

Basic Data Structures: Stacks and Queues

Neil Rhodes

Department of Computer Science and Engineering
University of California, San Diego

Data Structures
Data Structures and Algorithms

Outline

1 Stacks

2 Queues

Definition

Stack: Abstract data type with the following operations:

Definition

Stack: Abstract data type with the following operations:

- **Push(Key):** adds key to collection

Definition

Stack: Abstract data type with the following operations:

- `Push(Key)`: adds key to collection
- `Key Top()`: returns most recently-added key

Definition

Stack: Abstract data type with the following operations:

- **Push(Key):** adds key to collection
- **Key Top():** returns most recently-added key
- **Key Pop():** removes and returns most recently-added key

Definition

LIFO = Last In First Out

Stack: Abstract data type with the following operations:

- `Push(Key)`: adds key to collection
- `Key Top()`: returns most recently-added key
- `Key Pop()`: removes and returns most recently-added key
- `Boolean Empty()`: are there any elements?

Balanced Brackets

Input: A string *str* consisting of '(', ')', '[', ']' characters.

Output: Return whether or not the string's parentheses and square brackets are balanced.

Balanced Brackets

Balanced:

- “([]) [] ()”,
- “((([([])])) ())”

Unbalanced:

- “([]] ()”
- “] [”

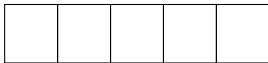
IsBalanced(*str*)

```
Stack stack
for char in str:
    if char in ['(', '[']:
        stack.Push(char)
    else:
        if stack.Empty(): return False
        top ← stack.Pop()
        if (top = '[' and char != ']') or
           (top = '(' and char != ')'):
            return False
return stack.Empty()
```

Stack Implementation with Array

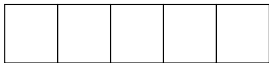


numElements: 0



Stack Implementation with Array

numElements: 0



Push(a)

Stack Implementation with Array

numElements: 1

a				
---	--	--	--	--

Push(a)

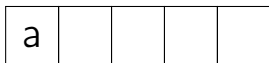
Stack Implementation with Array

numElements: 1

a				
---	--	--	--	--

Stack Implementation with Array

numElements: 1



Push(b)

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Push(b)

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Top()

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Top() \rightarrow b

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Push(c)

Stack Implementation with Array

numElements: 3

a	b	c		
---	---	---	--	--

Push(c)

Stack Implementation with Array

numElements: 3

a	b	c		
---	---	---	--	--

Stack Implementation with Array

numElements: 3

a	b	c		
---	---	---	--	--

Pop()

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Pop() \rightarrow c

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Push(d)

Stack Implementation with Array

numElements: 3

a	b	d		
---	---	---	--	--

Push(d)

Stack Implementation with Array

numElements: 3

a	b	d		
---	---	---	--	--

Stack Implementation with Array

numElements: 3

a	b	d		
---	---	---	--	--

Push(e)

Stack Implementation with Array

numElements: 4

a	b	d	e	
---	---	---	---	--

Push(e)

Stack Implementation with Array

numElements: 4

a	b	d	e	
---	---	---	---	--

Stack Implementation with Array

numElements: 4

a	b	d	e	
---	---	---	---	--

Push(f)

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Push(f)

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Push(g)

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Push(g) → ERROR

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Empty()

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Empty() \rightarrow False

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Stack Implementation with Array

numElements: 5

a	b	d	e	f
---	---	---	---	---

Pop()

Stack Implementation with Array

numElements: 4

a	b	d	e	
---	---	---	---	--

Pop() \rightarrow f

Stack Implementation with Array

numElements: 4

a	b	d	e	
---	---	---	---	--

Stack Implementation with Array

numElements: 4

a	b	d	e	
---	---	---	---	--

Pop()

Stack Implementation with Array

numElements: 3

a	b	d		
---	---	---	--	--

Pop() \rightarrow e

Stack Implementation with Array

numElements: 3

a	b	d		
---	---	---	--	--

Stack Implementation with Array

numElements: 3

a	b	d		
---	---	---	--	--

Pop()

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Pop() \rightarrow d

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Stack Implementation with Array

numElements: 2

a	b			
---	---	--	--	--

Pop()

Stack Implementation with Array

numElements: 1

a				
---	--	--	--	--

Pop() \rightarrow b

Stack Implementation with Array

numElements: 1

a				
---	--	--	--	--

Stack Implementation with Array

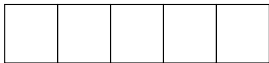
numElements: 1

a				
---	--	--	--	--

Pop()

Stack Implementation with Array

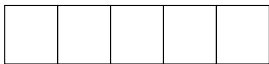
numElements: 0



Pop() \rightarrow a

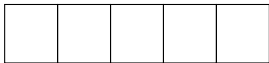
Stack Implementation with Array

numElements: 0



Stack Implementation with Array

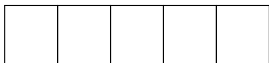
numElements: 0



Empty()

Stack Implementation with Array

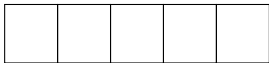
numElements: 0



Empty() \rightarrow True

Stack Implementation with Array

numElements: 0



Stack Implementation with Linked List



head

Stack Implementation with Linked List



head

Push Front == Push()

Pop Front == Pop()

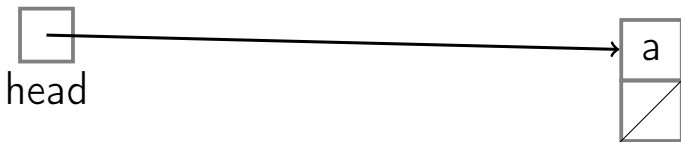
Head == Top()

Singly Linked List (can be with no tail)

$O(1)$

Push(a)

Stack Implementation with Linked List



Push(a)

Stack Implementation with Linked List



Stack Implementation with Linked List



Push(b)

Stack Implementation with Linked List



Push(b)

Stack Implementation with Linked List



Stack Implementation with Linked List



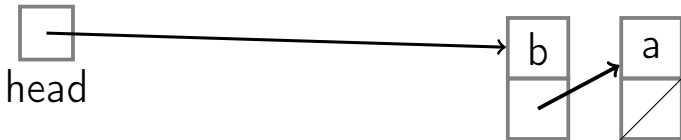
Top()

Stack Implementation with Linked List

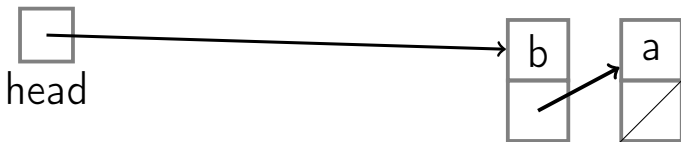


$\text{Top}() \rightarrow b$

Stack Implementation with Linked List

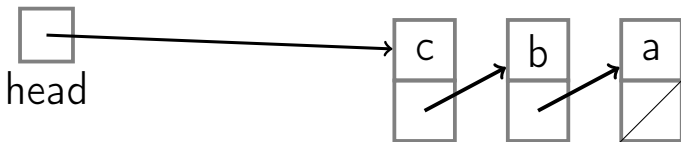


Stack Implementation with Linked List



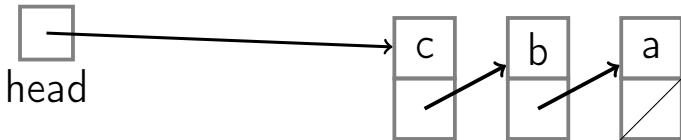
Push(c)

Stack Implementation with Linked List

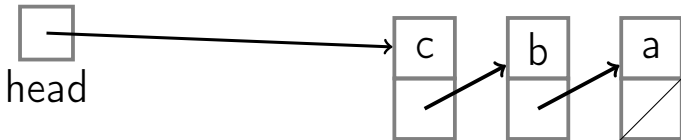


Push(c)

Stack Implementation with Linked List



Stack Implementation with Linked List



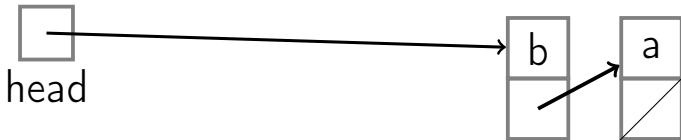
Pop()

Stack Implementation with Linked List



Pop() \rightarrow c

Stack Implementation with Linked List

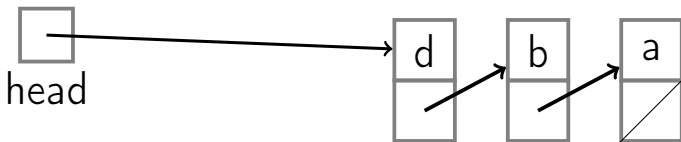


Stack Implementation with Linked List



Push(d)

Stack Implementation with Linked List

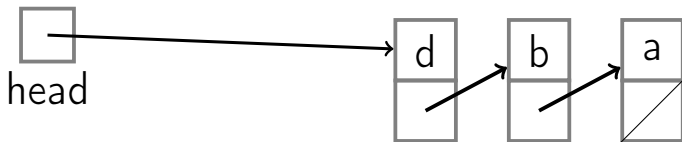


Push(d)

Stack Implementation with Linked List

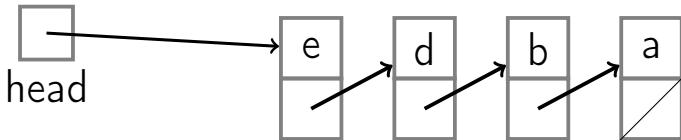


Stack Implementation with Linked List



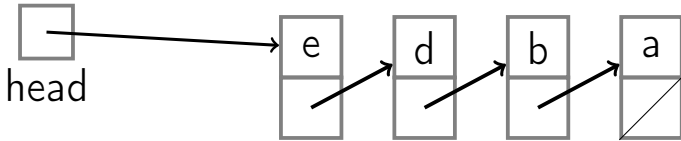
Push(e)

Stack Implementation with Linked List

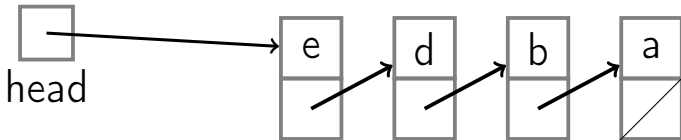


Push(e)

Stack Implementation with Linked List

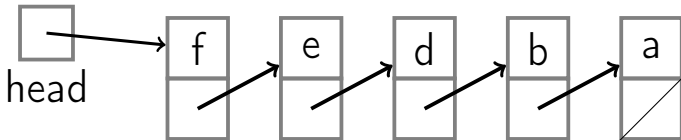


Stack Implementation with Linked List



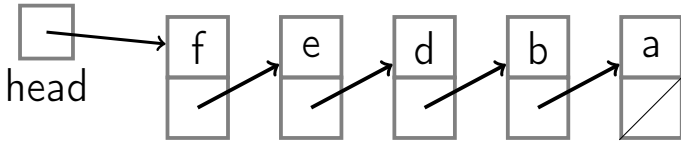
Push(f)

Stack Implementation with Linked List

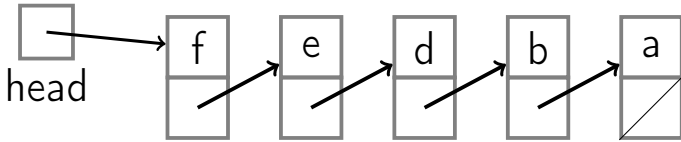


Push(f)

Stack Implementation with Linked List

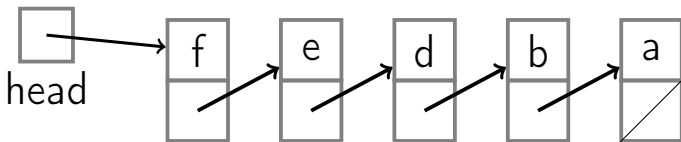


Stack Implementation with Linked List



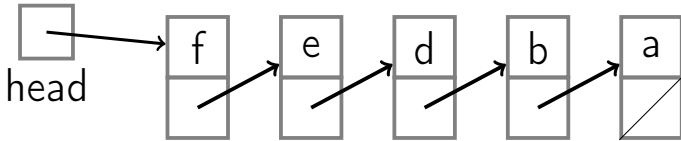
Empty()

Stack Implementation with Linked List

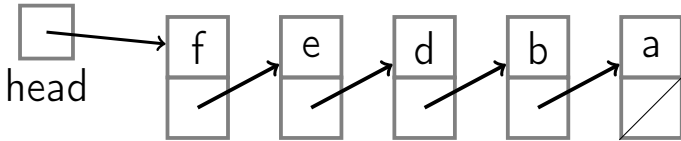


`Empty()` \rightarrow `False`

Stack Implementation with Linked List

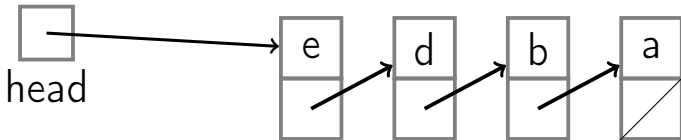


Stack Implementation with Linked List



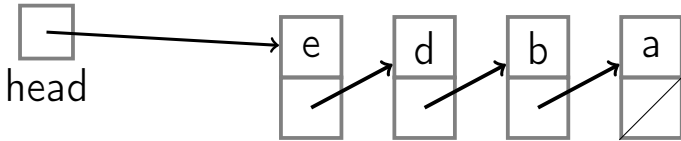
Pop()

Stack Implementation with Linked List

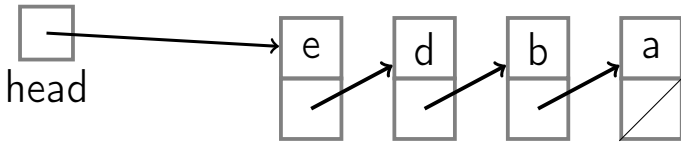


Pop() → f

Stack Implementation with Linked List

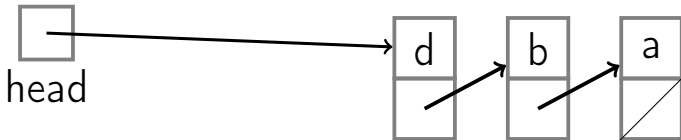


Stack Implementation with Linked List



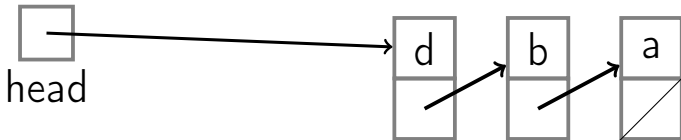
Pop()

Stack Implementation with Linked List



Pop() → e

Stack Implementation with Linked List

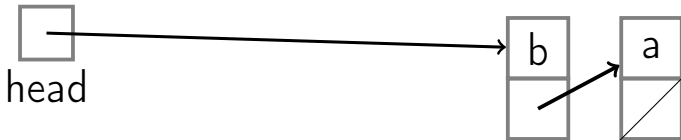


Stack Implementation with Linked List



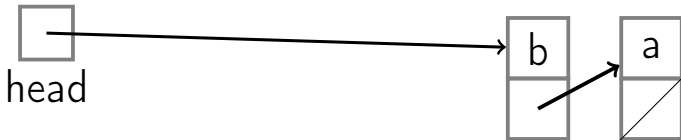
Pop()

Stack Implementation with Linked List



Pop() \rightarrow d

Stack Implementation with Linked List



Stack Implementation with Linked List



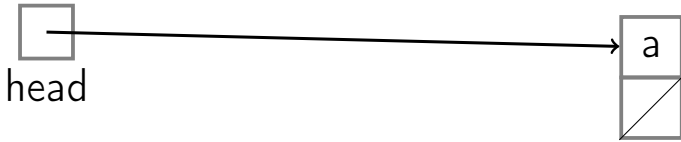
Pop()

Stack Implementation with Linked List



Pop() \rightarrow b

Stack Implementation with Linked List



Stack Implementation with Linked List



Pop()

Stack Implementation with Linked List



Pop() \rightarrow a

Stack Implementation with Linked List



head

Stack Implementation with Linked List



head

Empty()

Stack Implementation with Linked List



head

`Empty() → True`

Stack Implementation with Linked List



head

Stack Implementation with Linked List

Stack Implementation with Linked List

Stack Implementation with Linked List

Summary

- Stacks can be implemented with either an array or a linked list.

Summary

- Stacks can be implemented with either an array or a linked list.
- Each stack operation is $O(1)$: Push, Pop, Top, Empty.

Summary

- Stacks can be implemented with either an array or a linked list.
- Each stack operation is $O(1)$: Push, Pop, Top, Empty.
- Stacks are occasionally known as LIFO queues.

Outline

1 Stacks

2 Queues

Definition

Queue: Abstract data type with the following operations:

Definition

Queue: Abstract data type with the following operations:

- **Enqueue(Key):** adds key to collection

Definition

Queue: Abstract data type with the following operations:

- `Enqueue(Key)`: adds key to collection
- `Key Dequeue()`: removes and returns least recently-added key

Definition

Queue: Abstract data type with the following operations:

- `Enqueue(Key)`: adds key to collection
- `Key Dequeue()`: removes and returns least recently-added key
- `Boolean Empty()`: are there any elements?

Definition

Queue: Abstract data type with the following operations:

- `Enqueue(Key)`: adds key to collection
- `Key Dequeue()`: removes and returns least recently-added key
- `Boolean Empty()`: are there any elements?

FIFO: First-In, First-Out

Queue Implementation with Linked List



head



tail

Singly Linked List with Tail

Push Back == $O(1)$ Enqueue;

Pop Front + TopFront == $O(1)$ Dequeue;

Doubly Linked List with Tail

Push Front == $O(1)$ Enqueue;

Pop Back + TopBack == $O(1)$ Dequeue;

Queue Implementation with Linked List



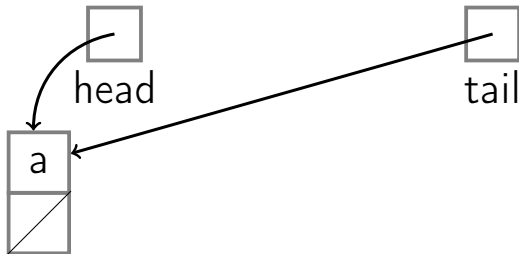
head



tail

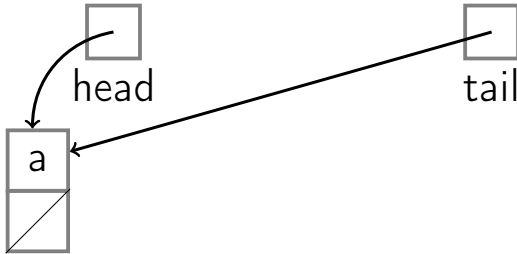
Enqueue(a)

Queue Implementation with Linked List

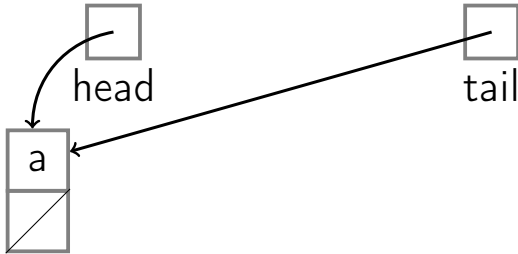


Enqueue(a)

Queue Implementation with Linked List

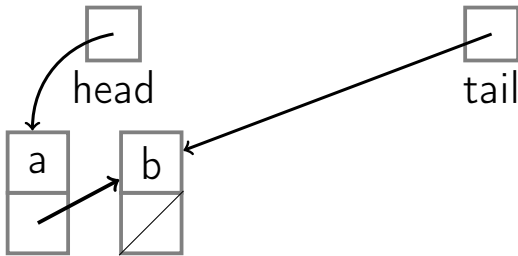


Queue Implementation with Linked List



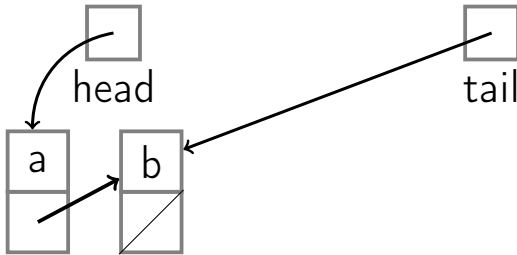
Enqueue(b)

Queue Implementation with Linked List

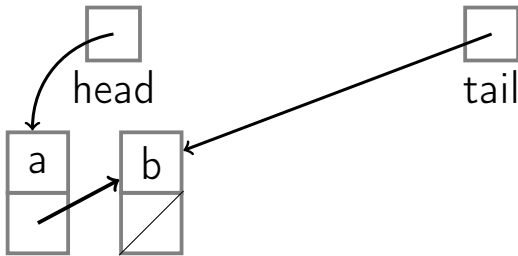


Enqueue(b)

Queue Implementation with Linked List

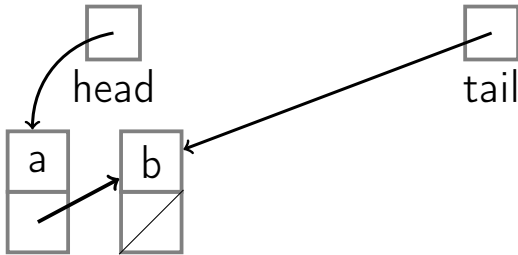


Queue Implementation with Linked List



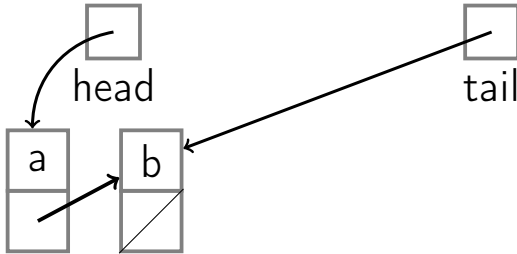
Empty()

Queue Implementation with Linked List

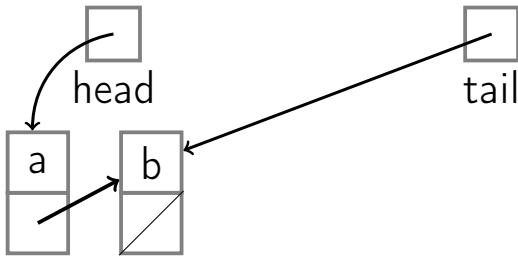


`Empty() → False`

Queue Implementation with Linked List

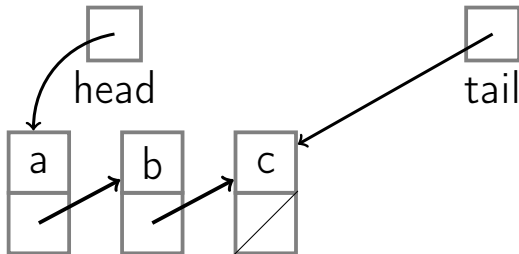


Queue Implementation with Linked List



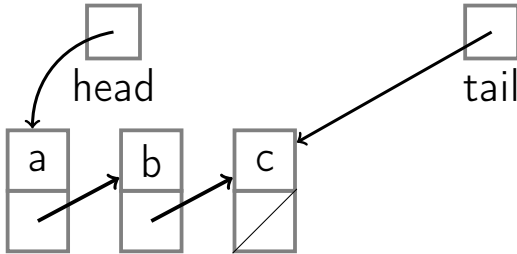
Enqueue(c)

Queue Implementation with Linked List

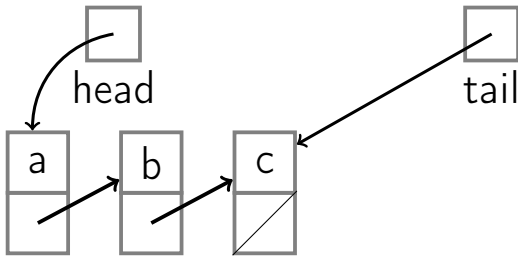


Enqueue(c)

Queue Implementation with Linked List

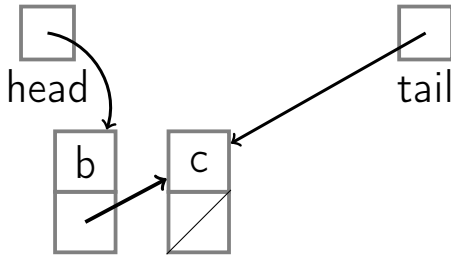


Queue Implementation with Linked List



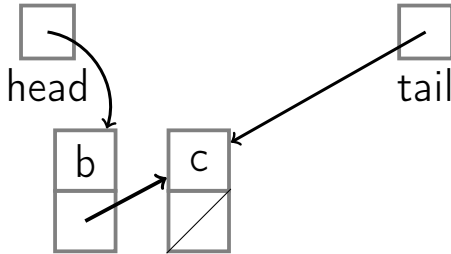
Dequeue()

Queue Implementation with Linked List

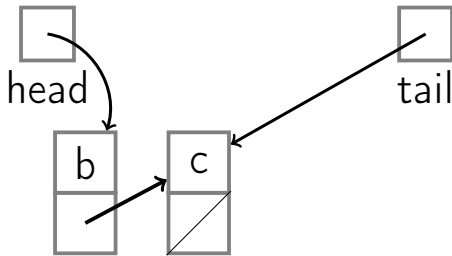


Dequeue() \rightarrow a

Queue Implementation with Linked List

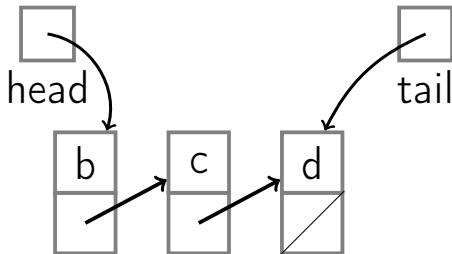


Queue Implementation with Linked List



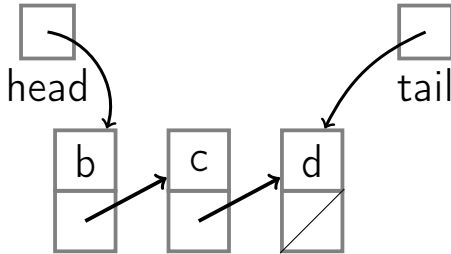
Enqueue(d)

Queue Implementation with Linked List

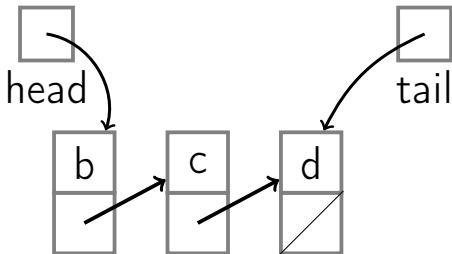


Enqueue(d)

Queue Implementation with Linked List

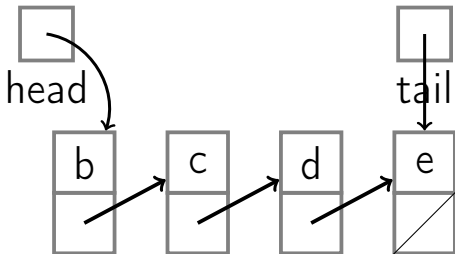


Queue Implementation with Linked List



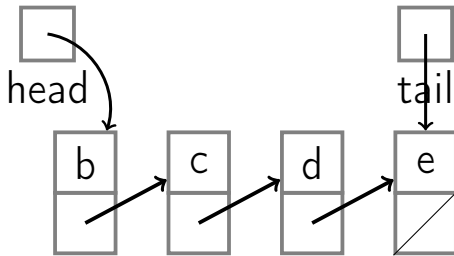
Enqueue(e)

Queue Implementation with Linked List

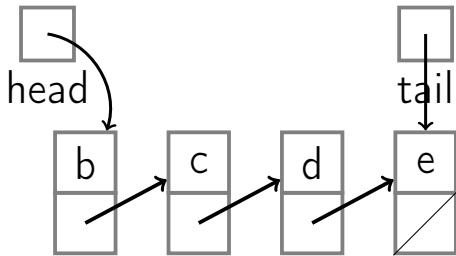


Enqueue(e)

Queue Implementation with Linked List

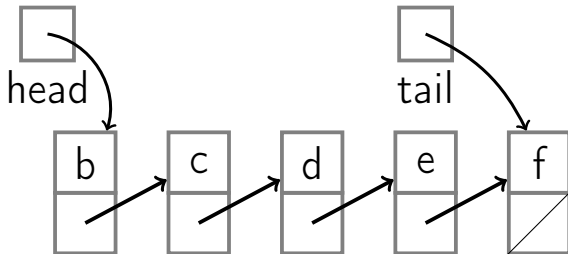


Queue Implementation with Linked List



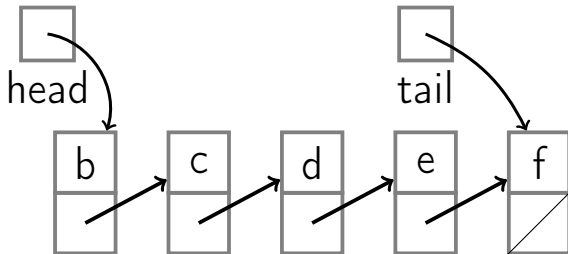
Enqueue(f)

Queue Implementation with Linked List

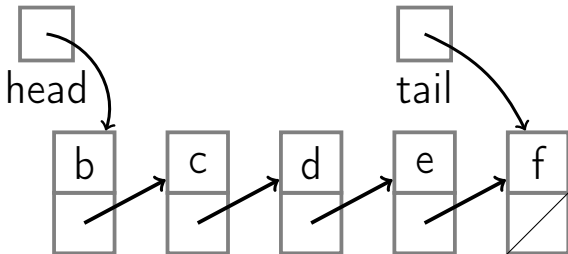


Enqueue(f)

Queue Implementation with Linked List

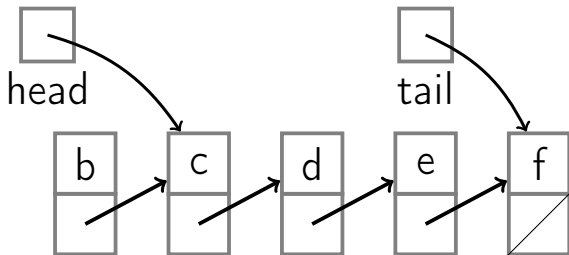


Queue Implementation with Linked List



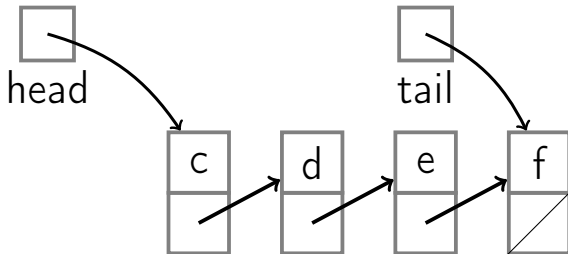
Dequeue()

Queue Implementation with Linked List

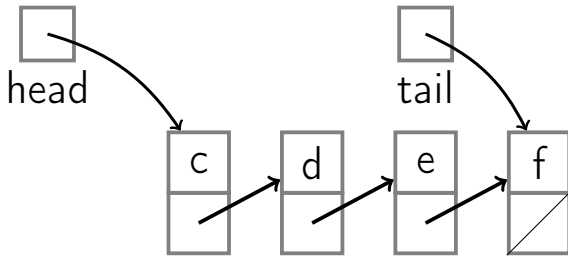


Dequeue() → b

Queue Implementation with Linked List

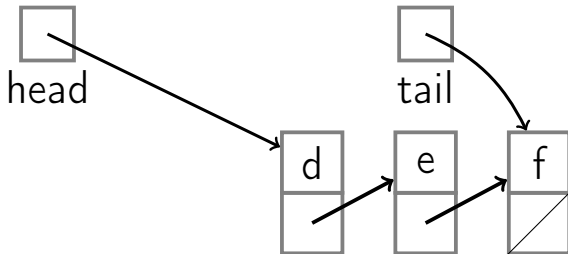


Queue Implementation with Linked List



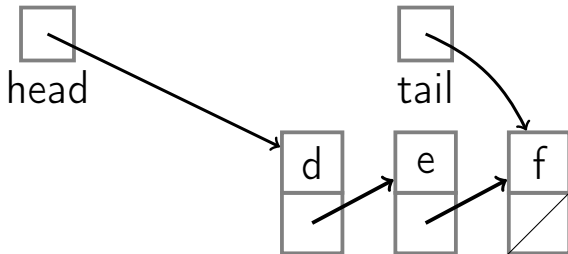
Dequeue()

Queue Implementation with Linked List

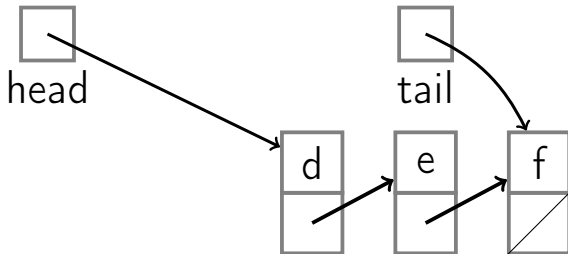


Dequeue() \rightarrow c

Queue Implementation with Linked List

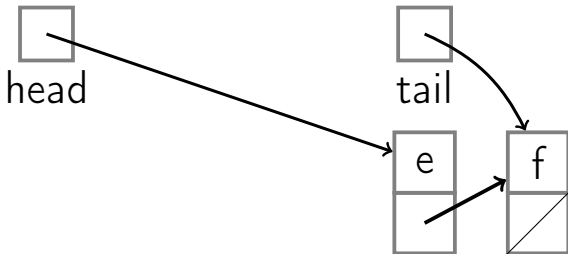


Queue Implementation with Linked List



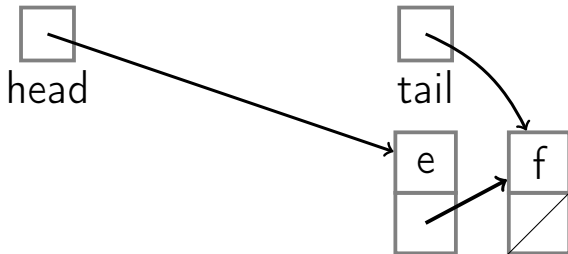
Dequeue()

Queue Implementation with Linked List

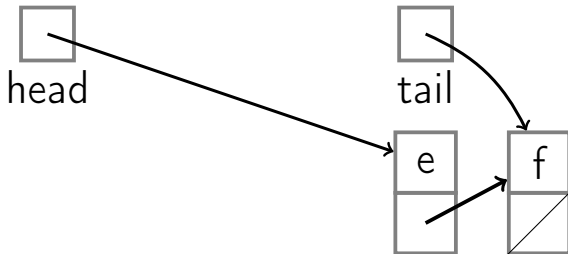


`Dequeue()` \rightarrow `d`

Queue Implementation with Linked List

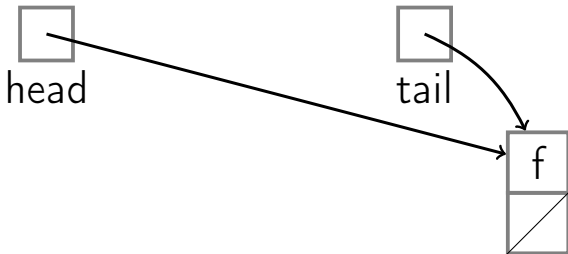


Queue Implementation with Linked List



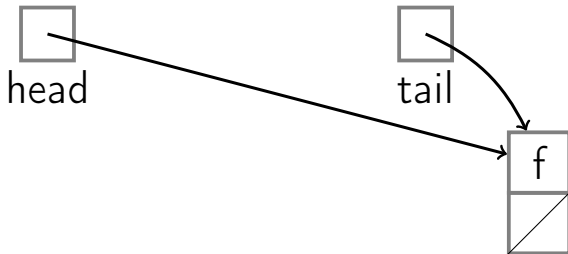
`Dequeue()`

Queue Implementation with Linked List

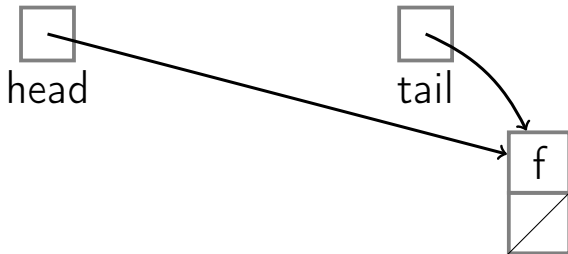


`Dequeue()` \rightarrow e

Queue Implementation with Linked List



Queue Implementation with Linked List



Dequeue()

Queue Implementation with Linked List



head



tail

Dequeue() \rightarrow f

Queue Implementation with Linked List



head



tail

Queue Implementation with Linked List



head



tail

Empty()

Queue Implementation with Linked List



head



tail

`Empty()` \rightarrow `True`

Queue Implementation with Linked List



head



tail

Queue Implementation with Linked List

- Enqueue: use `List.PushBack`

Queue Implementation with Linked List

- Enqueue: use `List.PushBack`
- Dequeue: use `List.TopFront` and `List.PopFront`

Queue Implementation with Linked List

- Enqueue: use `List.PushBack`
- Dequeue: use `List.TopFront` and `List.PopFront`
- Empty: use `List.Empty`

Queue Implementation with Array

Dequeue return arr[read]

read++

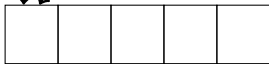
0
read

Enqueue arr[write]=element

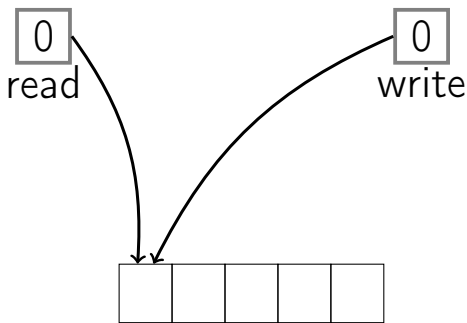
write++

0
write

read == write --> Empty

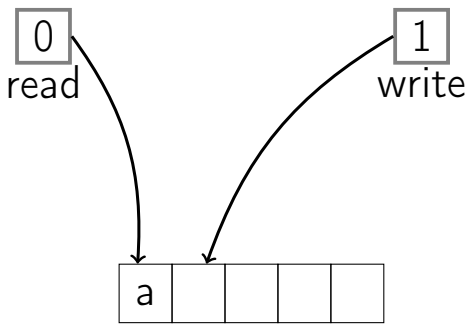


Queue Implementation with Array



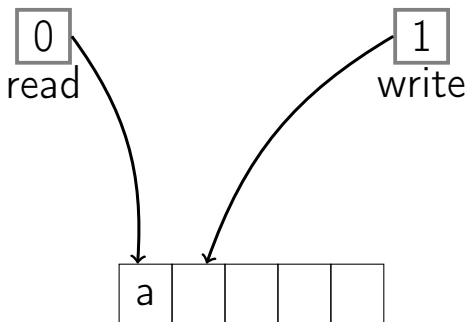
Enqueue(a)

Queue Implementation with Array

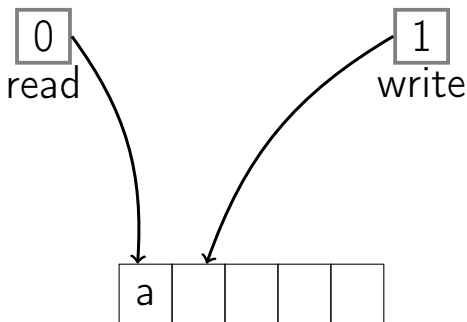


Enqueue(a)

Queue Implementation with Array

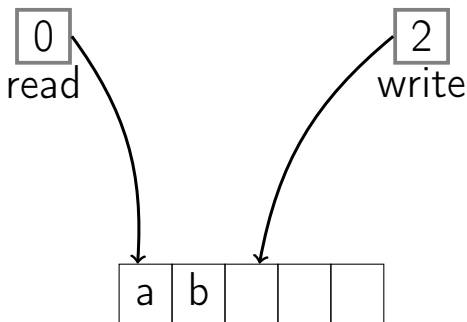


Queue Implementation with Array



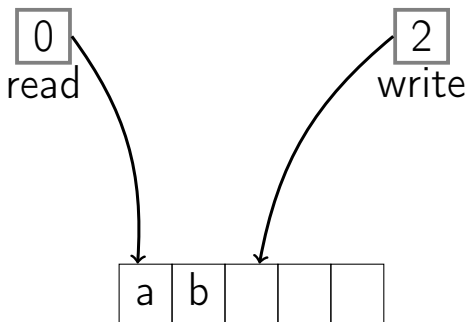
Enqueue(b)

Queue Implementation with Array

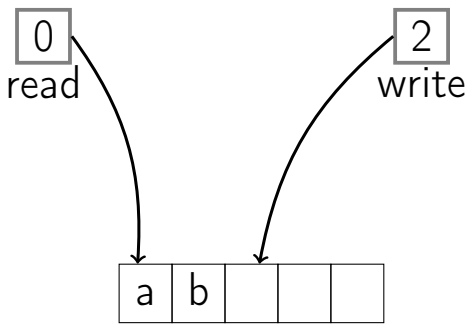


Enqueue(b)

Queue Implementation with Array

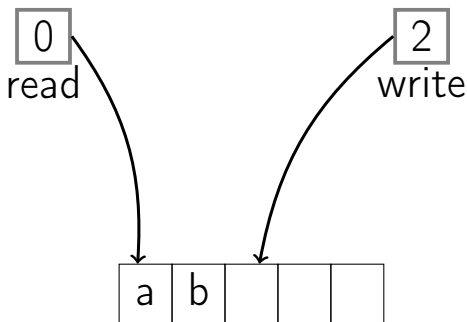


Queue Implementation with Array



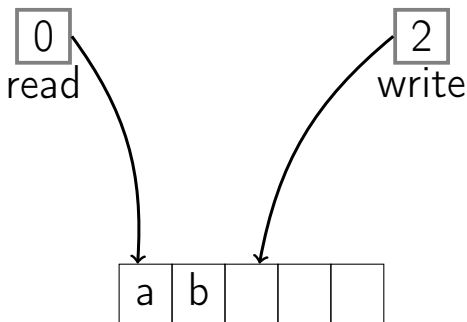
Empty()

Queue Implementation with Array

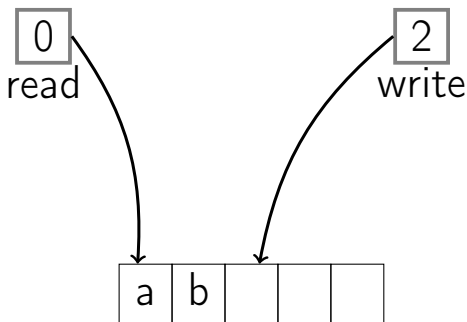


`Empty()` \rightarrow `False`

Queue Implementation with Array

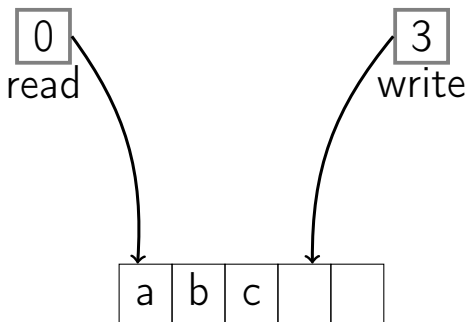


Queue Implementation with Array



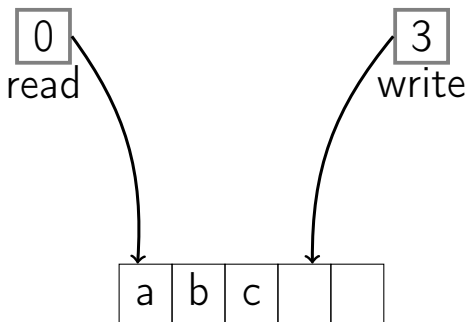
Enqueue(c)

Queue Implementation with Array

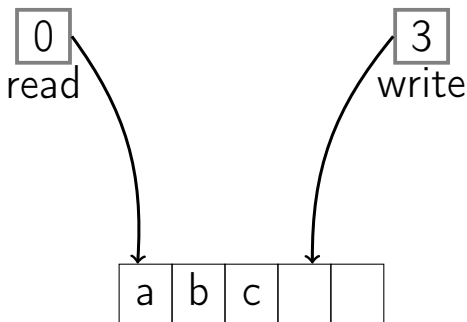


Enqueue(c)

Queue Implementation with Array

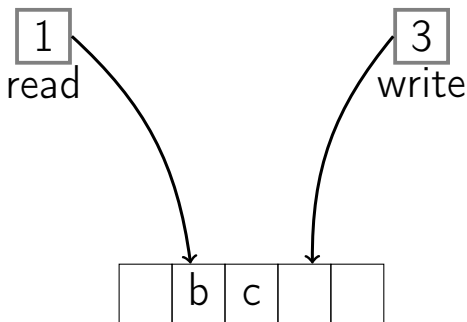


Queue Implementation with Array



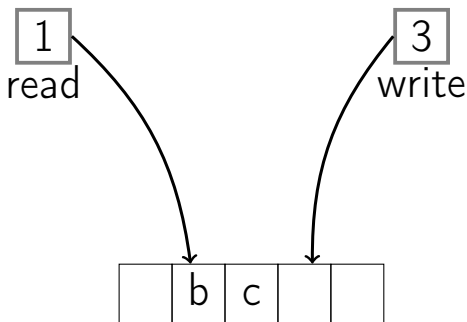
Dequeue()

Queue Implementation with Array

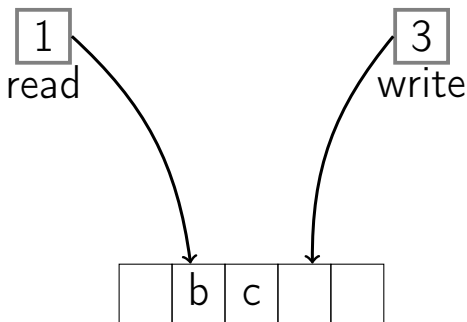


Dequeue() \rightarrow a

Queue Implementation with Array

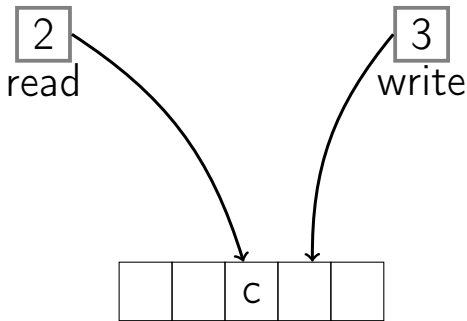


Queue Implementation with Array



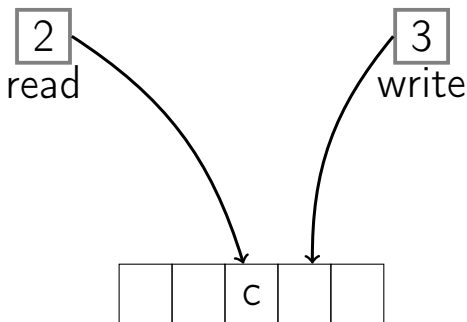
Dequeue()

Queue Implementation with Array

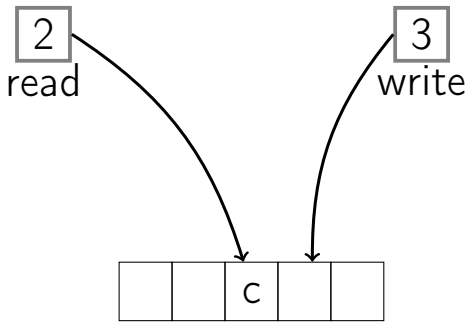


Dequeue() \rightarrow b

Queue Implementation with Array

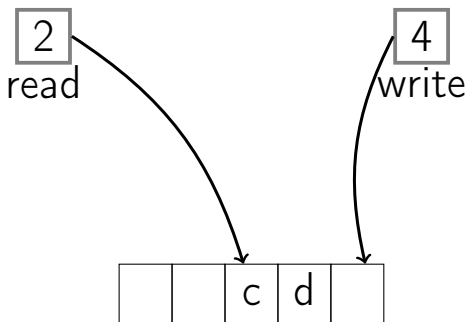


Queue Implementation with Array



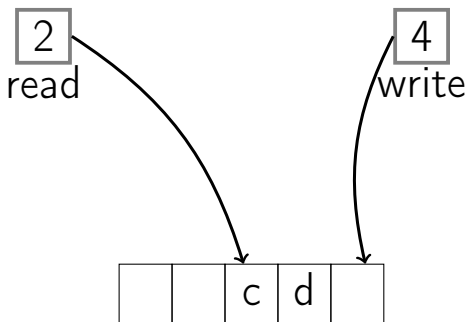
Enqueue(d)

Queue Implementation with Array

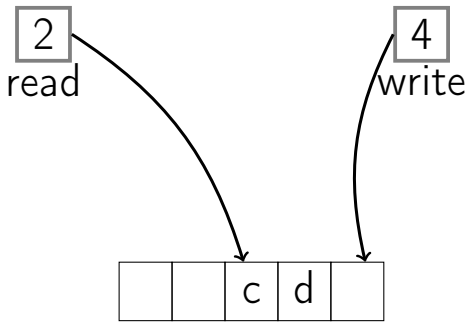


Enqueue(d)

Queue Implementation with Array

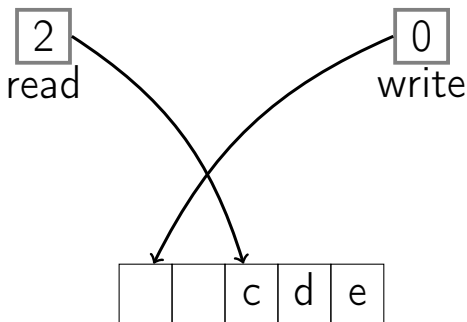


Queue Implementation with Array



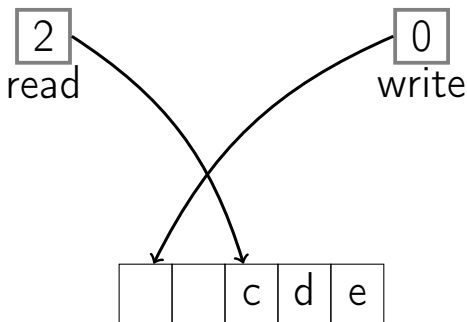
Enqueue(e)

Queue Implementation with Array

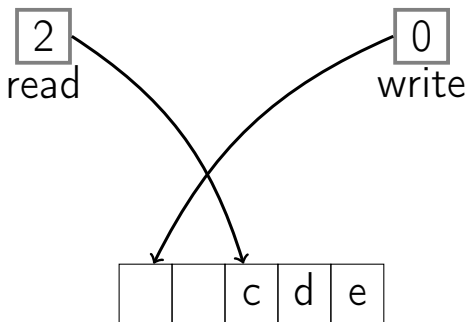


Enqueue(e)

Queue Implementation with Array

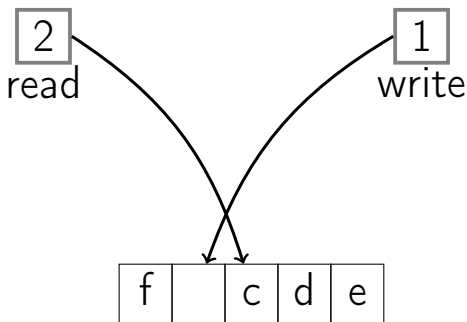


Queue Implementation with Array



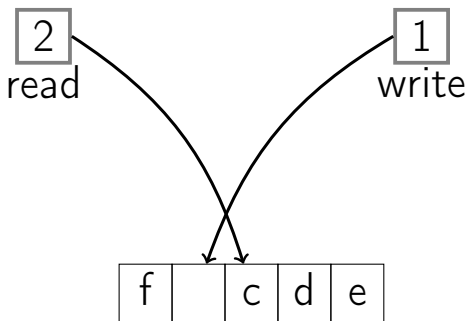
Enqueue(f)

Queue Implementation with Array

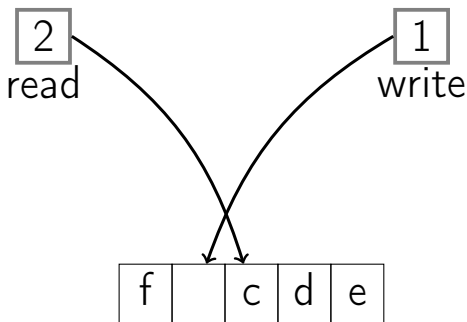


Enqueue(f)

Queue Implementation with Array

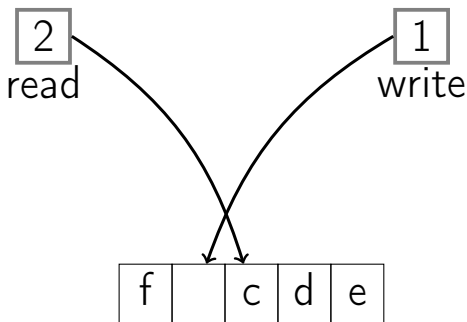


Queue Implementation with Array



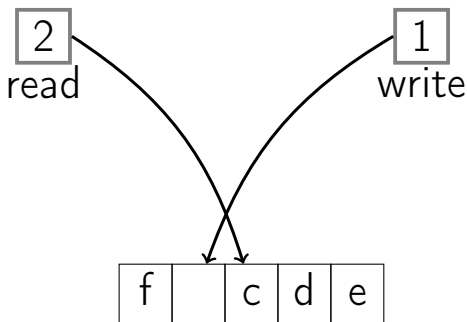
Enqueue(g)

Queue Implementation with Array

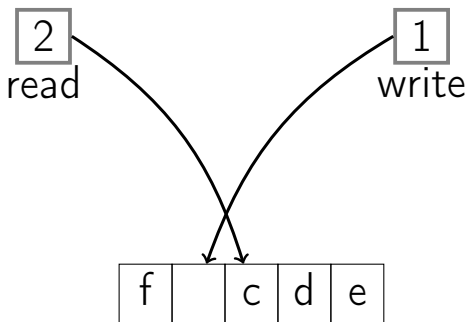


Enqueue(g) \rightarrow ERROR

Queue Implementation with Array

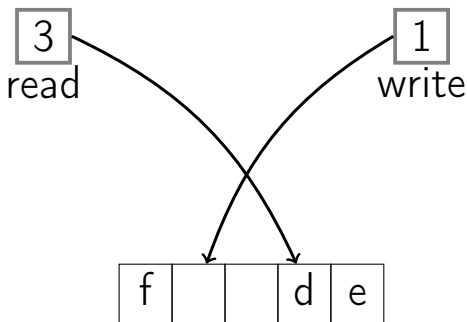


Queue Implementation with Array



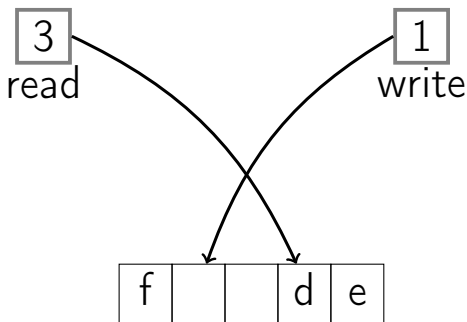
Dequeue()

Queue Implementation with Array

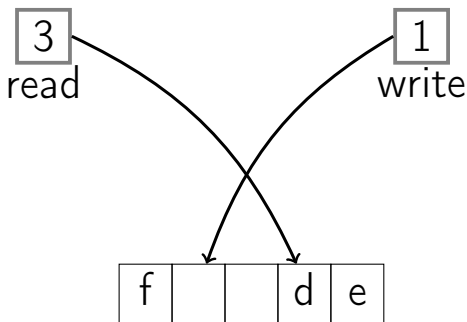


Dequeue() \rightarrow c

Queue Implementation with Array

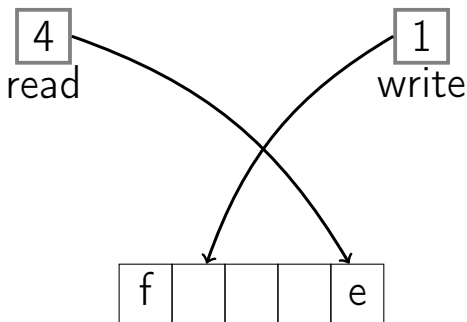


Queue Implementation with Array



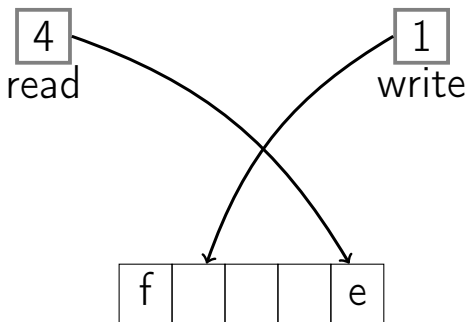
Dequeue()

Queue Implementation with Array

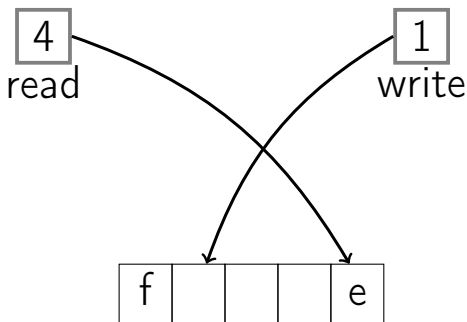


Dequeue() \rightarrow d

Queue Implementation with Array

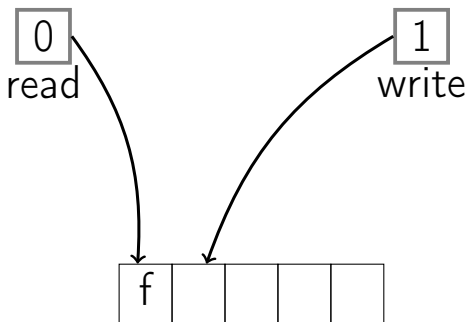


Queue Implementation with Array



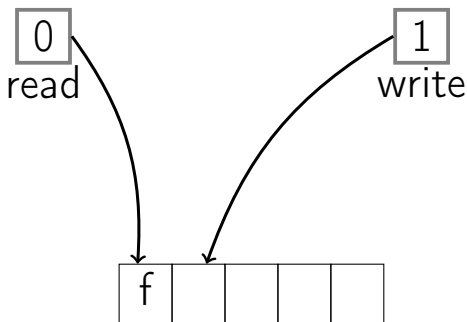
Dequeue()

Queue Implementation with Array

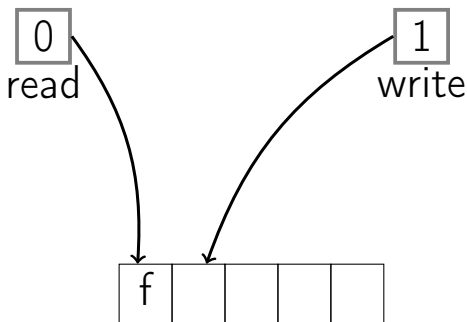


Dequeue() \rightarrow e

Queue Implementation with Array

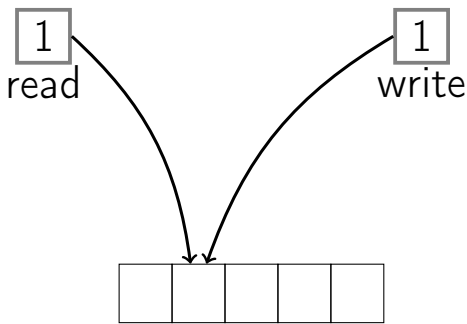


Queue Implementation with Array



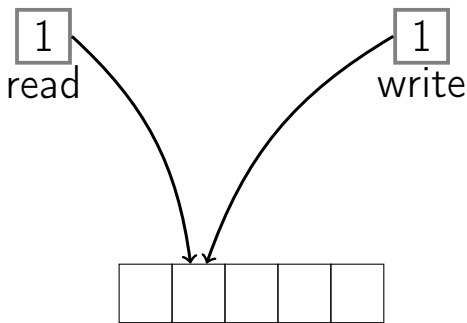
Dequeue()

Queue Implementation with Array

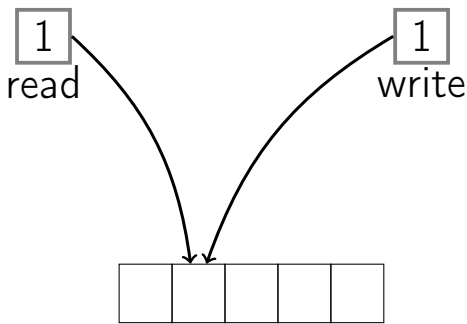


Dequeue() \rightarrow f

Queue Implementation with Array

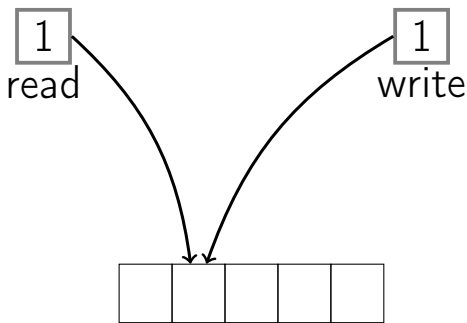


Queue Implementation with Array



Empty()

Queue Implementation with Array



`Empty() → True`

Summary

Summary

- Queues can be implemented with either a linked list (with tail pointer) or an array.

Summary

- Queues can be implemented with either a linked list (with tail pointer) or an array.
- Each queue operation is $O(1)$: Enqueue, Dequeue, Empty.