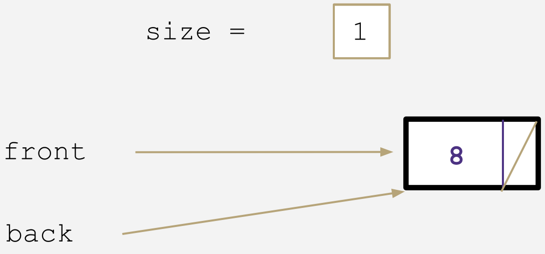
Behavior

```
add – add node to back
remove – return and remove
node at front
peek – return node at front
size – return size
isEmpty – return size == 0
```

Big-Oh Analysis

```
remove()

peek()

size()

isEmpty()

add()
```

```
add(5)
add(8)
remove()
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

State

```
Node front
Node back
size
```

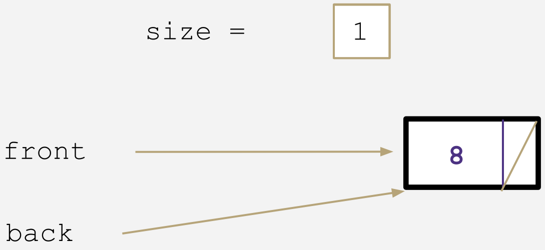
Behavior

```
add - add node to back
remove - return and remove
node at front
peek - return node at front
size - return size
isEmpty - return size == 0
```

Big-Oh Analysis

remove()	O(1) Constant
peek()	
size()	
isEmpty()	
add()	

```
add(5)
add(8)
remove()
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

State

```
Node front
Node back
size
```

Behavior

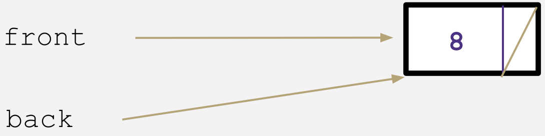
```
add - add node to back
remove - return and remove
node at front
peek - return node at front
size - return size
isEmpty - return size == 0
```

Big-Oh Analysis

remove()	O(1) Constant
peek()	O(1) Constant
size()	
isEmpty()	
add()	

```
size =    1
```

```
add(5)
add(8)
remove()
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

State

```
Node front
Node back
size
```

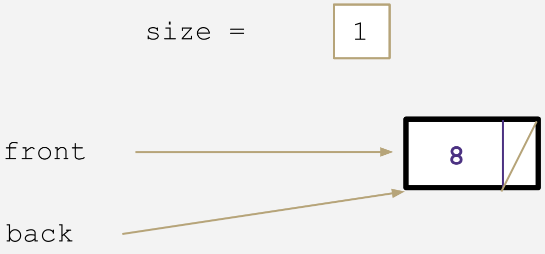
Behavior

```
add - add node to back
remove - return and remove
node at front
peek - return node at front
size - return size
isEmpty - return size == 0
```

Big-Oh Analysis

remove()	O(1) Constant
peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	
add()	

```
add(5)
add(8)
remove()
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

State

```
Node front
Node back
size
```

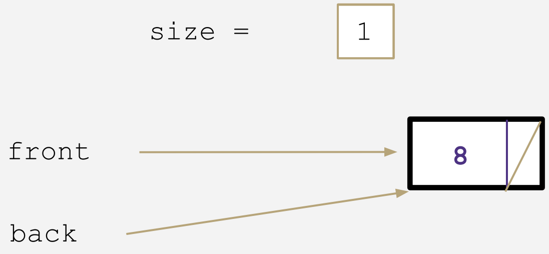
Behavior

```
add – add node to back
remove – return and remove
node at front
peek – return node at front
size – return size
isEmpty – return size == 0
```

Big-Oh Analysis

remove()	O(1) Constant
peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	O(1) Constant
add()	

```
add(5)
add(8)
remove()
```



Implementing a Queue with Linked Nodes

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

LinkedQueue<E>

State

```
Node front
Node back
size
```

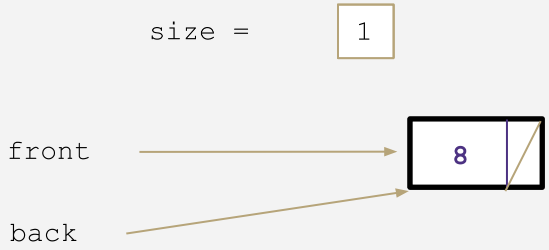
Behavior

```
add – add node to back
remove – return and remove
node at front
peek – return node at front
size – return size
isEmpty – return size == 0
```

Big-Oh Analysis

remove()	O(1) Constant
peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	O(1) Constant
add()	O(1) Constant

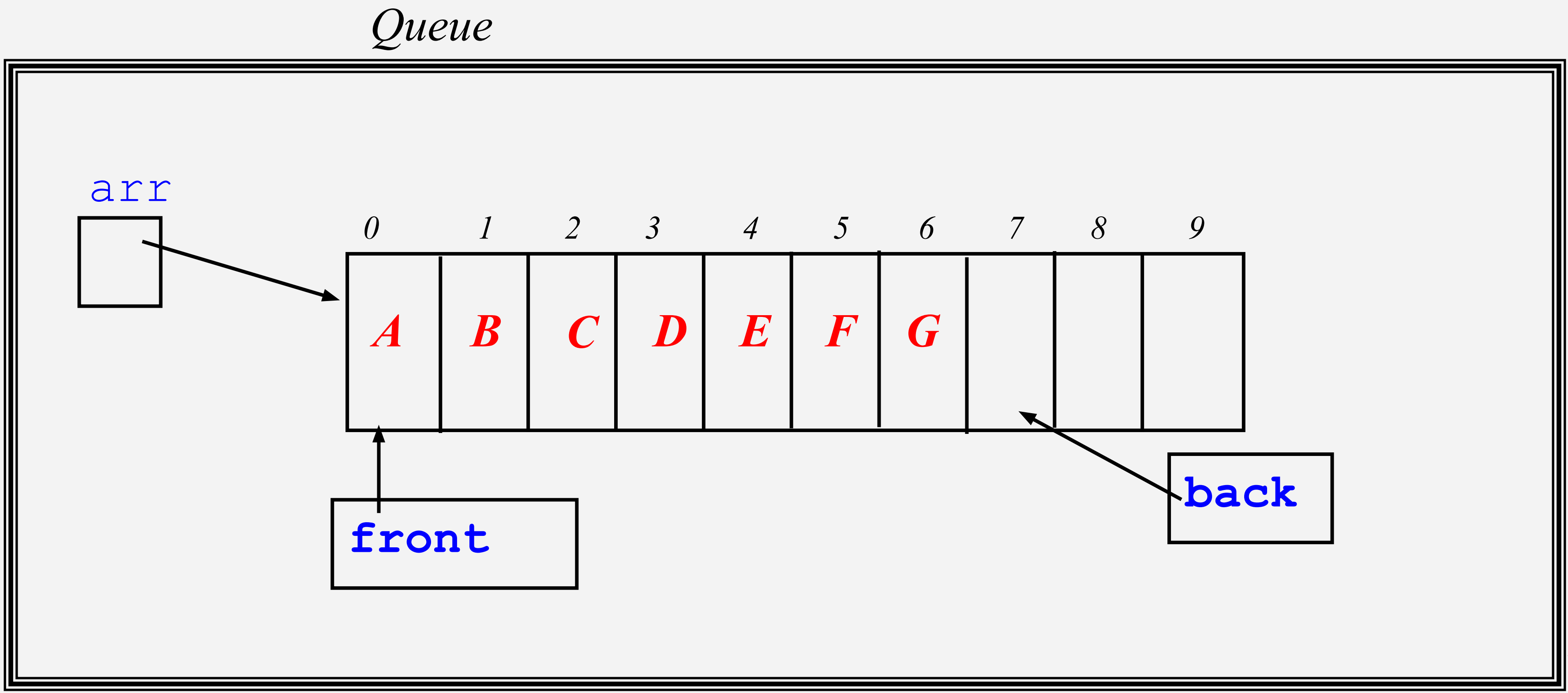
```
add(5)
add(8)
remove()
```



سوال؟

پیاده سازی صف با آرایه

Idea



Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

0	1	2	3	4

size = 0

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

ArrayQueueV1<E>

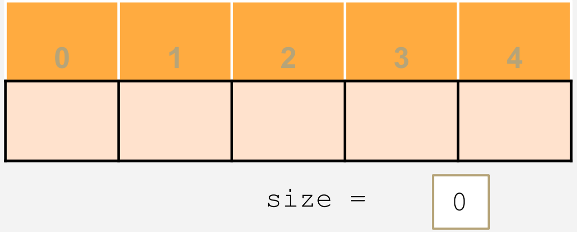
State

```
data[]
size
```

Behavior

```
add – data[size] = value,
if out of room grow
remove – return/remove at
0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0
```

add(5)



0 1 2 3 4

size = 0

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

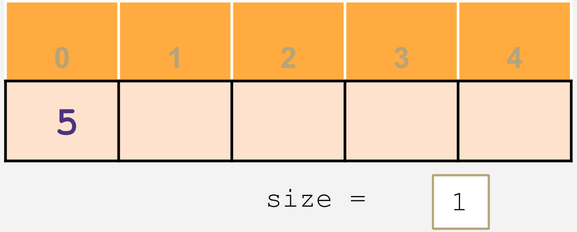
State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

add(5)



0	1	2	3	4
5				

size = 1

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

```
add(5)
add(8)
```

0	1	2	3	4
5				

size = 1

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

```
add(5)
add(8)
```

0	1	2	3	4
5	8			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

```
add(5)
add(8)
add(9)
```

0	1	2	3	4
5	8			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

```
add(5)
add(8)
add(9)
```

0	1	2	3	4
5	8	9		

size = 3

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

ArrayQueueV1<E>

State

```
data[]
size
```

Behavior

```
add – data[size] = value,
if out of room grow
remove – return/remove at
0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0
```

```
add(5)
add(8)
add(9)
remove()
```

0	1	2	3	4
5	8	9		

size = 3

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

`add(item)` add item to back
`remove()` remove and return item at front
`peek()` return item at front
`size()` count of items
`isEmpty()` count is 0?

ArrayQueueV1<E>

State

`data[]`
`size`

Behavior

`add` – data[size] = value, if out of room grow
`remove` – return/remove at 0, shift everything
`peek` – return node at 0
`size` – return size
`isEmpty` – return size == 0

```
add(5)
add(8)
add(9)
remove()
```

0	1	2	3	4
8	9			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

Big-Oh Analysis
peek()

size()

isEmpty()

add()

remove()

add(5)
add(8)
add(9)
remove()

0	1	2	3	4
8	9			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV1<E>

State

data[]
size

Behavior

add – data[size] = value, if out of room grow
remove – return/remove at 0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0

Big-Oh Analysis

peek()	O(1) Constant
size()	
isEmpty()	
add()	
remove()	

add(5)
add(8)
add(9)
remove()

0	1	2	3	4
8	9			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

ArrayQueueV1<E>

State

```
data[]
size
```

Behavior

```
add – data[size] = value,
if out of room grow
remove – return/remove at
0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0
```

Big-Oh Analysis

peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	
add()	
remove()	

```
add(5)
add(8)
add(9)
remove()
```

0	1	2	3	4
8	9			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

ArrayQueueV1<E>

State

```
data[]
size
```

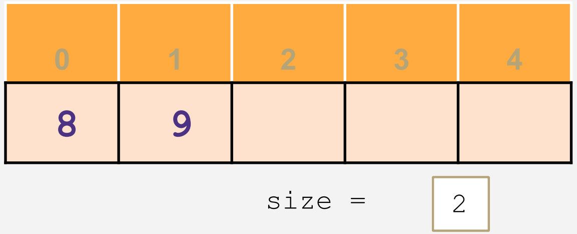
Behavior

```
add – data[size] = value,
if out of room grow
remove – return/remove at
0, shift everything
peek – return node at 0
size – return size
isEmpty – return size == 0
```

Big-Oh Analysis

peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	O(1) Constant
add()	
remove()	

```
add(5)
add(8)
add(9)
remove()
```



0	1	2	3	4
8	9			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

ArrayQueueV1<E>

State

```
data[]
size
```

Behavior

```
add - data[size] = value,
if out of room grow
remove - return/remove at
0, shift everything
peek - return node at 0
size - return size
isEmpty - return size == 0
```

Big-Oh Analysis

`peek()`	O(1) Constant
`size()`	O(1) Constant
`isEmpty()`	O(1) Constant
`add()`	What are different cases?
`remove()`	

```
add(5)
add(8)
add(9)
remove()
```

0	1	2	3	4
8	9			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

`add(item)` add item to back
`remove()` remove and return item at front
`peek()` return item at front
`size()` count of items
`isEmpty()` count is 0?

ArrayQueueV1<E>

State

`data[]`
`size`

Behavior

`add` - data[size] = value, if out of room grow
`remove` - return/remove at 0, shift everything
`peek` - return node at 0
`size` - return size
`isEmpty` - return size == 0

Big-Oh Analysis

peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	O(1) Constant
add()	O(n) Linear: if we need to resize O(1) Constant: otherwise
remove()	

```
add(5)
add(8)
add(9)
remove()
```

0	1	2	3	4
8	9			

size = 2

Implementing a Queue with an Array

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

`add(item)` add item to back
`remove()` remove and return item at front
`peek()` return item at front
`size()` count of items
`isEmpty()` count is 0?

ArrayQueueV1<E>

State

`data[]`
`size`

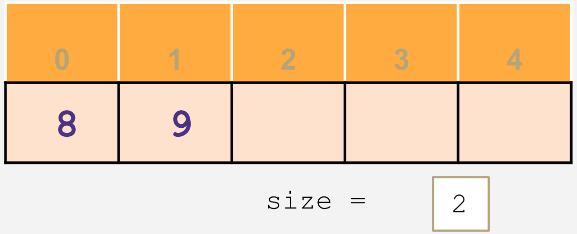
Behavior

`add` - data[size] = value, if out of room grow
`remove` - return/remove at 0, shift everything
`peek` - return node at 0
`size` - return size
`isEmpty` - return size == 0

Big-Oh Analysis

`peek()`	O(1) Constant
`size()`	O(1) Constant
`isEmpty()`	O(1) Constant
`add()`	O(n) Linear: if we need to resize O(1) Constant: otherwise
`remove()`	O(n) Linear

```
add(5)
add(8)
add(9)
remove()
```



0	1	2	3	4
8	9			

size = 2

سوال؟

Data Structure Invariants

- Invariant: a property of a data structure that is always true between operations
    - True when finishing any operation
    - It can be counted on to be true when starting an operation

Data Structure Invariants

- Invariant: a property of a data structure that is always true between operations
  - True when finishing any operation
  - It can be counted on to be true when starting an operation
- ArrayQueue is basically an ArrayList
- What invariants does ArrayList have for its data array?

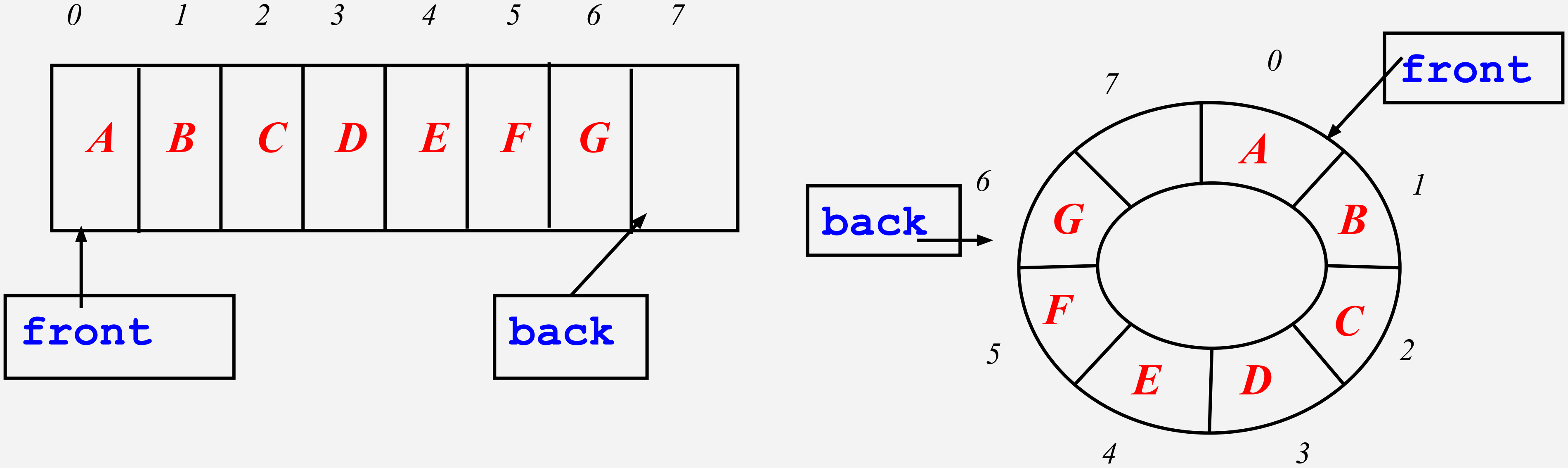
Data Structure Invariants

- Invariant: a property of a data structure that is always true between operations
    - True when finishing any operation
    - It can be counted on to be true when starting an operation
- ArrayQueue is basically an ArrayList
- What invariants does ArrayList have for its data array?
    - The i-th item in the list is stored in data[i]

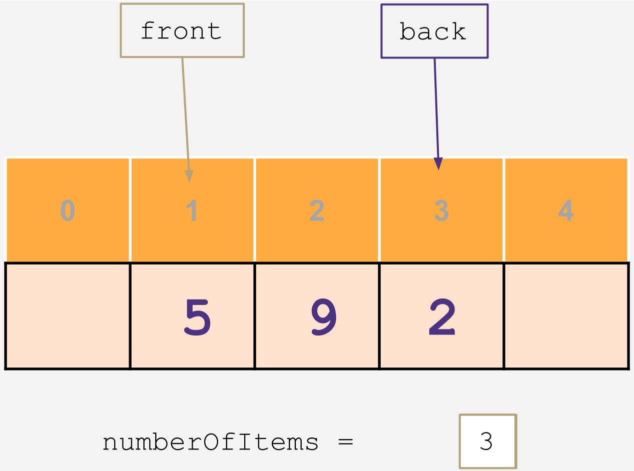
Data Structure Invariants

- Invariant: a property of a data structure that is always true between operations
  - True when finishing any operation
  - It can be counted on to be true when starting an operation
- ArrayQueue is basically an ArrayList
- What invariants does ArrayList have for its data array?
  - The i-th item in the list is stored in data[i]
- Notice: serving this invariant is what slows down the operation.
- Could we choose a different invariant?

Circular Array



Wrapping Around with “front” and “back” indices



front

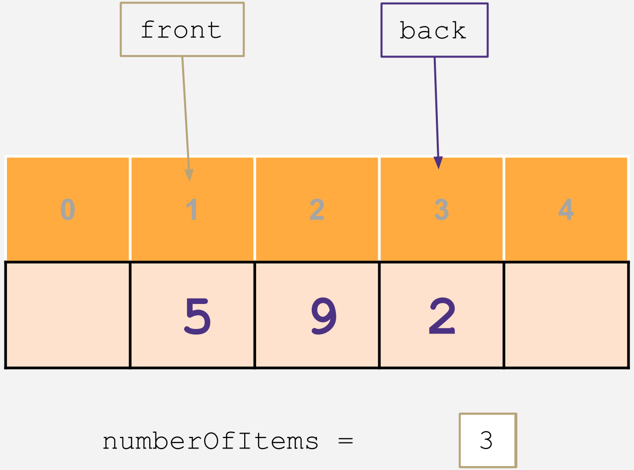
back

0	1	2	3	4
	5	9	2	

numberOfItems = 3

Wrapping Around with “front” and “back” indices

add(7)



front

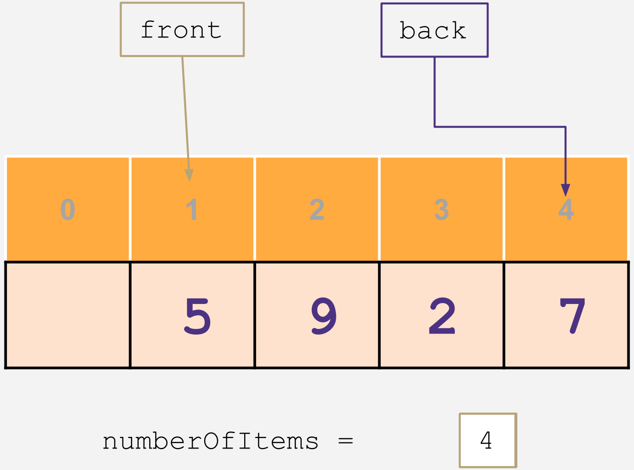
back

0	1	2	3	4
	5	9	2	

numberOfItems = 3

Wrapping Around with “front” and “back” indices

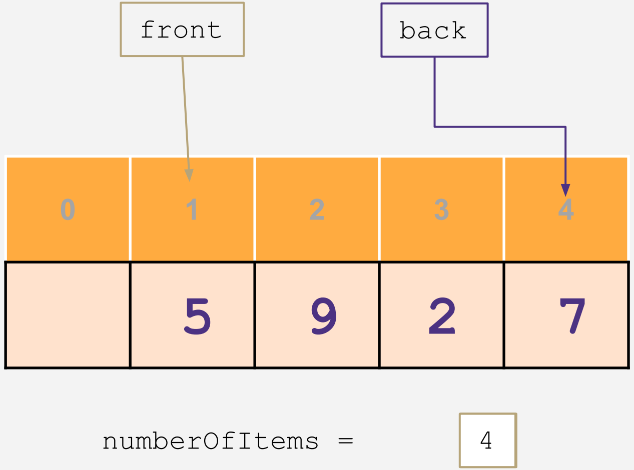
add(7)



Wrapping Around with “front” and “back” indices

add(7)

add(4)



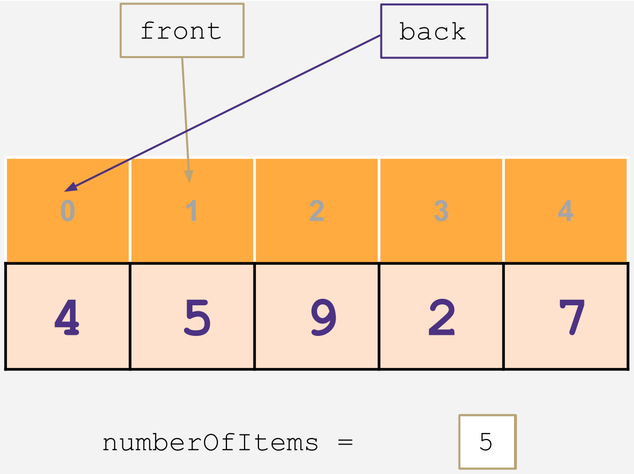
0	1	2	3	4
	5	9	2	7

numberOfItems = 4

Wrapping Around with “front” and “back” indices

add(7)

add(4)



0	1	2	3	4
4	5	9	2	7

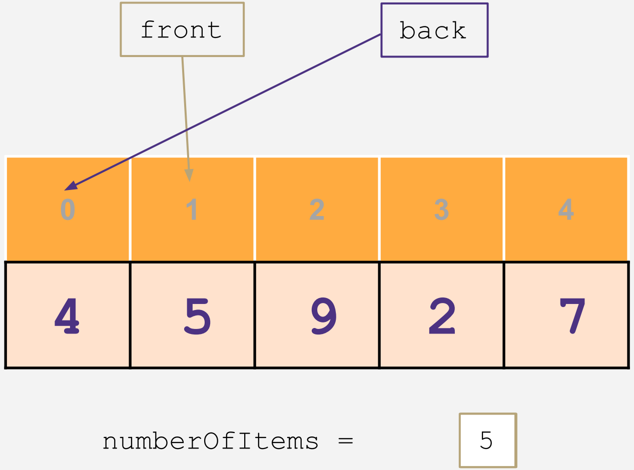
numberOfItems = 5

Wrapping Around with “front” and “back” indices

add(7)

add(4)

add(1)



0	1	2	3	4
4	5	9	2	7

numberOfItems = 5

Wrapping Around with “front” and “back” indices

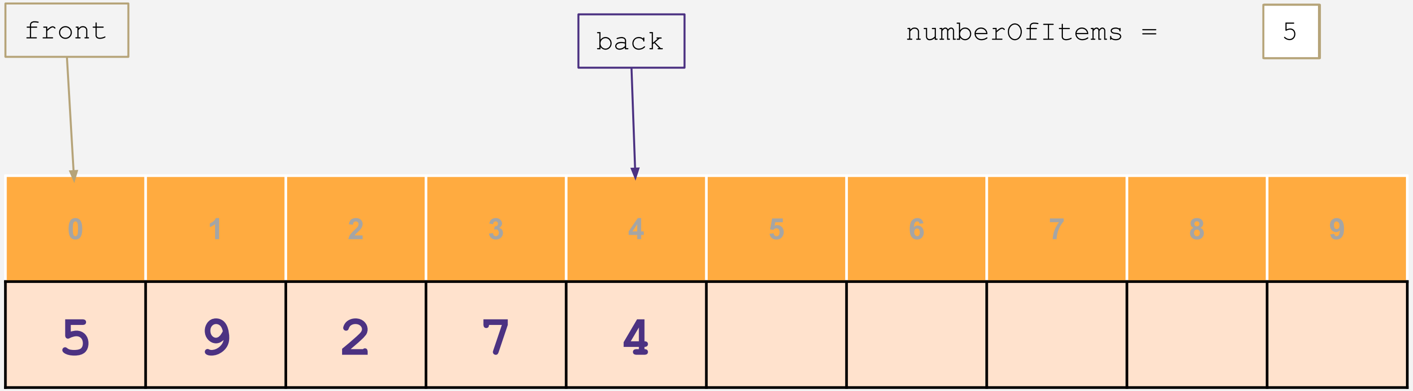
add(7)

add(4)

add(1)

0	1	2	3	4
4	5	9	2	7

`numberOfItems =` 5



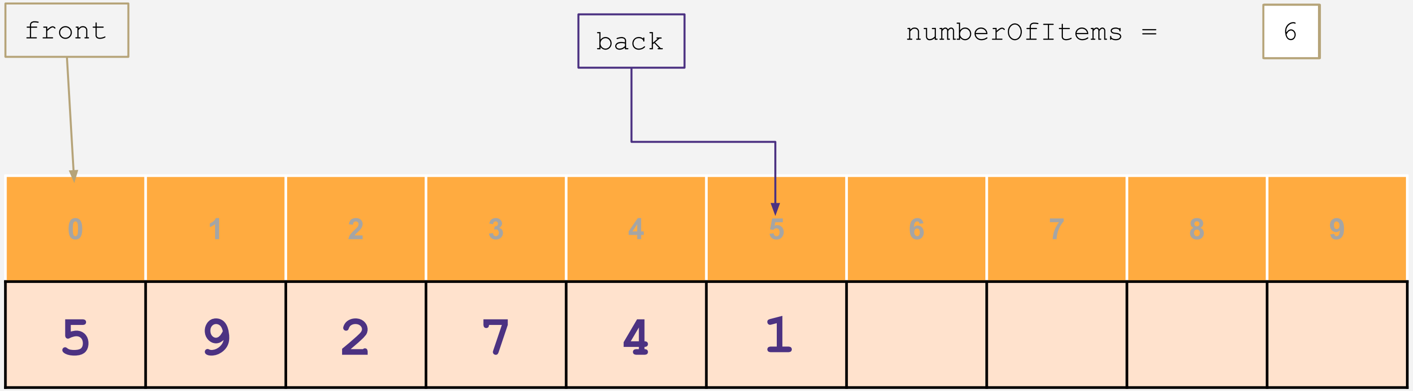
0	1	2	3	4	5	6	7	8	9
5	9	2	7	4					

Wrapping Around with “front” and “back” indices

add(7)

add(4)

add(1)



0	1	2	3	4	5	6	7	8	9
5	9	2	7	4	1				

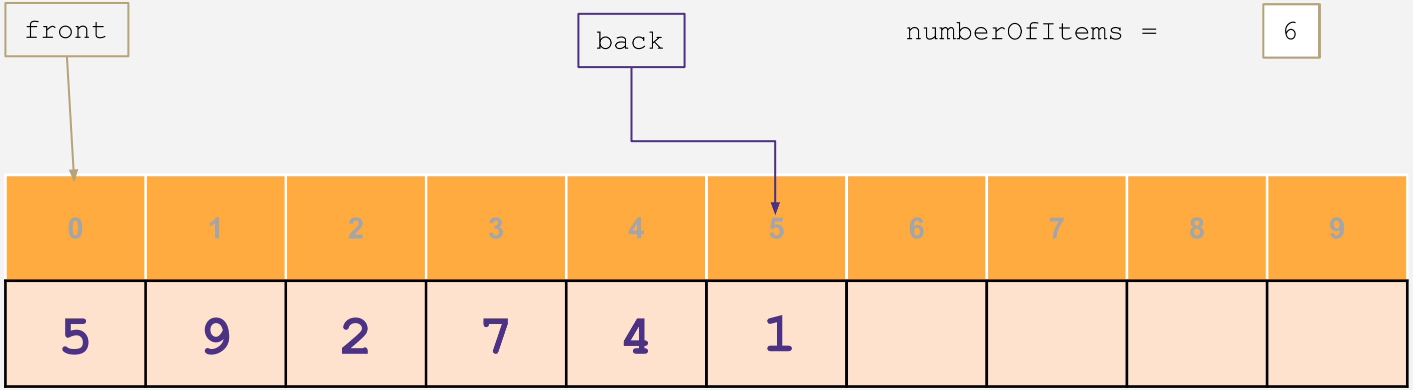
Wrapping Around with “front” and “back” indices

add(7)

add(4)

add(1)

remove()

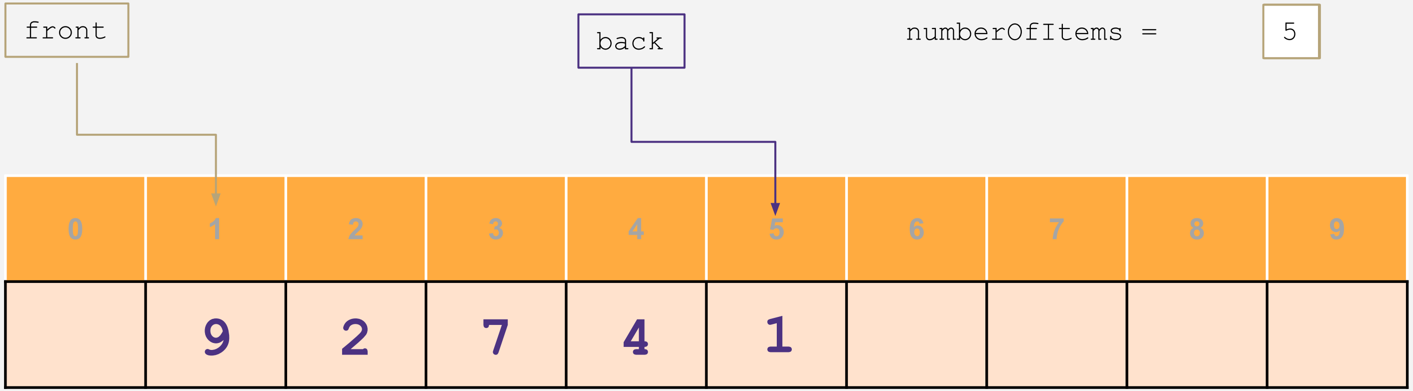


numberOfItems = 6

0	1	2	3	4	5	6	7	8	9
5	9	2	7	4	1				

Wrapping Around with “front” and “back” indices

add(7)
add(4)
add(1)
remove()



front

back

numberOfItems = 5

0	1	2	3	4	5	6	7	8	9
	9	2	7	4	1				

Implementing a Queue with an Array (v2)

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

ArrayQueueV2<E>

State

```
data[],  front,
size,    back
```

Behavior

```
add – data[back] = value,
back++, size++, if out of
room grow
remove – return data[front],
size--, front++
peek – return data[front]
size – return size
isEmpty – return size == 0
```

Implementing a Queue with an Array (v2)

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

`add(item)` add item to back
`remove()` remove and return item at front
`peek()` return item at front
`size()` count of items
`isEmpty()` count is 0?

ArrayQueueV2<E>

State

```
data[],  front,
size,    back
```

Behavior

`add` – data[back] = value, back++, size++, if out of room grow
`remove` – return data[front], size--, front++
`peek` – return data[front]
`size` – return size
`isEmpty` – return size == 0

Big-Oh Analysis

`peek()`

`size()`

`isEmpty()`

`add()`

`remove()`

Implementing a Queue with an Array (v2)

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV2<E>

State

```
data[],   front,
size,     back
```

Behavior

add – data[back] = value, back++, size++, if out of room grow
remove – return data[front], size--, front++
peek – return data[front]
size – return size
isEmpty – return size == 0

Big-Oh Analysis

peek()	O(1) Constant
size()	
isEmpty()	
add()	
remove()	

Implementing a Queue with an Array (v2)

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

`add(item)` add item to back
`remove()` remove and return item at front
`peek()` return item at front
`size()` count of items
`isEmpty()` count is 0?

ArrayQueueV2<E>

State

```
data[],  front,
size,    back
```

Behavior

`add` – data[back] = value, back++, size++, if out of room grow
`remove` – return data[front], size--, front++
`peek` – return data[front]
`size` – return size
`isEmpty` – return size == 0

Big-Oh Analysis

peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	
add()	
remove()	

Implementing a Queue with an Array (v2)

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

```
add(item) add item to back
remove() remove and return
item at front
peek() return item at front
size() count of items
isEmpty() count is 0?
```

ArrayQueueV2<E>

State

```
data[],   front,
size,     back
```

Behavior

```
add – data[back] = value,
back++, size++, if out of
room grow
remove – return data[front],
size--, front++
peek – return data[front]
size – return size
isEmpty – return size == 0
```

Big-Oh Analysis

peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	O(1) Constant
add()	
remove()	

Implementing a Queue with an Array (v2)

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV2<E>

State

```
data[],   front,
size,     back
```

Behavior

add – data[back] = value, back++, size++, if out of room grow
remove – return data[front], size--, front++
peek – return data[front]
size – return size
isEmpty – return size == 0

Big-Oh Analysis

peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	O(1) Constant
add()	O(n) Linear: if we need to resize O(1) Constant: otherwise
remove()	

Implementing a Queue with an Array (v2)

QUEUE ADT

State

Collection of ordered items
Count of items

Behavior

add(item) add item to back
remove() remove and return item at front
peek() return item at front
size() count of items
isEmpty() count is 0?

ArrayQueueV2<E>

State

data[], front,
size, back

Behavior

add – data[back] = value, back++, size++, if out of room grow
remove – return data[front], size--, front++
peek – return data[front]
size – return size
isEmpty – return size == 0

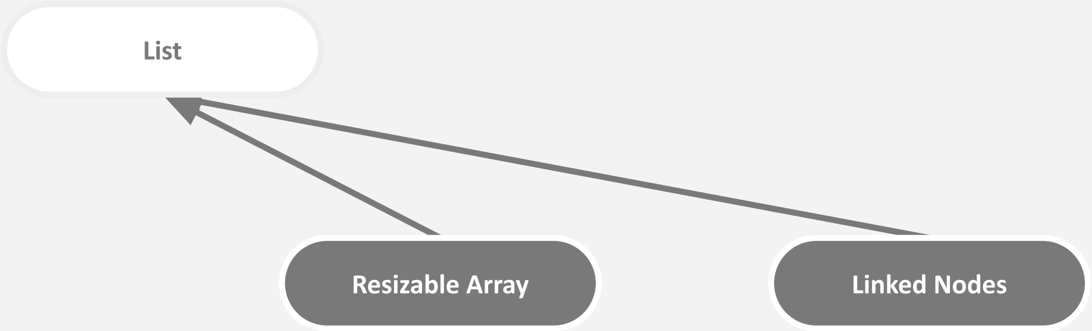
Big-Oh Analysis

peek()	O(1) Constant
size()	O(1) Constant
isEmpty()	O(1) Constant
add()	O(n) Linear: if we need to resize O(1) Constant: otherwise
remove()	O(1) Constant

سوال؟

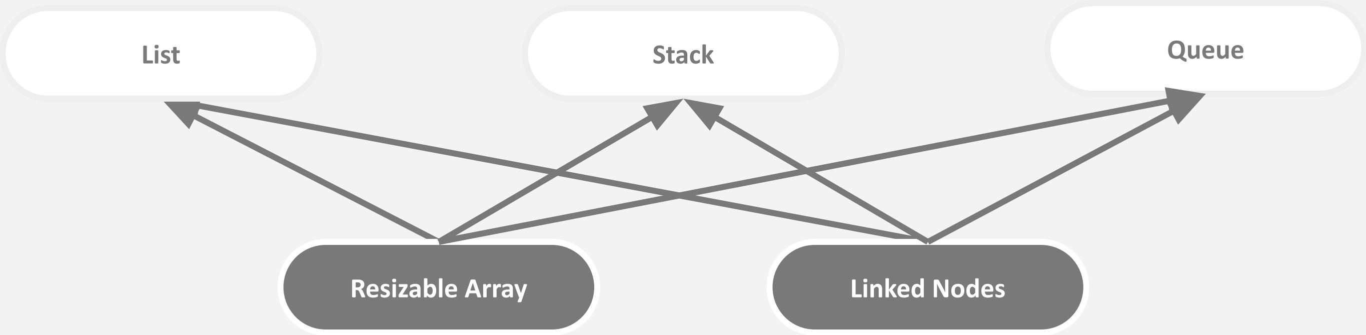
ADTs & Data Structures

- ADT can be implemented by multiple data structures



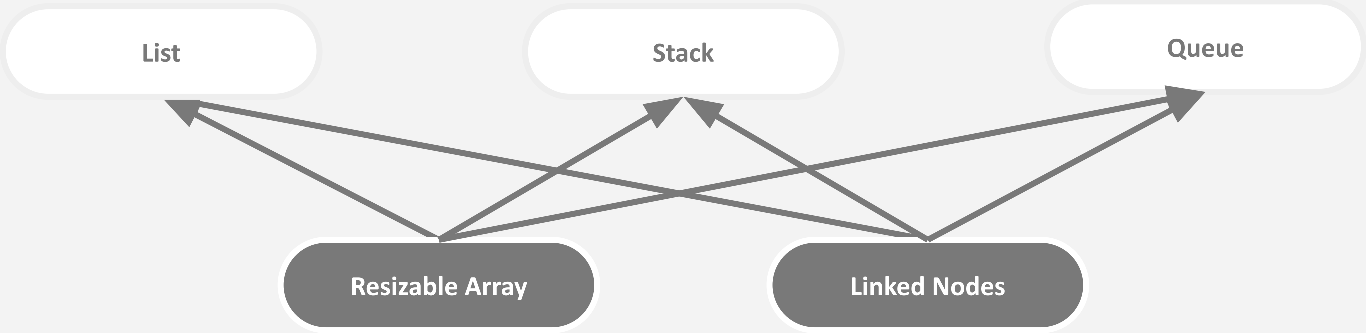
ADTs & Data Structures

- ADT can be implemented by multiple data structures
- Data structure can implement multiple ADTs



ADTs & Data Structures

- ADT can be implemented by multiple data structures
- Data structure can implement multiple ADTs
  - But the ADT decides how it can be used
    - An ArrayList used as a List should support get()
    - An ArrayList used as a Stack should not support get()



سوال؟