Basic Data Structures: Stacks and Queues

Neil Rhodes

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

Data Structures Fundamentals
Algorithms and Data Structures

Outline

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

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:

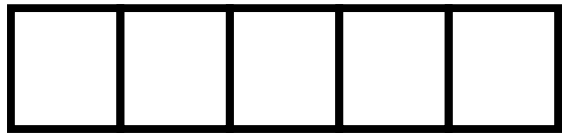
- ``([]]()''
- ``][''

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()
```

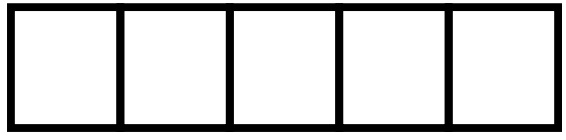
Stack Implementation with Array

numElements: 0



Stack Implementation with Array

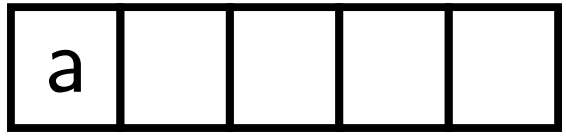
numElements: 0



`Push(a)`

Stack Implementation with Array

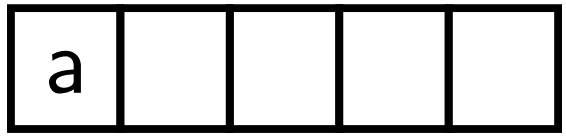
numElements: 1



`Push(a)`

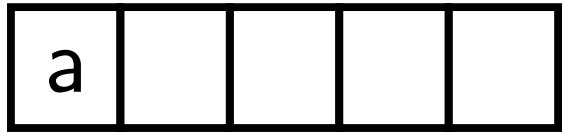
Stack Implementation with Array

numElements: 1



Stack Implementation with Array

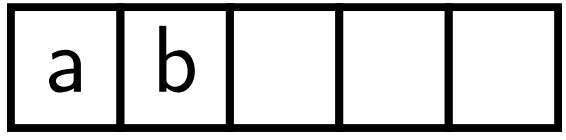
numElements: 1



`Push(b)`

Stack Implementation with Array

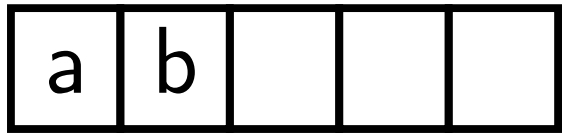
numElements: 2



`Push(b)`

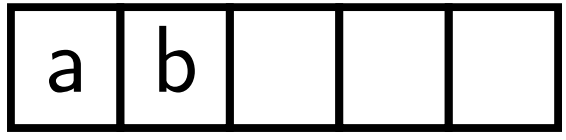
Stack Implementation with Array

numElements: 2



Stack Implementation with Array

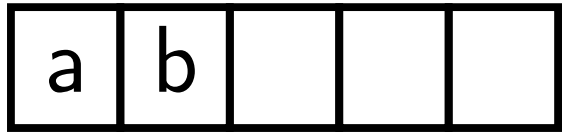
numElements: 2



`Top()`

Stack Implementation with Array

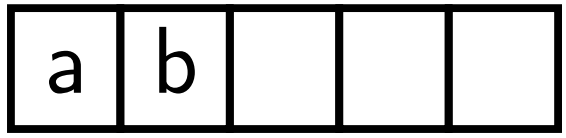
numElements: 2



`Top()` $\rightarrow$ `b`

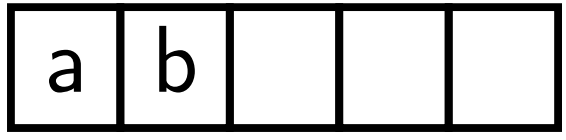
Stack Implementation with Array

numElements: 2



Stack Implementation with Array

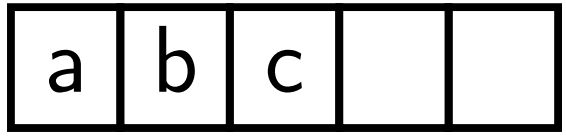
numElements: 2



`Push(c)`

Stack Implementation with Array

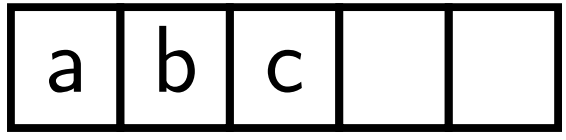
numElements: 3



`Push(c)`

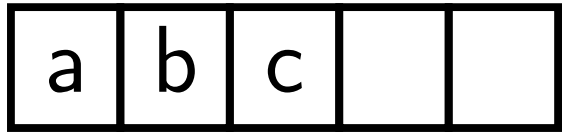
Stack Implementation with Array

numElements: 3



Stack Implementation with Array

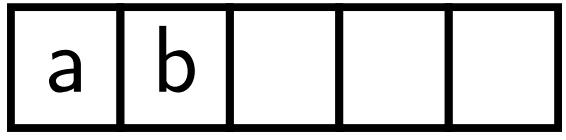
numElements: 3



`Pop()`

Stack Implementation with Array

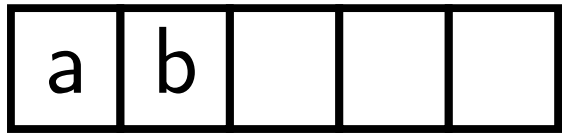
numElements: 2



`Pop()` $\rightarrow$ `c`

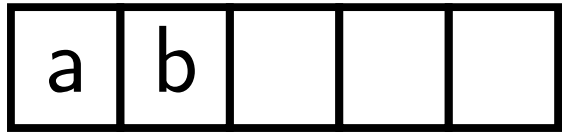
Stack Implementation with Array

numElements: 2



Stack Implementation with Array

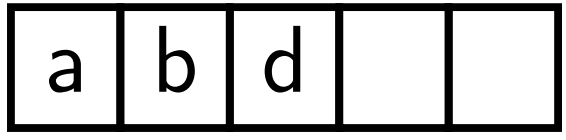
numElements: 2



`Push(d)`

Stack Implementation with Array

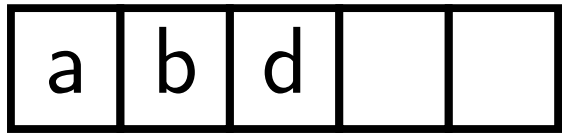
numElements: 3



`Push(d)`

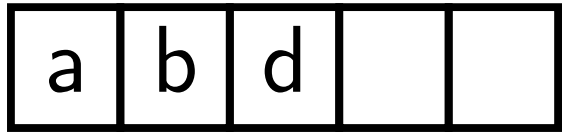
Stack Implementation with Array

numElements: 3



Stack Implementation with Array

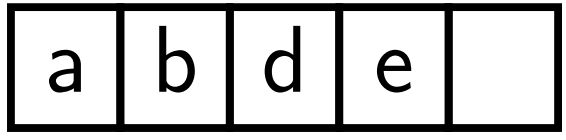
numElements: 3



`Push(e)`

Stack Implementation with Array

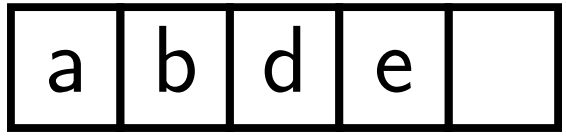
numElements: 4



`Push(e)`

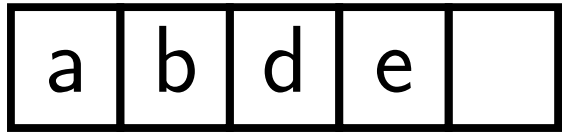
Stack Implementation with Array

numElements: 4



Stack Implementation with Array

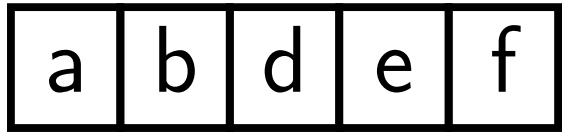
numElements: 4



`Push(f)`

Stack Implementation with Array

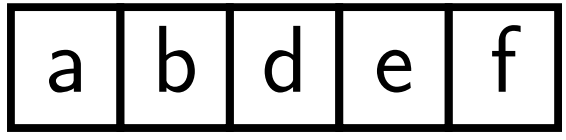
numElements: 5



`Push(f)`

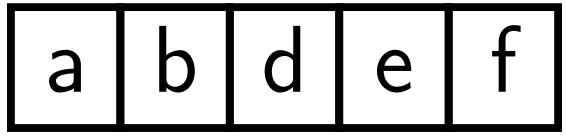
Stack Implementation with Array

numElements: 5



Stack Implementation with Array

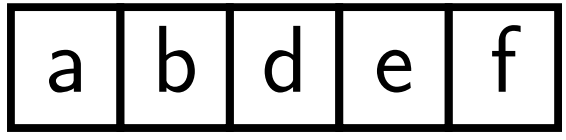
numElements: 5



`Push(g)`

Stack Implementation with Array

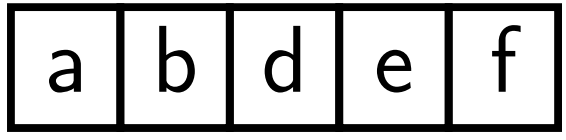
numElements: 5



`Push(g)` $\rightarrow$ `ERROR`

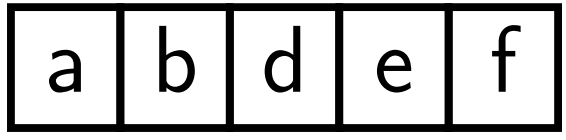
Stack Implementation with Array

numElements: 5



Stack Implementation with Array

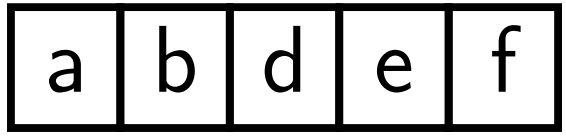
numElements: 5



`Empty()`

Stack Implementation with Array

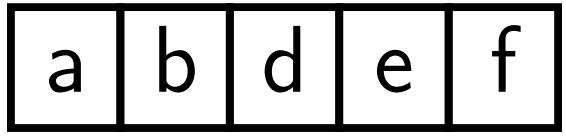
numElements: 5



`Empty()` $\rightarrow$ `False`

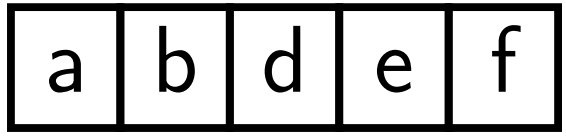
Stack Implementation with Array

numElements: 5



Stack Implementation with Array

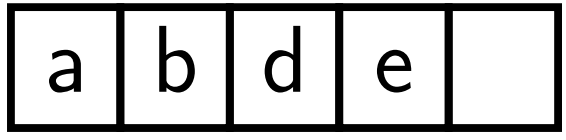
numElements: 5



`Pop()`

Stack Implementation with Array

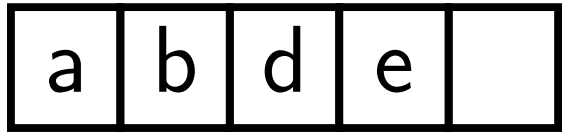
numElements: 4



`Pop()` $\rightarrow$ `f`

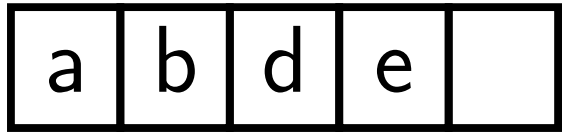
Stack Implementation with Array

numElements: 4



Stack Implementation with Array

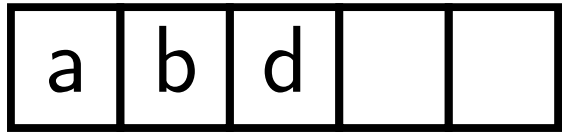
numElements: 4



`Pop()`

Stack Implementation with Array

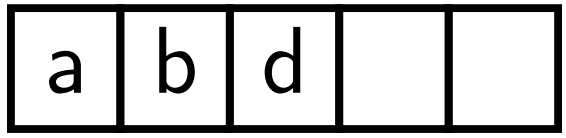
numElements: 3



`Pop()` $\rightarrow$ `e`

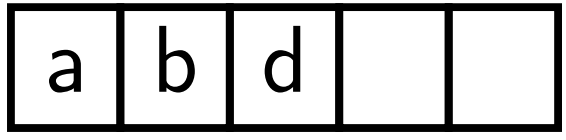
Stack Implementation with Array

numElements: 3



Stack Implementation with Array

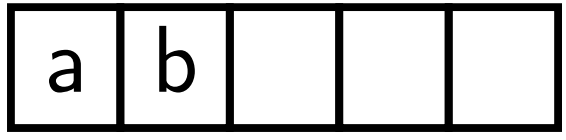
numElements: 3



`Pop()`

Stack Implementation with Array

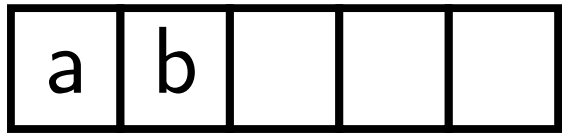
numElements: 2



`Pop()` $\rightarrow$ `d`

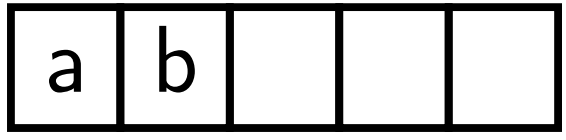
Stack Implementation with Array

numElements: 2



Stack Implementation with Array

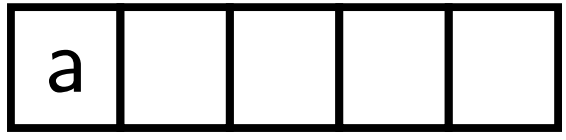
numElements: 2



`Pop()`

Stack Implementation with Array

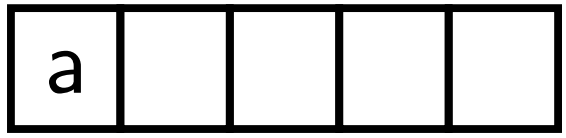
numElements: 1



`Pop()` $\rightarrow$ `b`

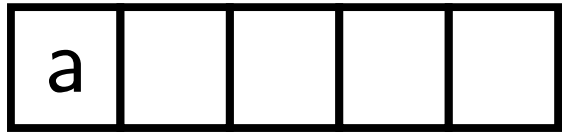
Stack Implementation with Array

numElements: 1



Stack Implementation with Array

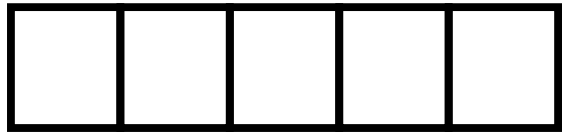
numElements: 1



`Pop()`

Stack Implementation with Array

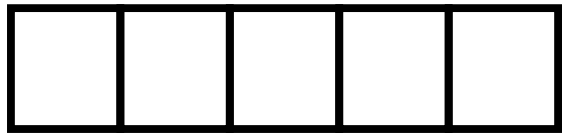
numElements: 0



`Pop()` $\rightarrow$ `a`

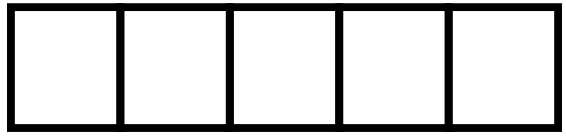
Stack Implementation with Array

numElements: 0



Stack Implementation with Array

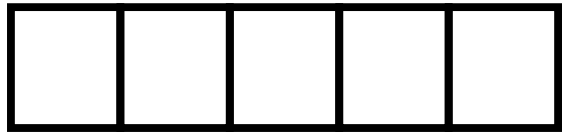
numElements: 0



`Empty()`

Stack Implementation with Array

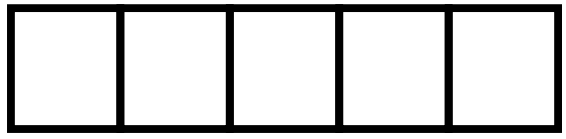
numElements: 0



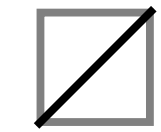
`Empty()` $\rightarrow$ `True`

Stack Implementation with Array

numElements: 0

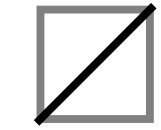


Stack Implementation with Linked List



head

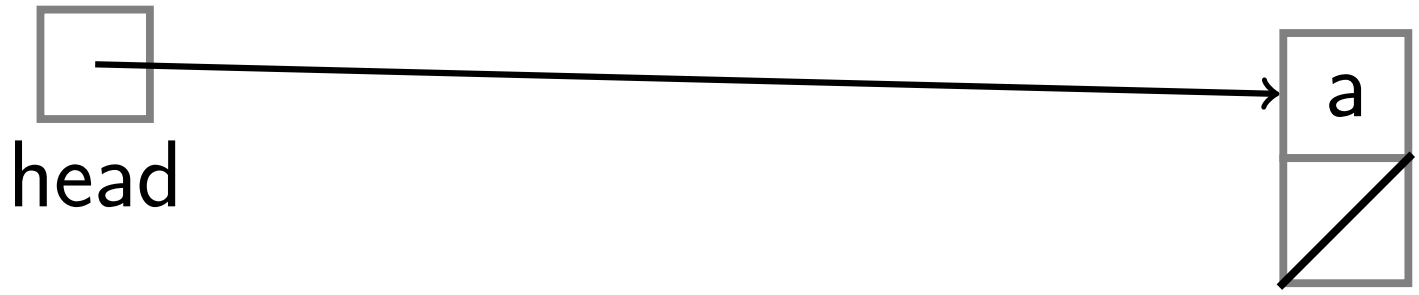
Stack Implementation with Linked List



head

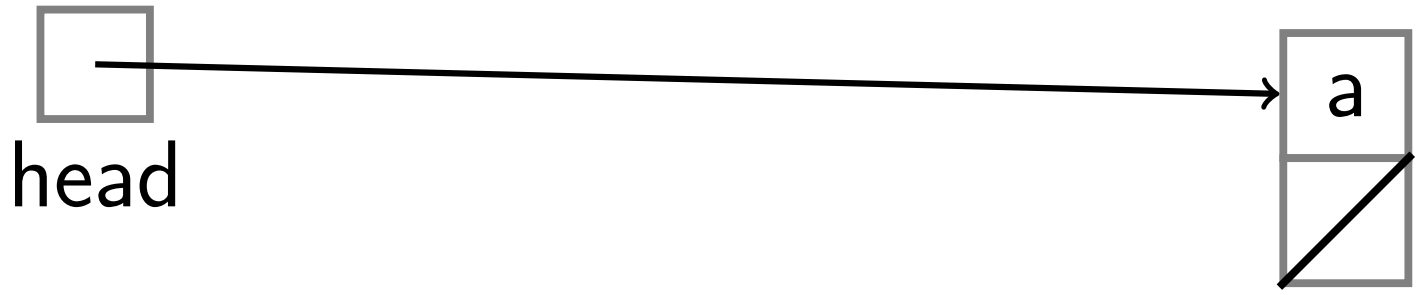
`Push(a)`

Stack Implementation with Linked List

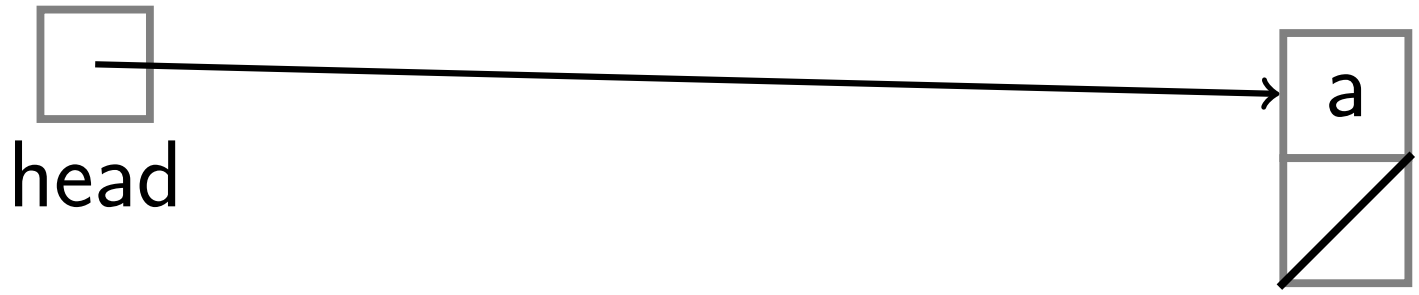


`Push(a)`

Stack Implementation with Linked List

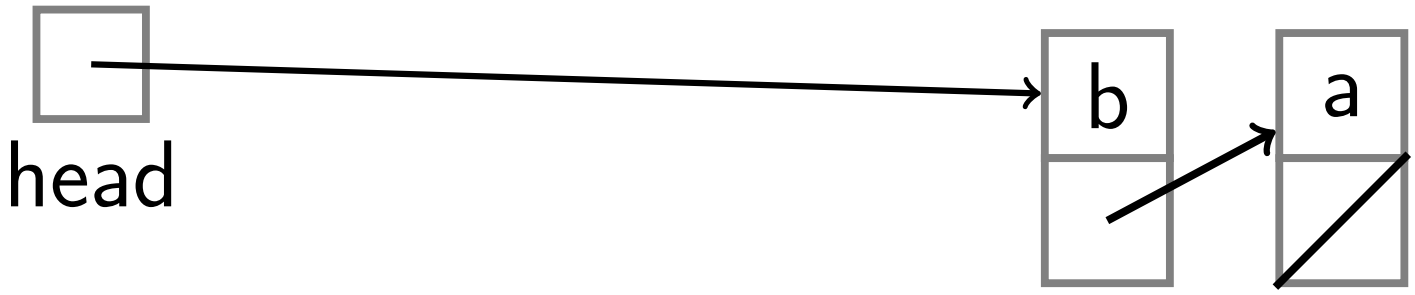


Stack Implementation with Linked List



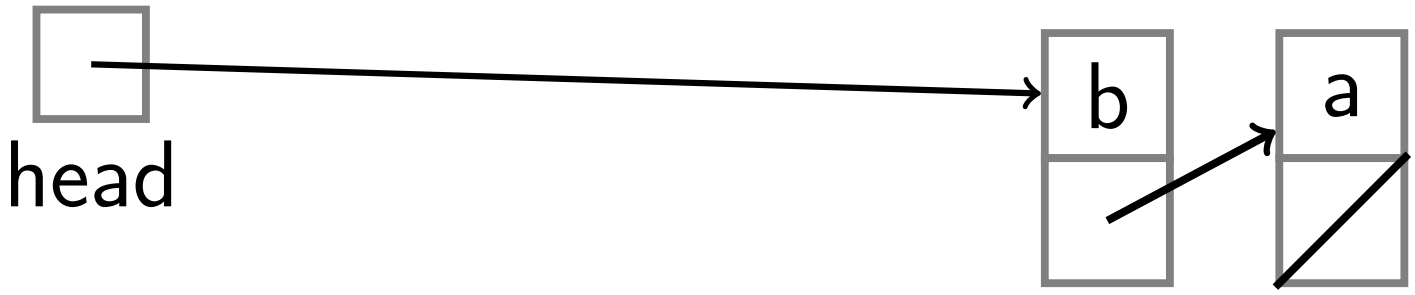
`Push(b)`

Stack Implementation with Linked List

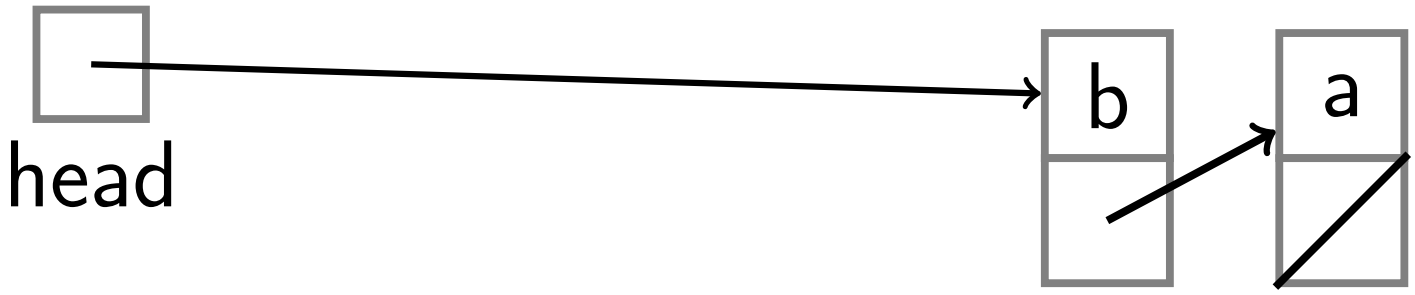


`Push(b)`

Stack Implementation with Linked List

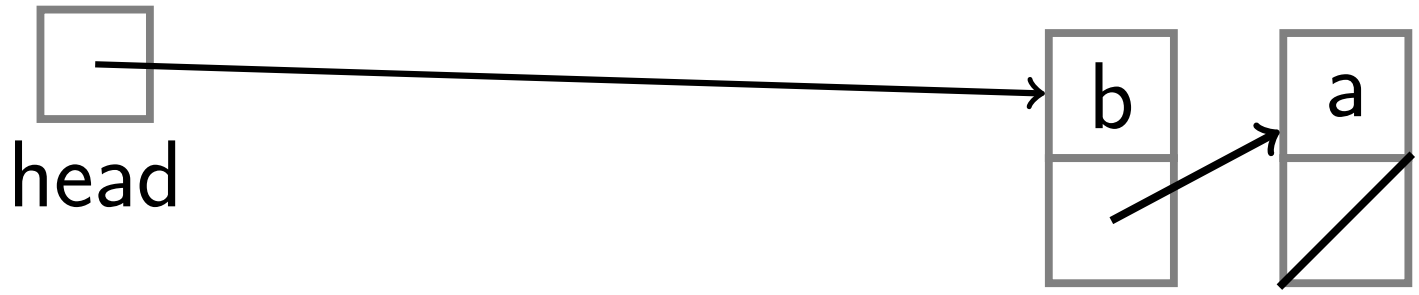


Stack Implementation with Linked List



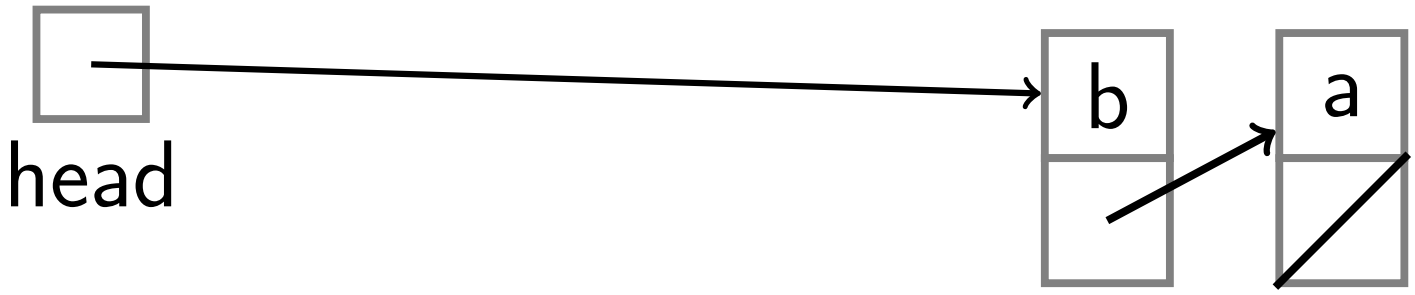
`Top()`

Stack Implementation with Linked List

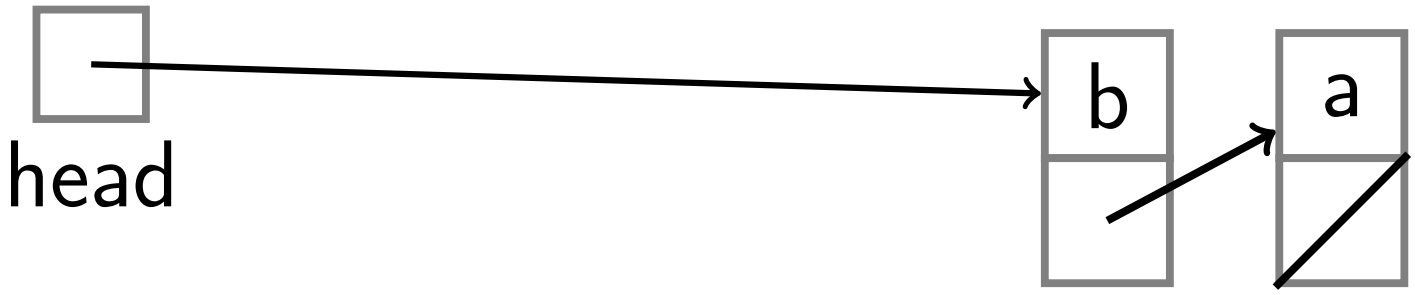


`Top()` $\rightarrow$ `b`

Stack Implementation with Linked List

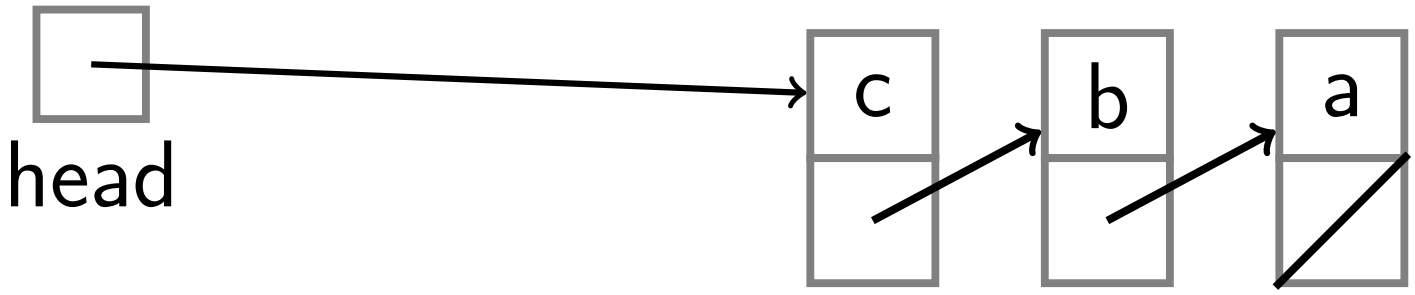


Stack Implementation with Linked List



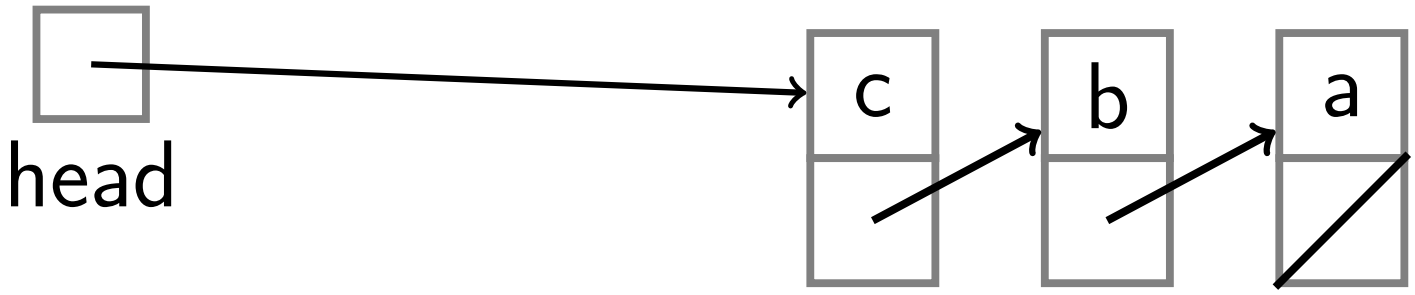
`Push(c)`

Stack Implementation with Linked List

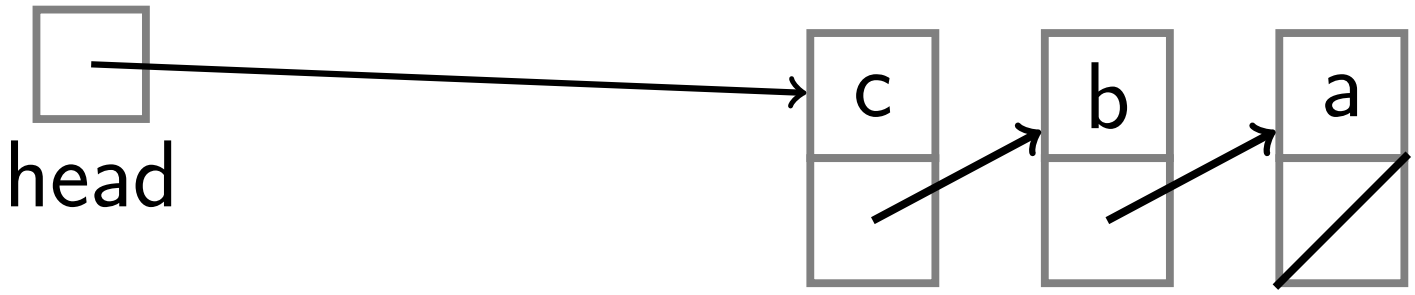


`Push(c)`

Stack Implementation with Linked List

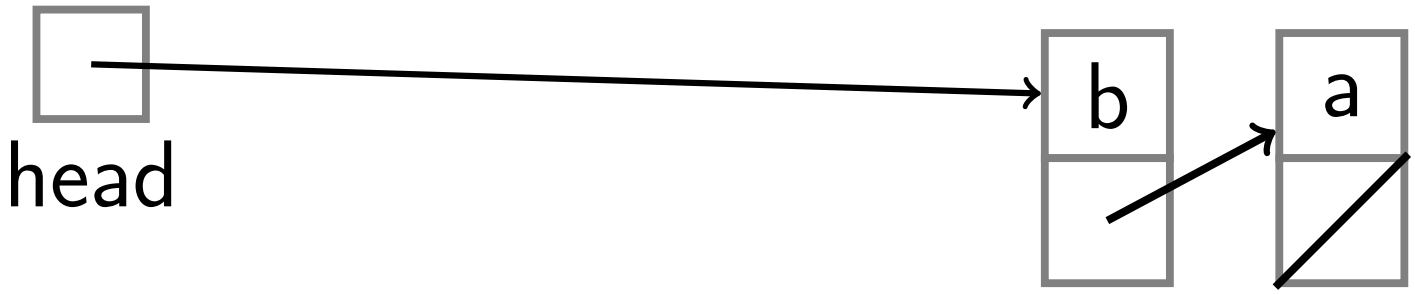


Stack Implementation with Linked List



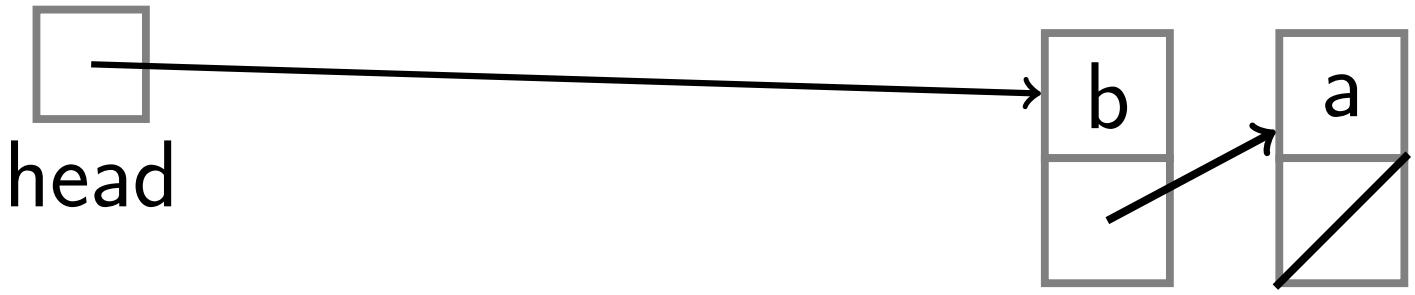
`Pop()`

Stack Implementation with Linked List

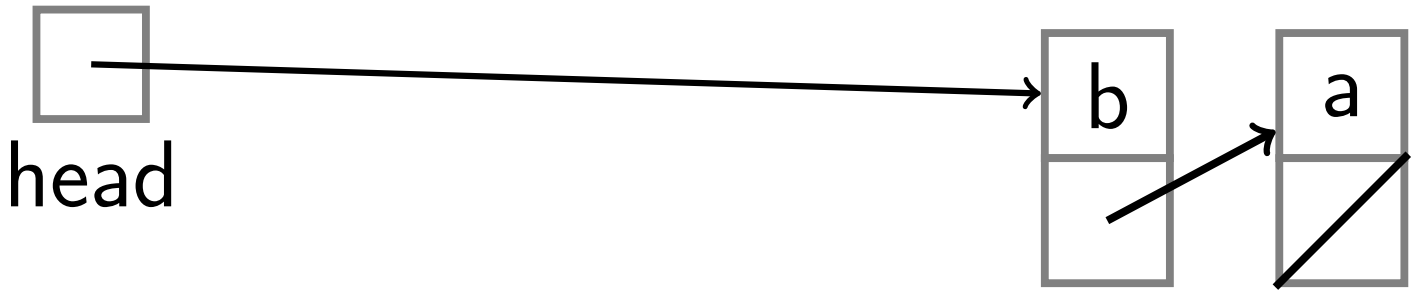


`Pop()` $\rightarrow$ `c`

Stack Implementation with Linked List

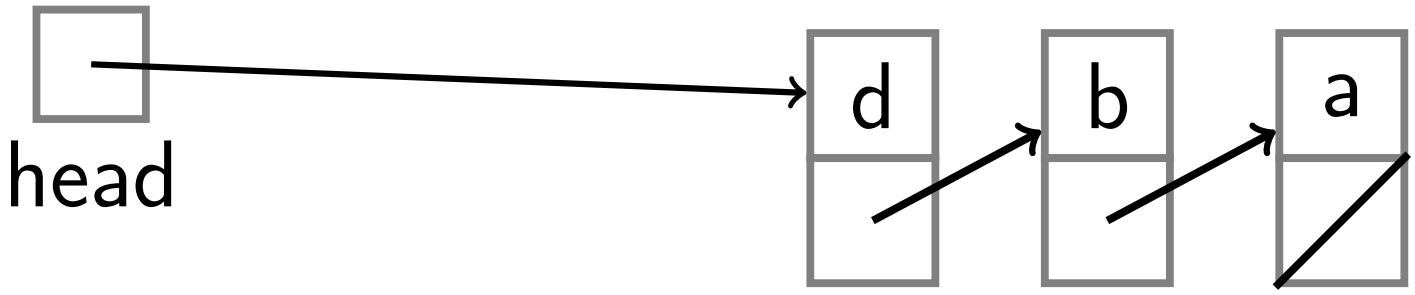


Stack Implementation with Linked List



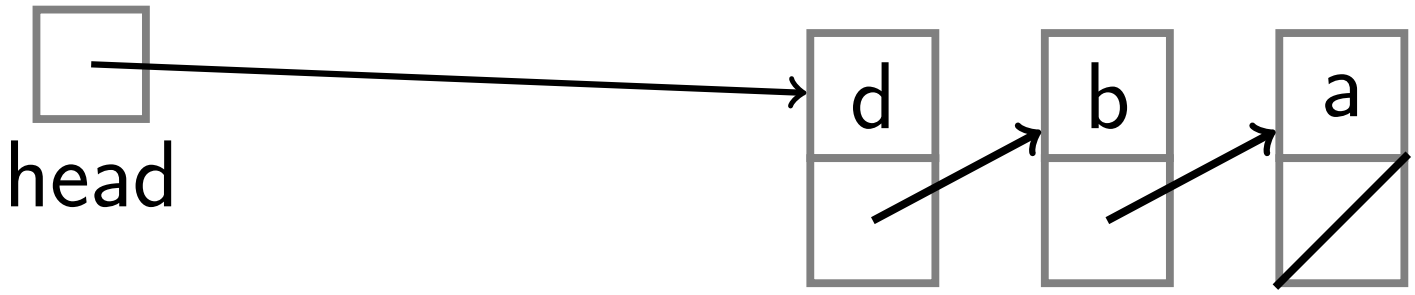
`Push(d)`

Stack Implementation with Linked List

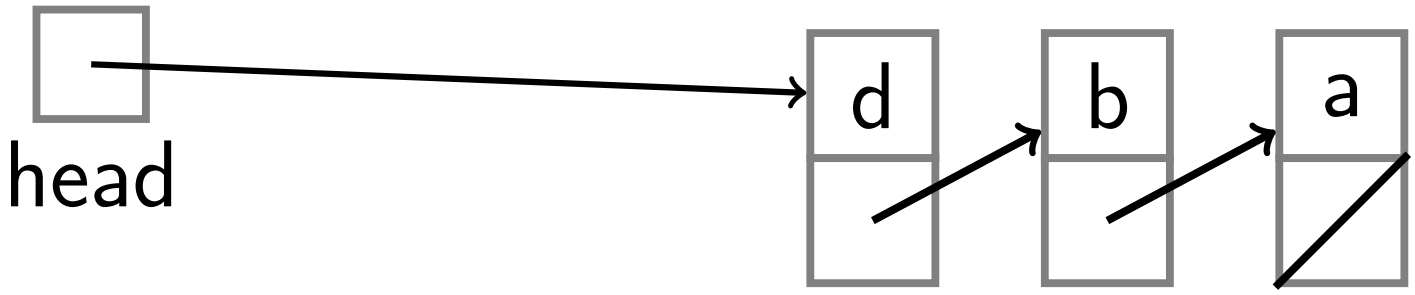


`Push(d)`

Stack Implementation with Linked List

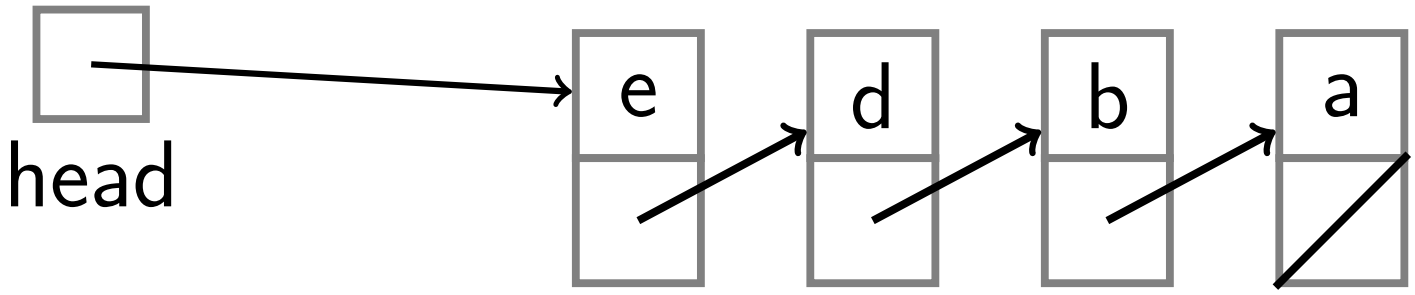


Stack Implementation with Linked List



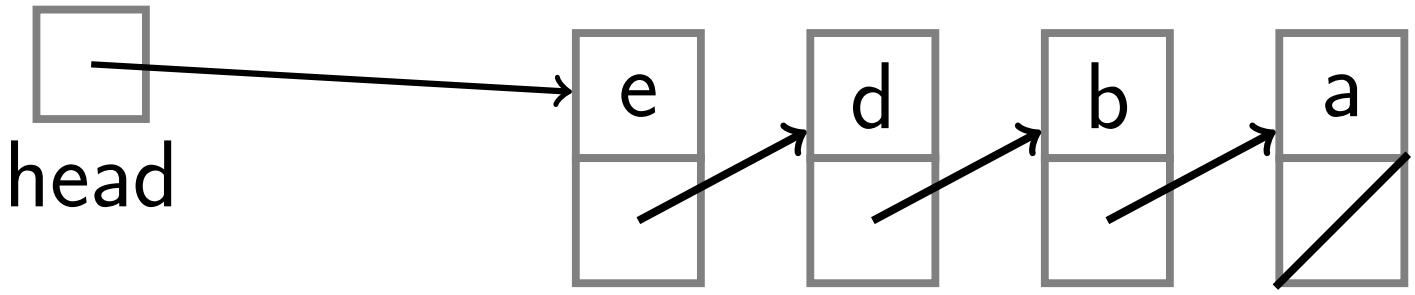
`Push(e)`

Stack Implementation with Linked List

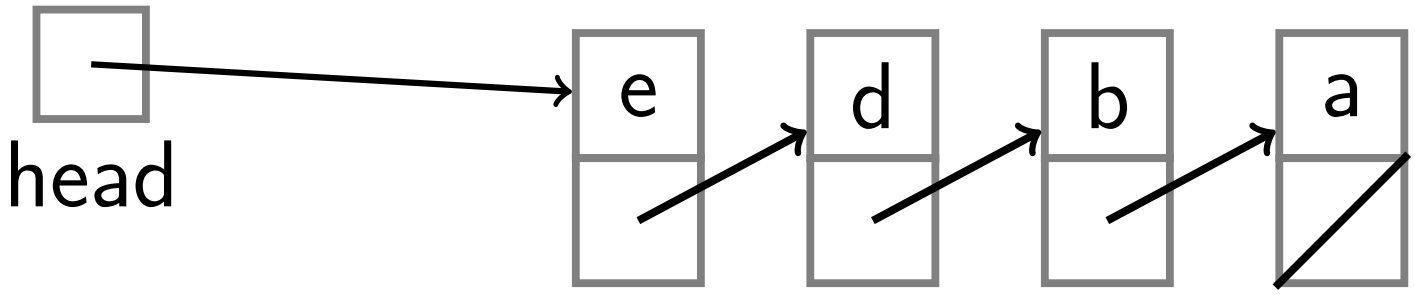


`Push(e)`

Stack Implementation with Linked List

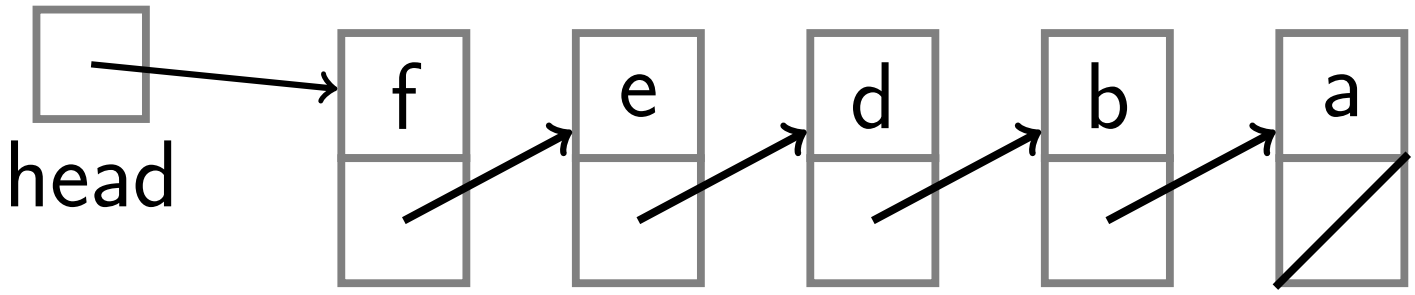


Stack Implementation with Linked List



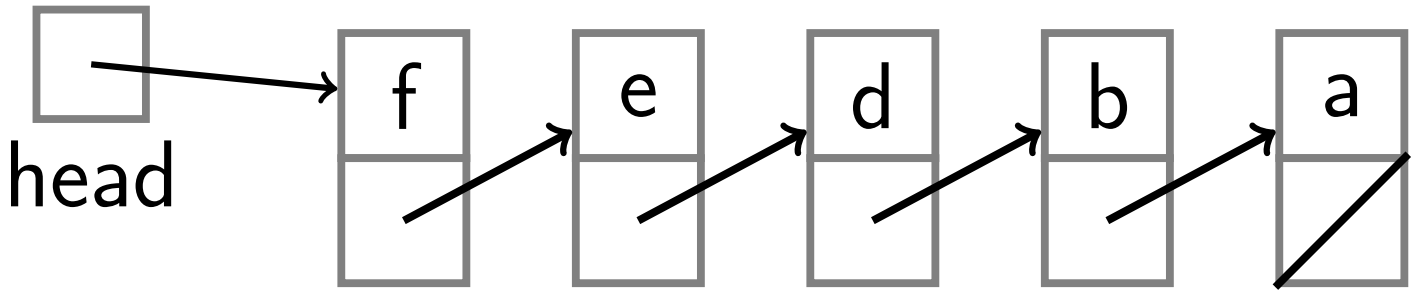
`Push(f)`

Stack Implementation with Linked List

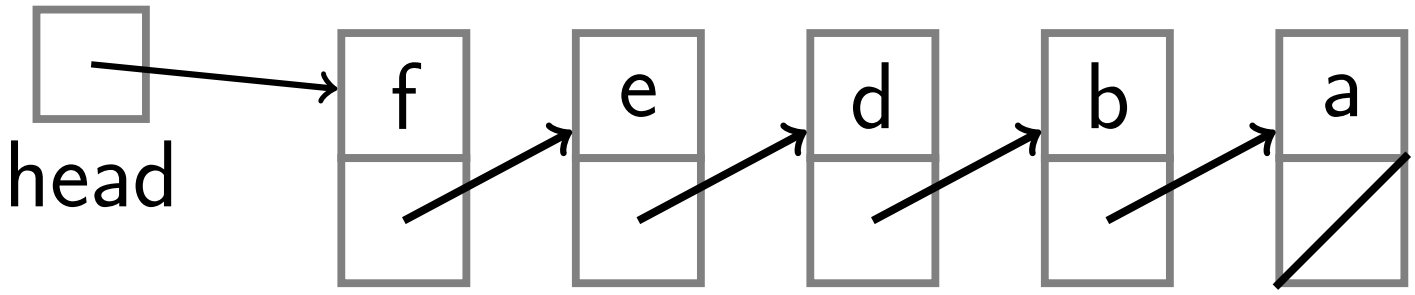


`Push(f)`

Stack Implementation with Linked List

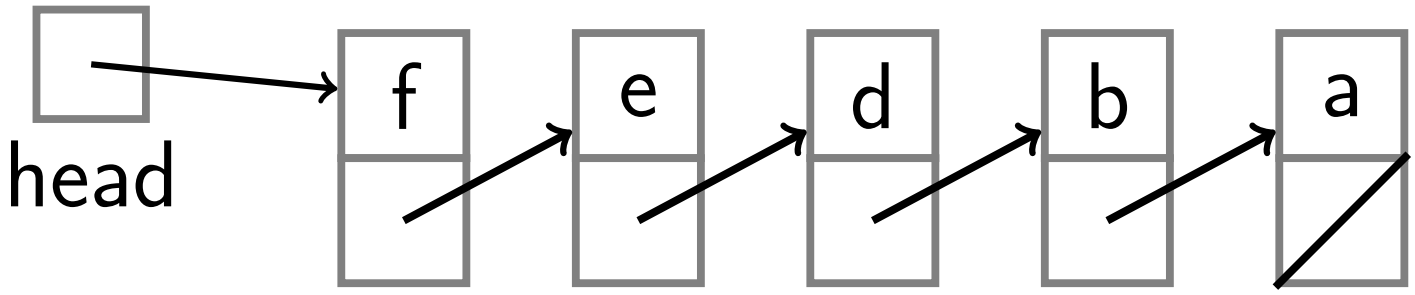


Stack Implementation with Linked List



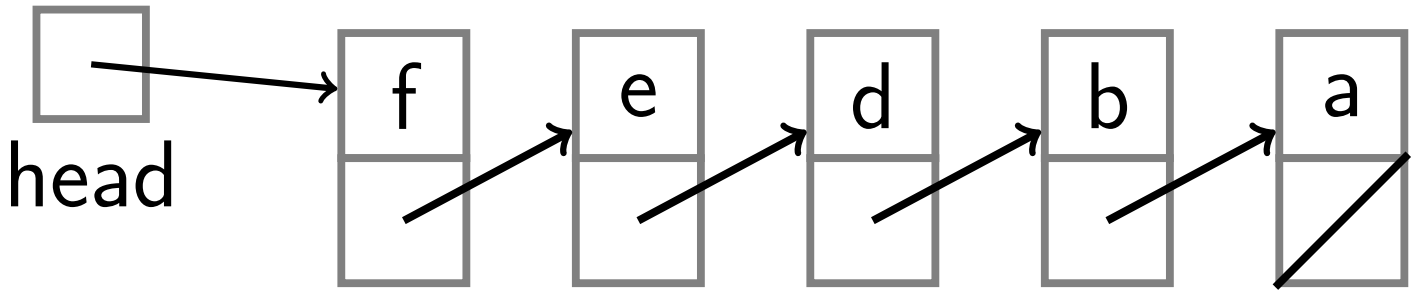

```
Empty()
```

Stack Implementation with Linked List

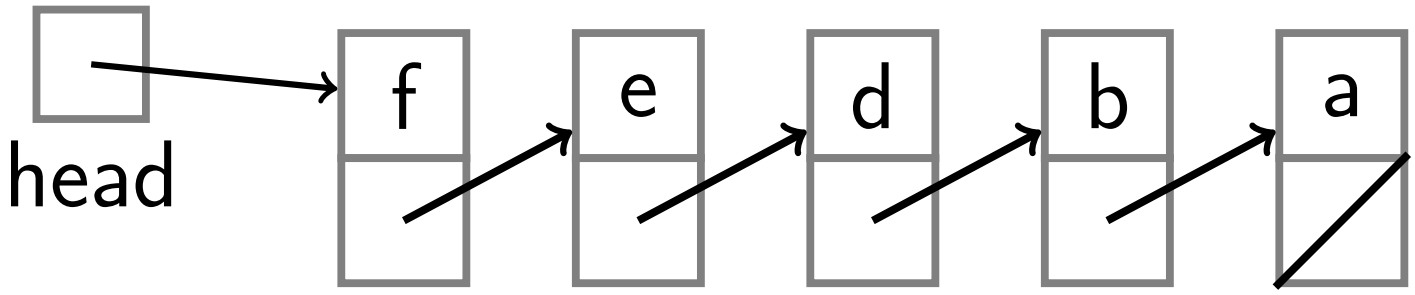


`Empty()` $\rightarrow$ `False`

Stack Implementation with Linked List

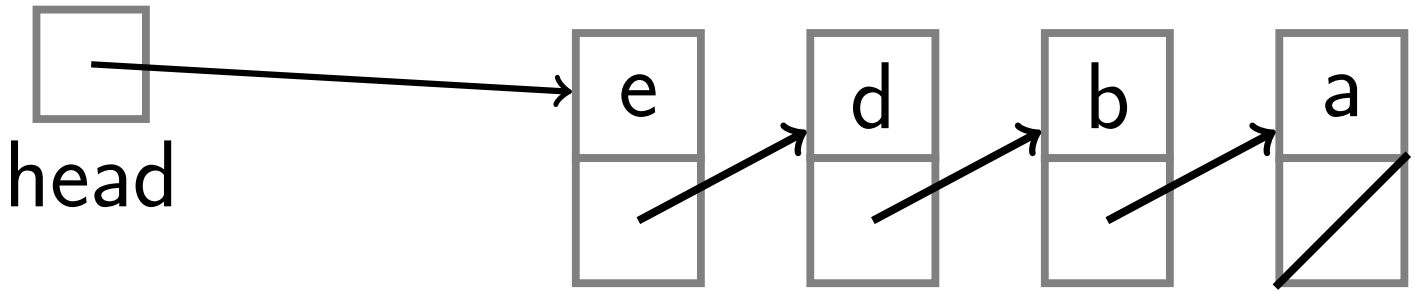


Stack Implementation with Linked List



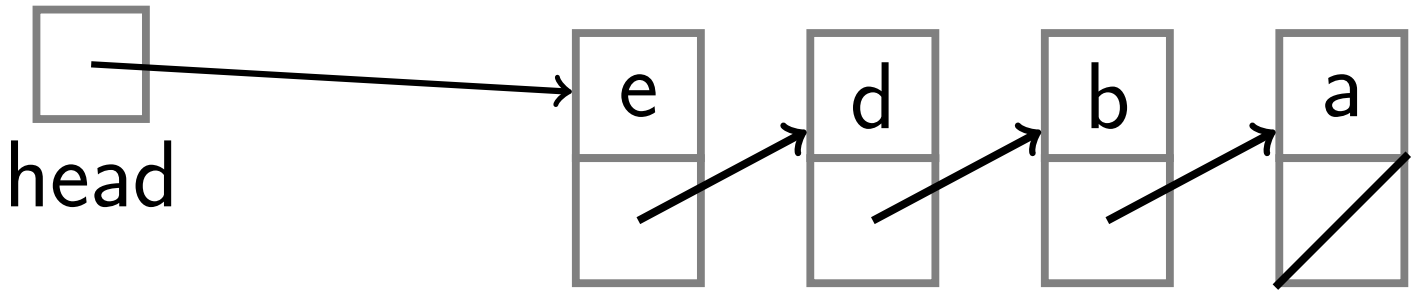
Pop()

Stack Implementation with Linked List

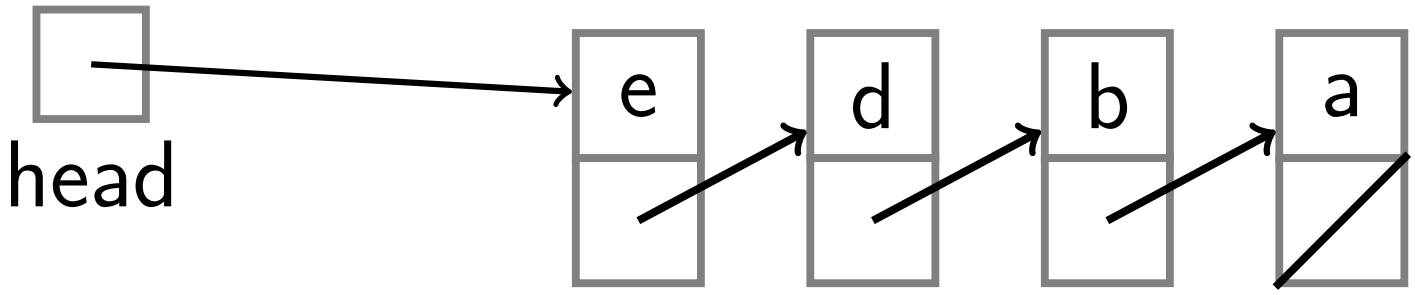


Pop()→ f

Stack Implementation with Linked List

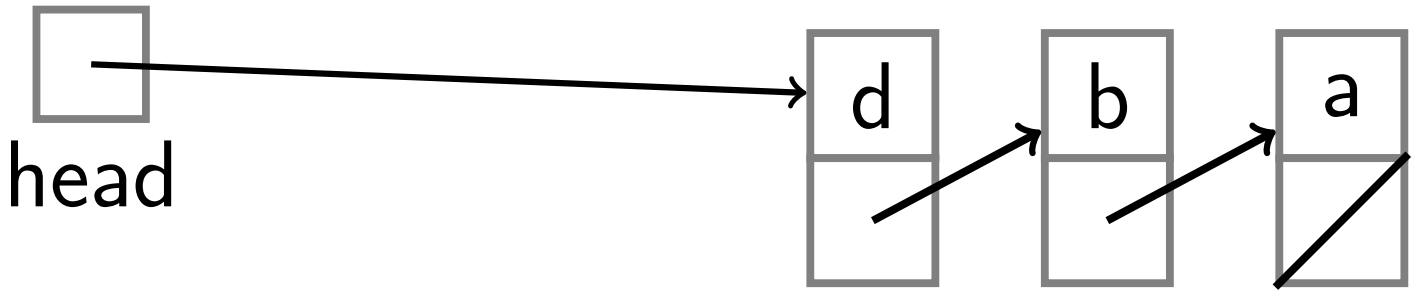


Stack Implementation with Linked List



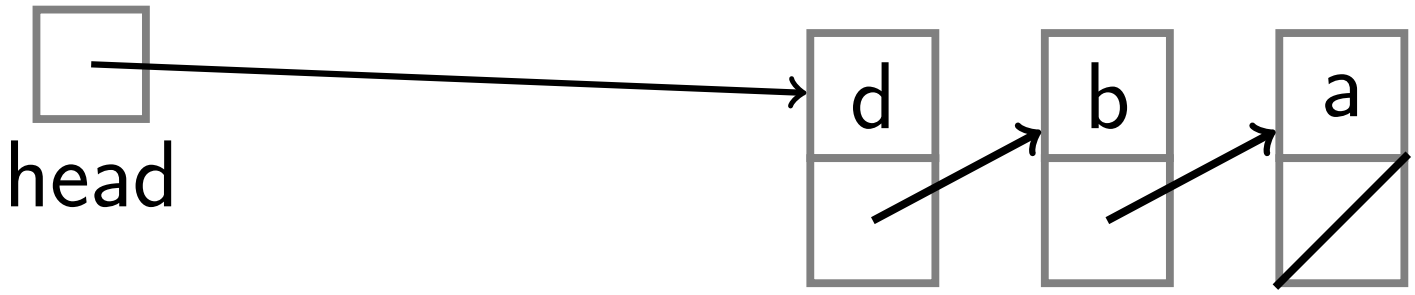
`Pop()`

Stack Implementation with Linked List

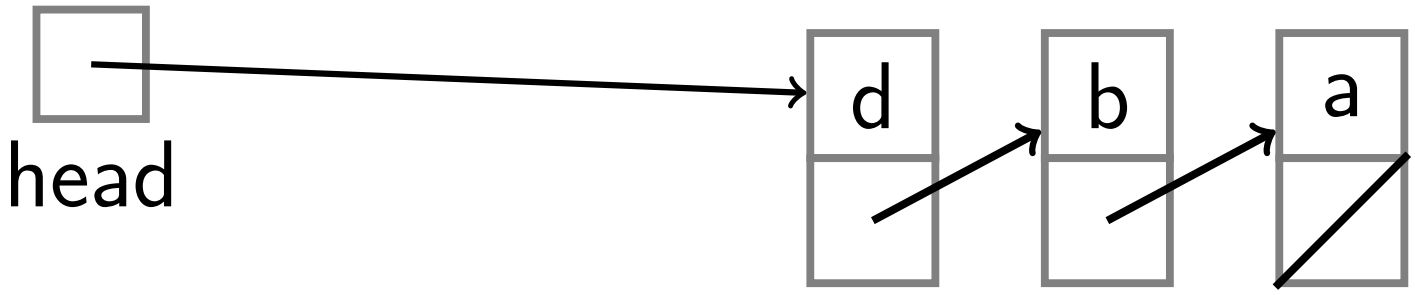


`Pop()` $\rightarrow$ `e`

Stack Implementation with Linked List

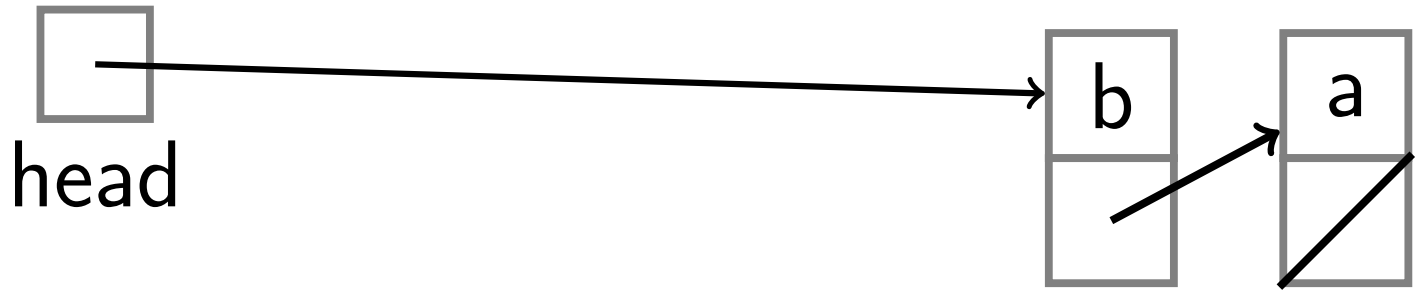


Stack Implementation with Linked List



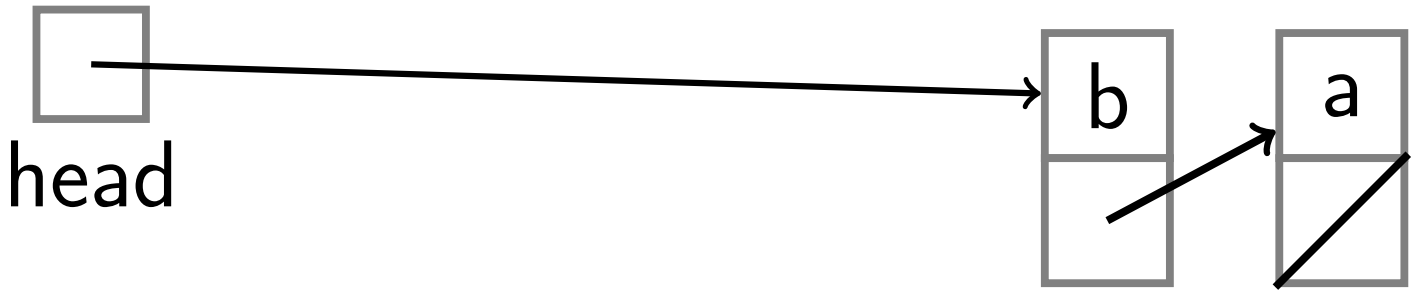
`Pop()`

Stack Implementation with Linked List

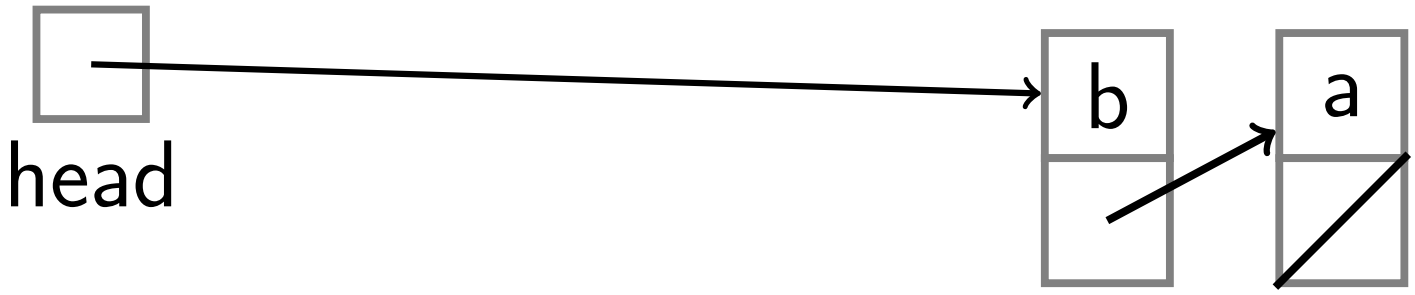


`Pop()` $\rightarrow$ `d`

Stack Implementation with Linked List

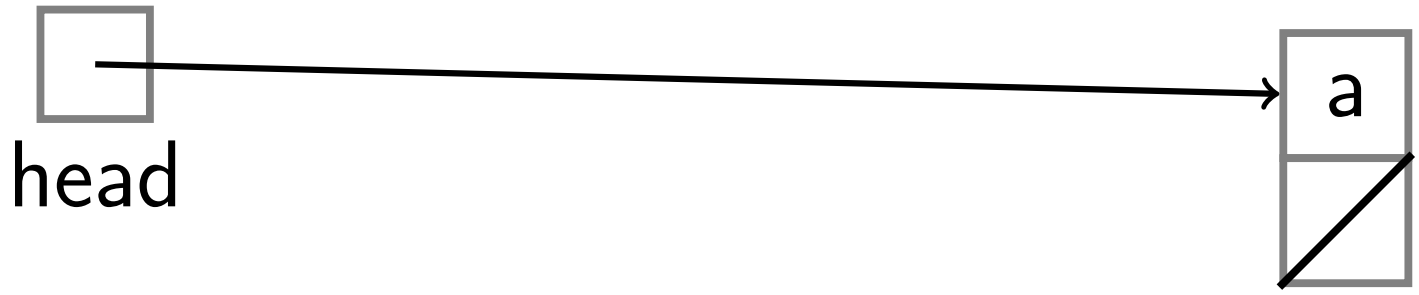


Stack Implementation with Linked List



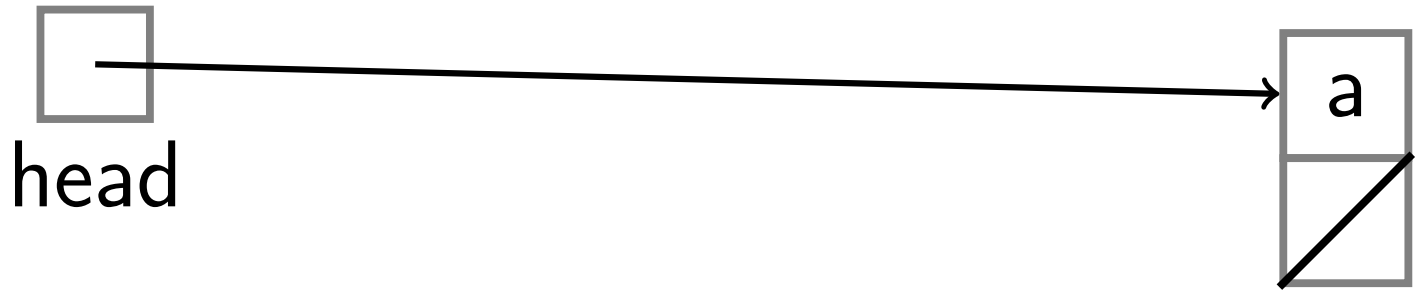
`Pop()`

Stack Implementation with Linked List

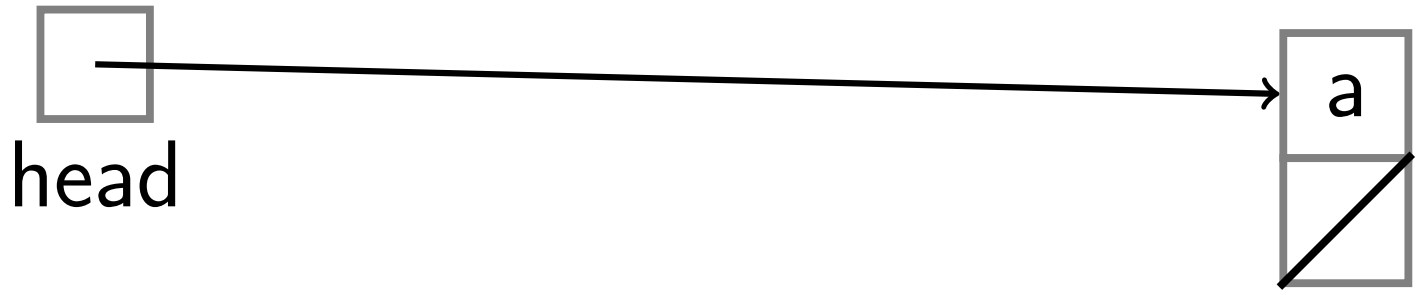


`Pop()` $\rightarrow$ `b`

Stack Implementation with Linked List

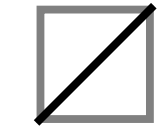


Stack Implementation with Linked List



`Pop()`

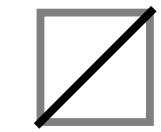
Stack Implementation with Linked List



head

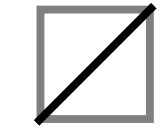
`Pop()` $\rightarrow$ `a`

Stack Implementation with Linked List



head

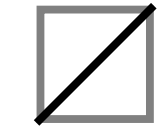
Stack Implementation with Linked List



head

`Empty()`

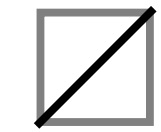
Stack Implementation with Linked List



head

`Empty()` $\rightarrow$ `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 ocassionaly known as LIFO queues.

Outline

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?

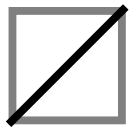
Definition

Queue: Abstract data type with the following operations:

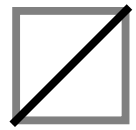
- `Enqueue(Key)`: adds key to collection
- `Key Dequeue()`: removes and returns least recently-added key
- `Boolean Empty()`: are there any elements?

FIFO: First-In, First-Out

Queue Implementation with Linked List

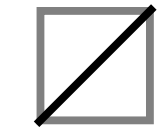


head

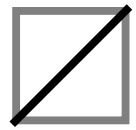


tail

Queue Implementation with Linked List



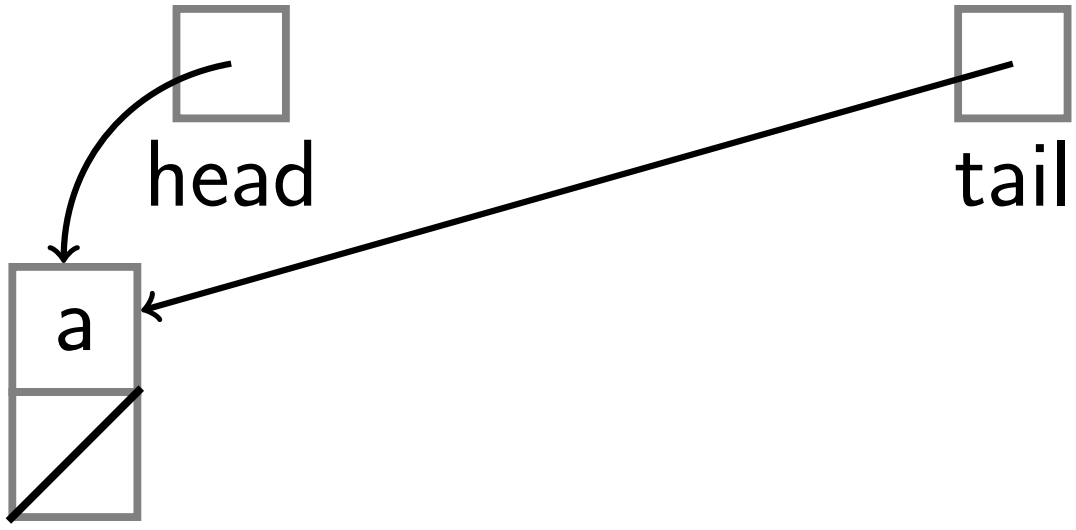
head



tail

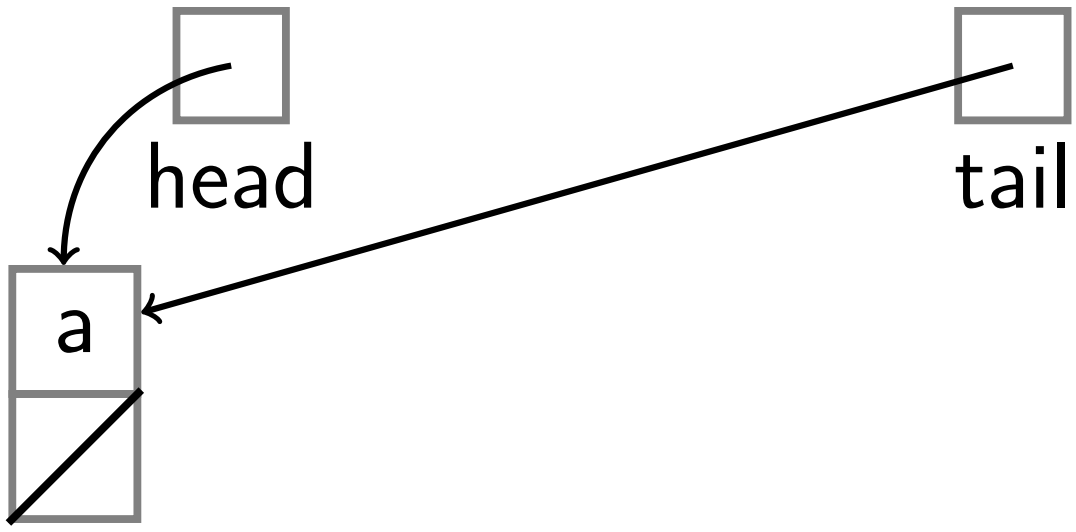
`Enqueue(a)`

Queue Implementation with Linked List

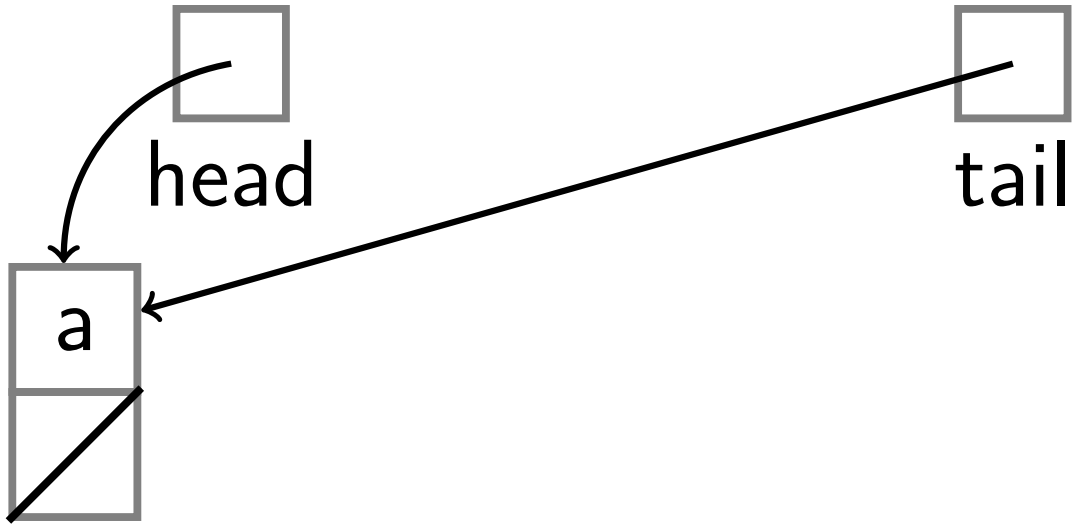


`Enqueue(a)`

Queue Implementation with Linked List

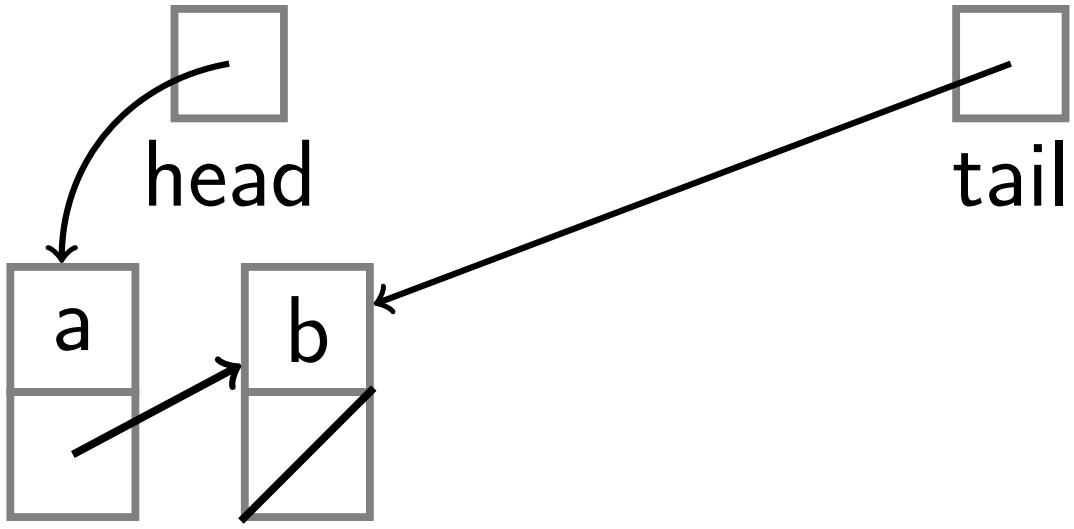


Queue Implementation with Linked List



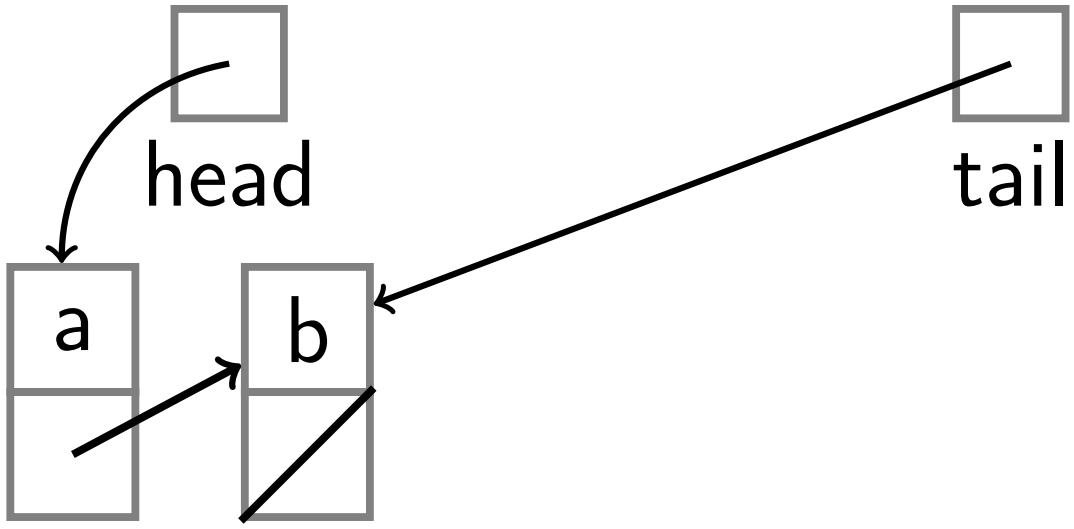
`Enqueue(b)`

Queue Implementation with Linked List

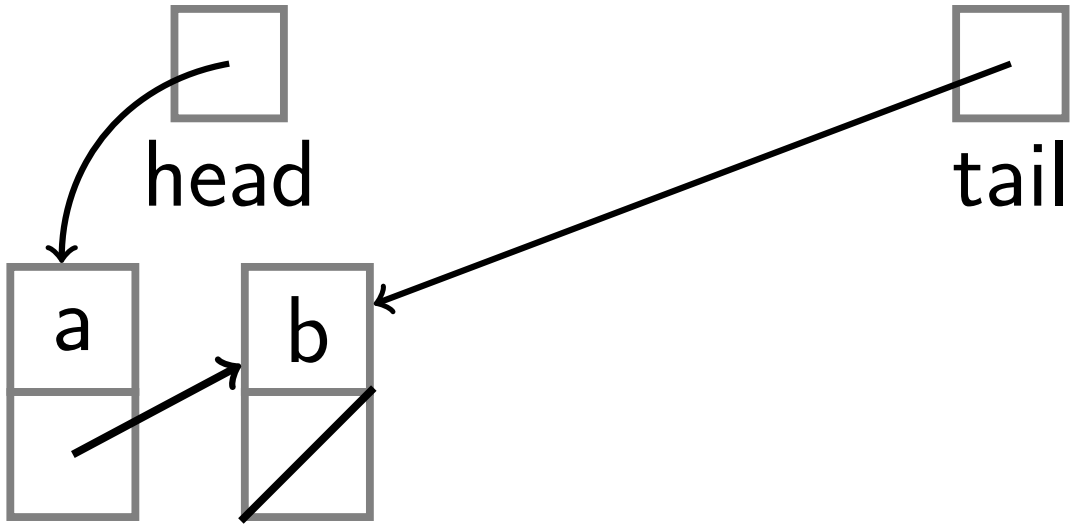


`Enqueue(b)`

Queue Implementation with Linked List

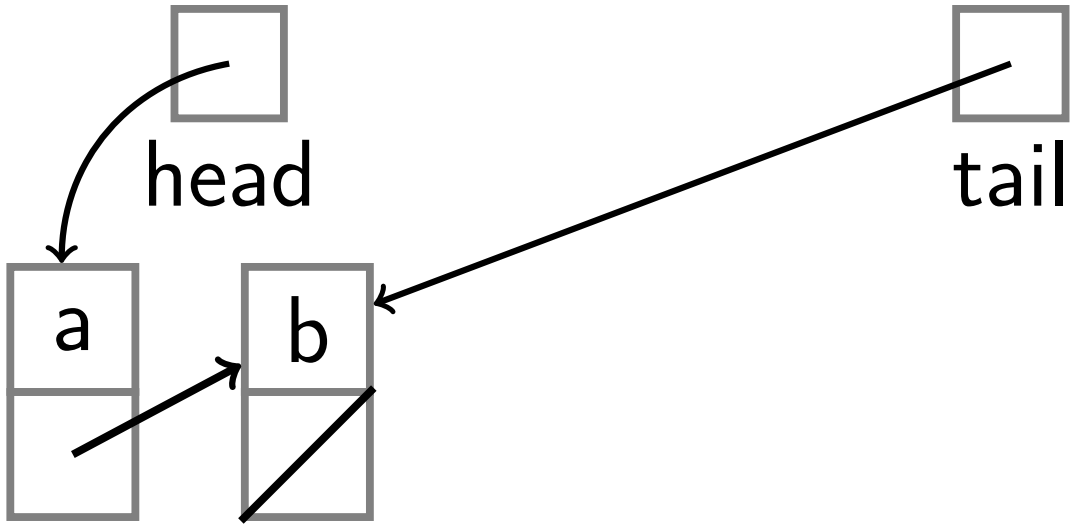


Queue Implementation with Linked List



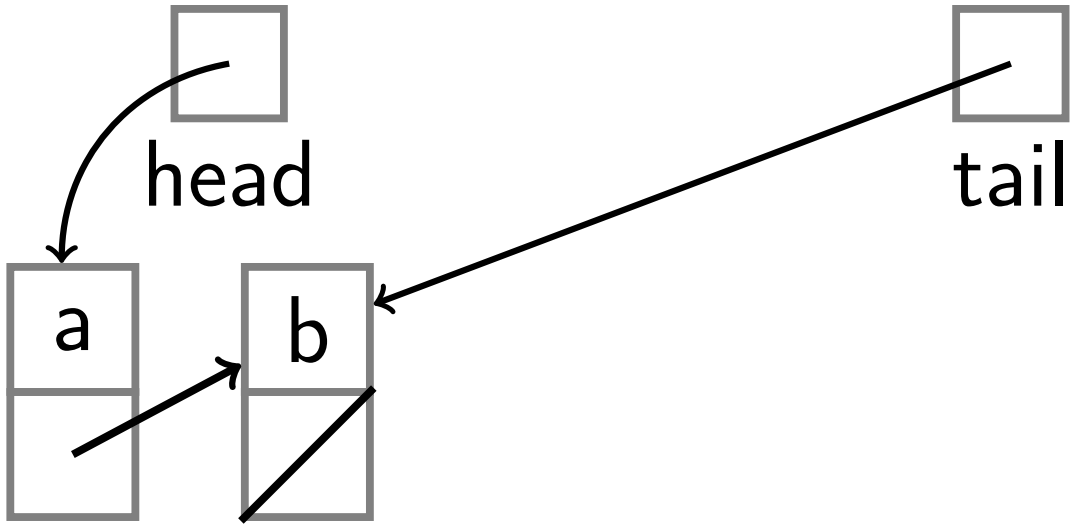
`Empty()`

Queue Implementation with Linked List

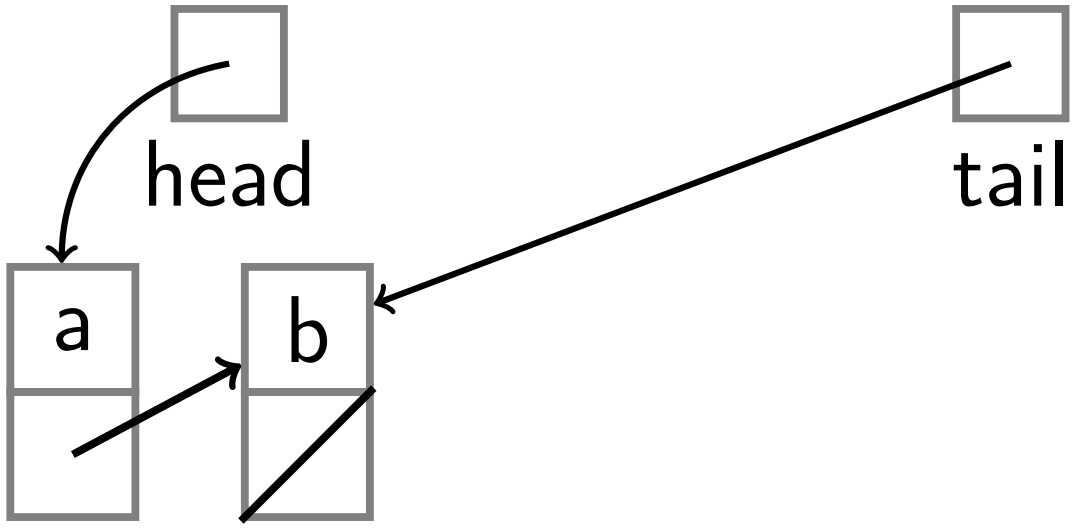


`Empty()` $\rightarrow$ `False`

Queue Implementation with Linked List

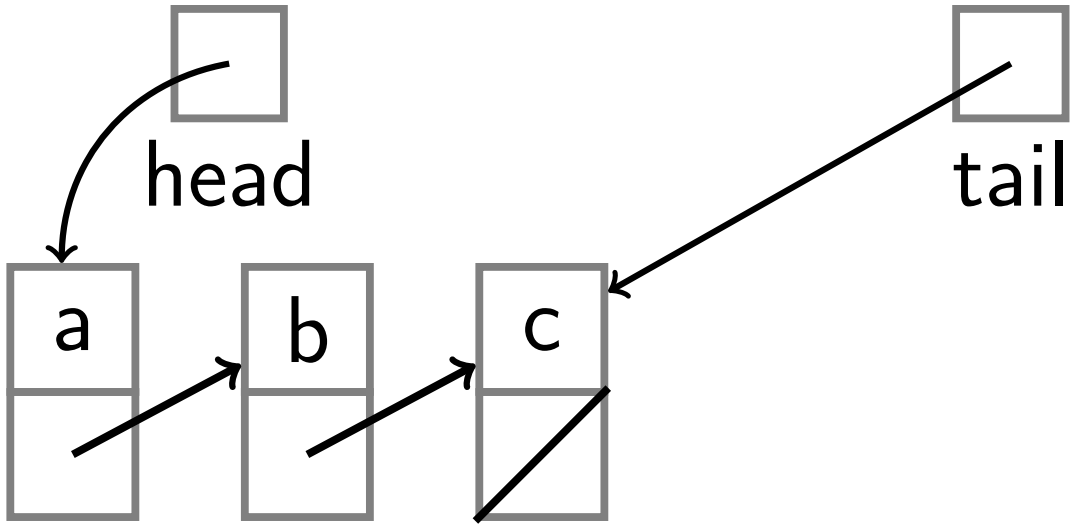


Queue Implementation with Linked List



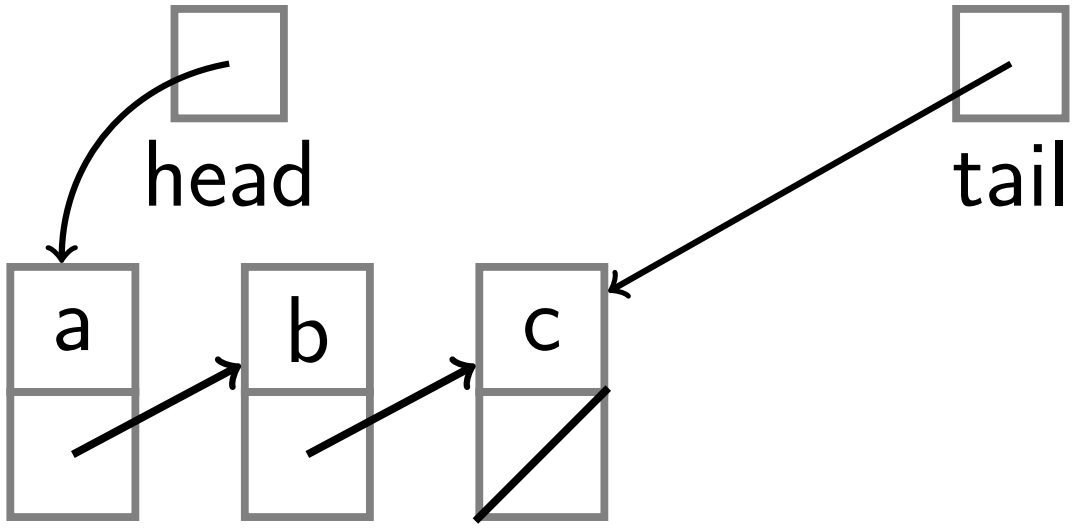
`Enqueue(c)`

Queue Implementation with Linked List

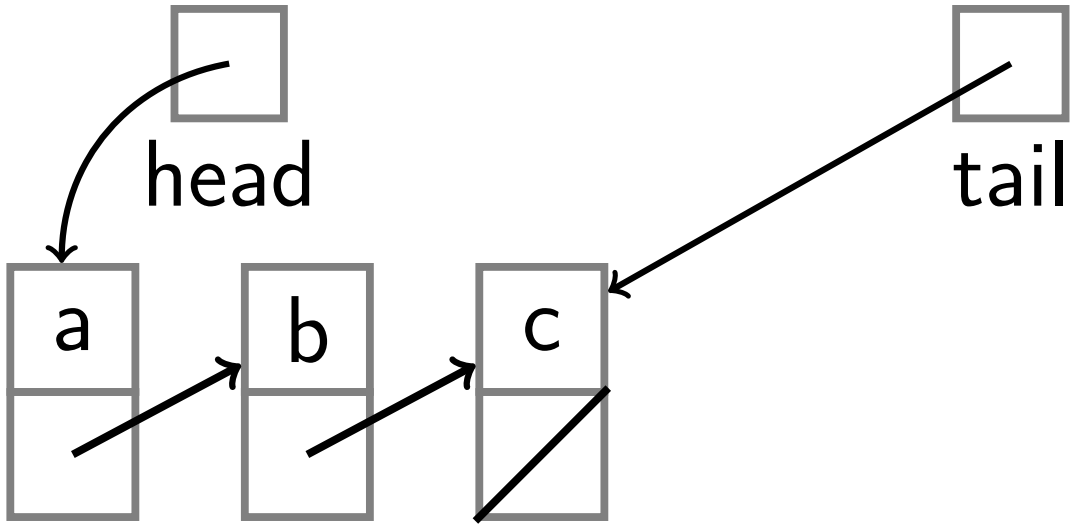


`Enqueue(c)`

Queue Implementation with Linked List

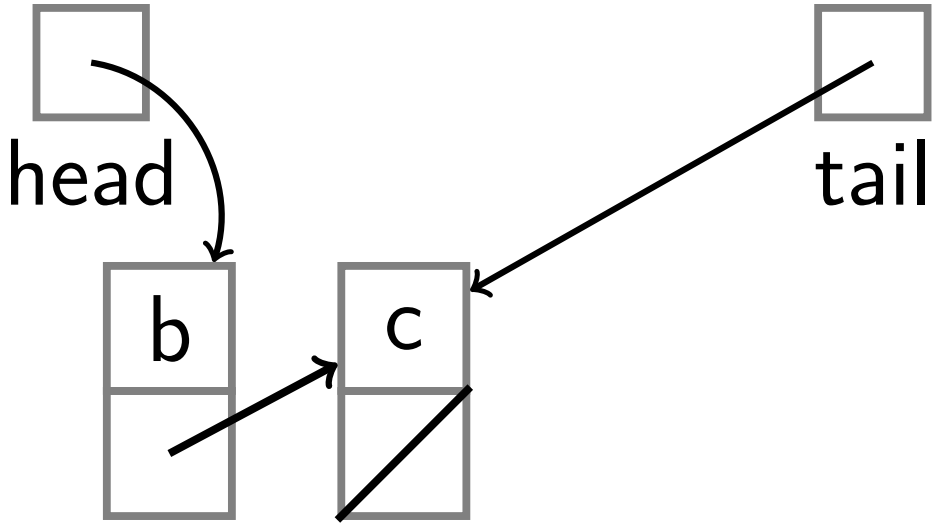


Queue Implementation with Linked List



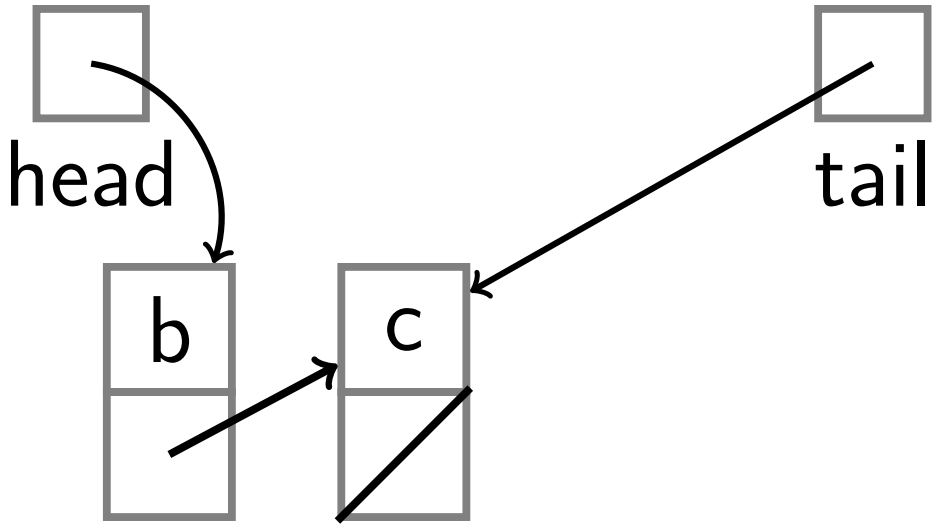
`Dequeue()`

Queue Implementation with Linked List

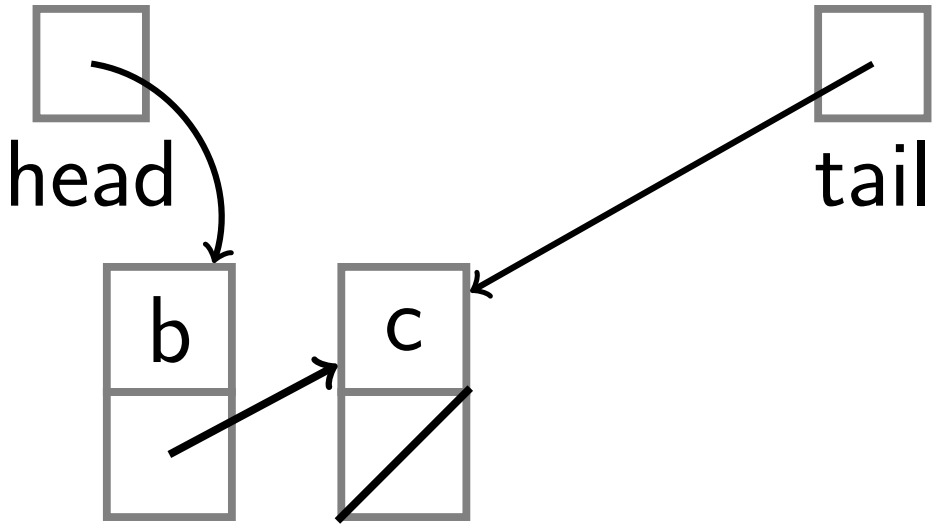


`Dequeue()` $\rightarrow$ `a`

Queue Implementation with Linked List

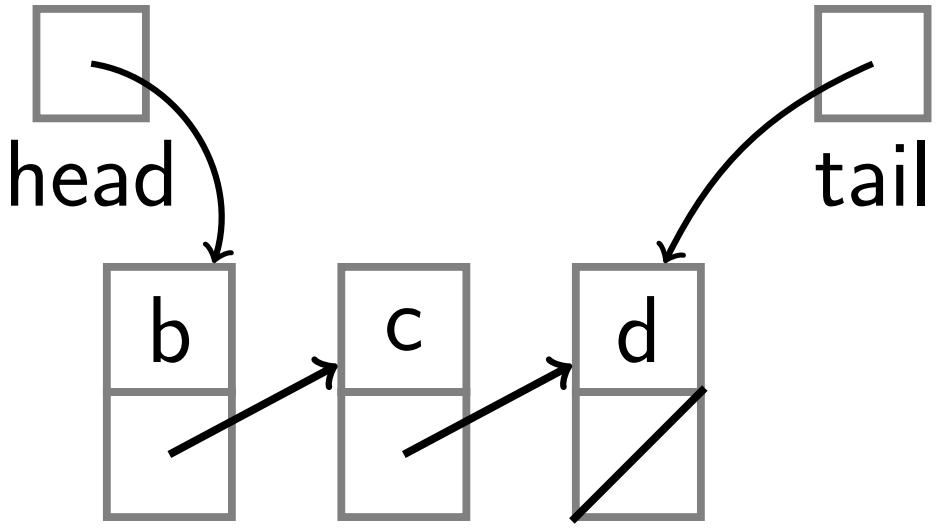


Queue Implementation with Linked List



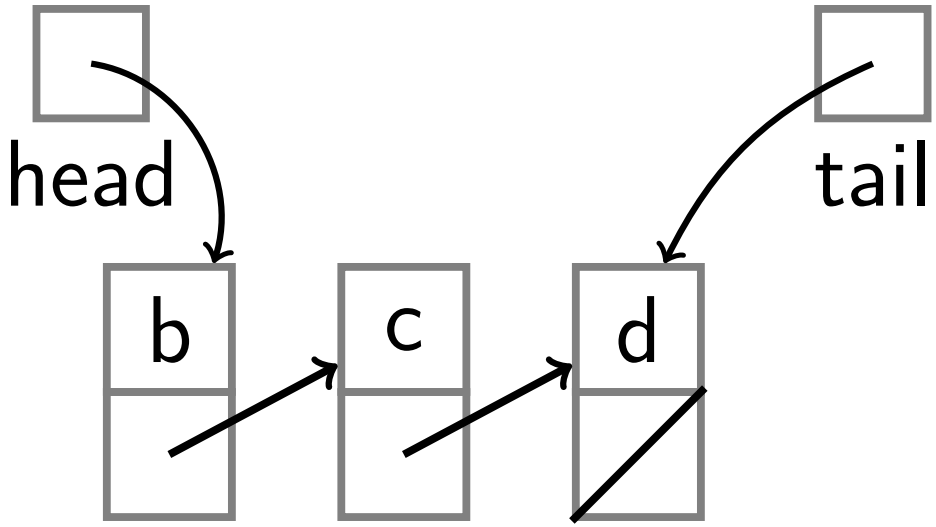
`Enqueue(d)`

Queue Implementation with Linked List

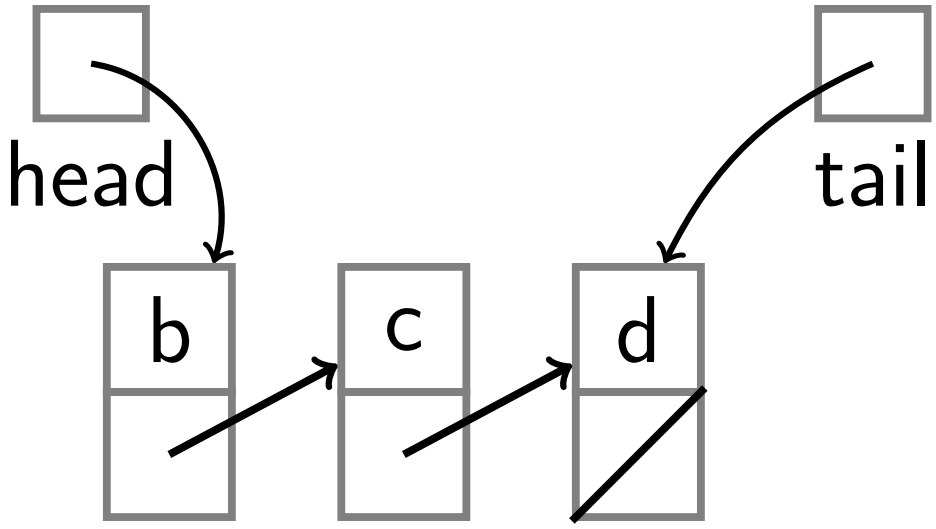


`Enqueue(d)`

Queue Implementation with Linked List

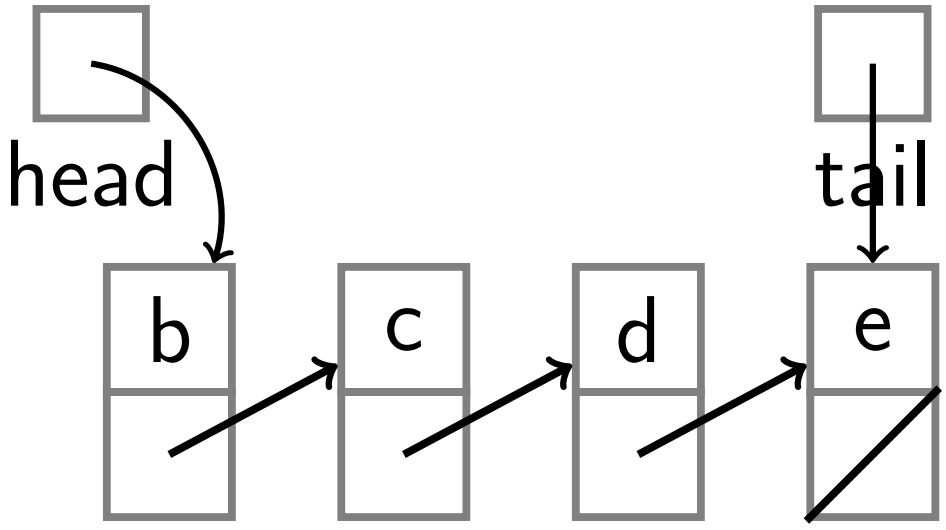


Queue Implementation with Linked List



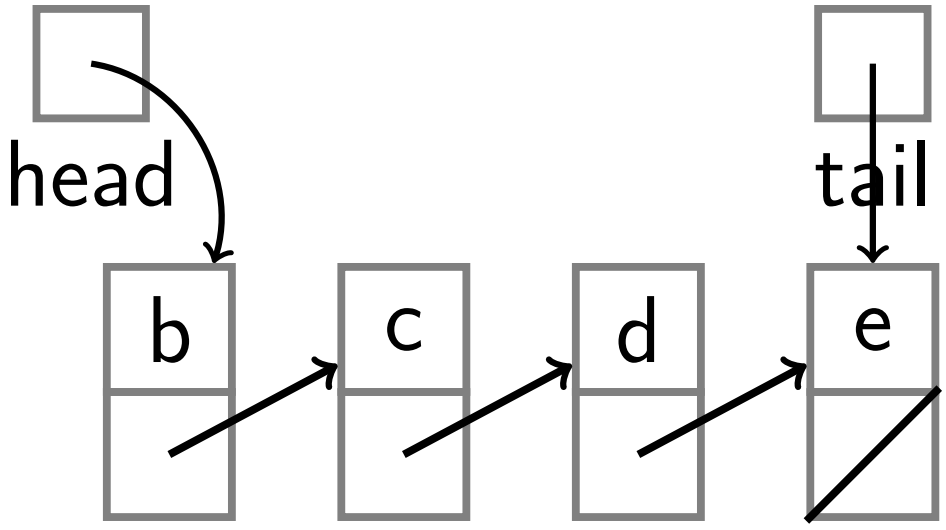
Enqueue(e)

Queue Implementation with Linked List

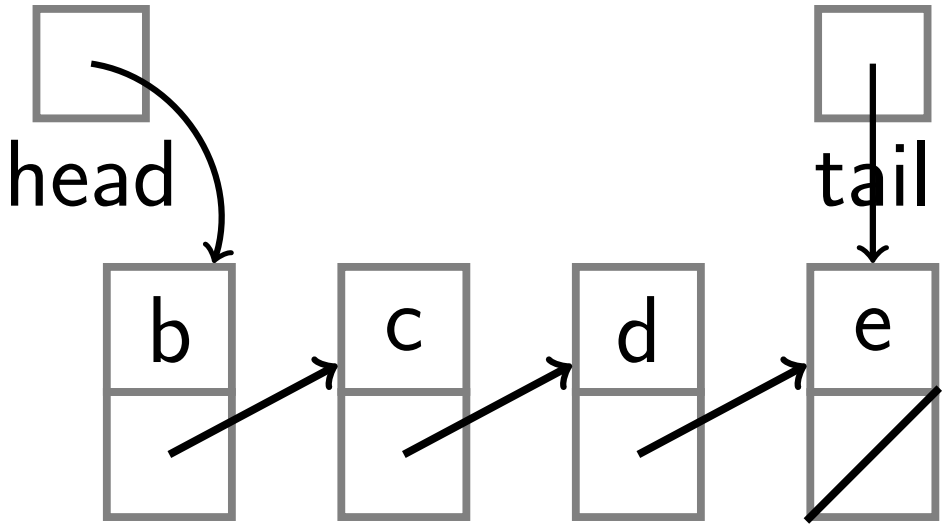


Enqueue(e)

Queue Implementation with Linked List

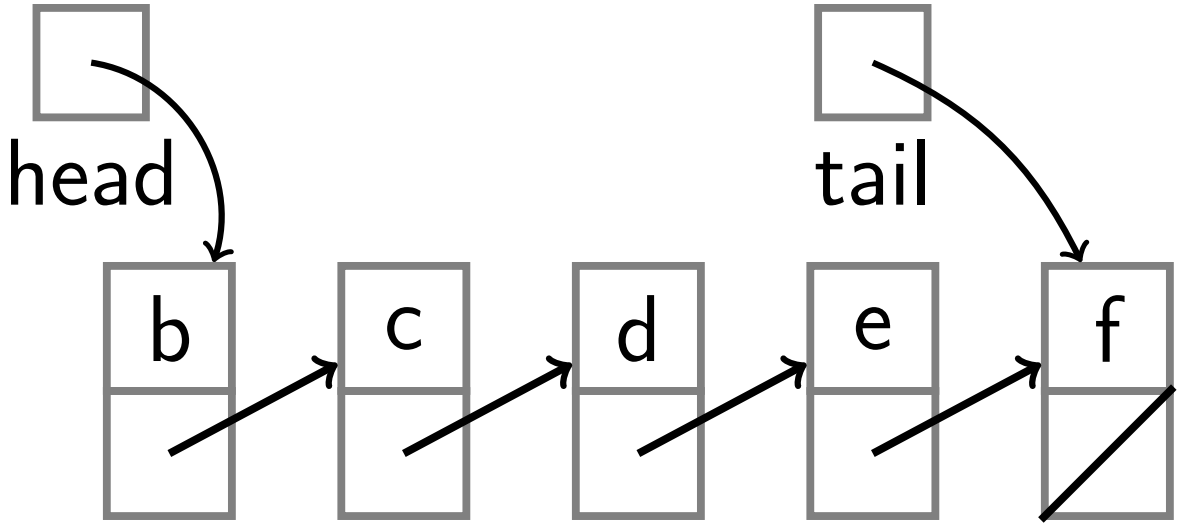


Queue Implementation with Linked List



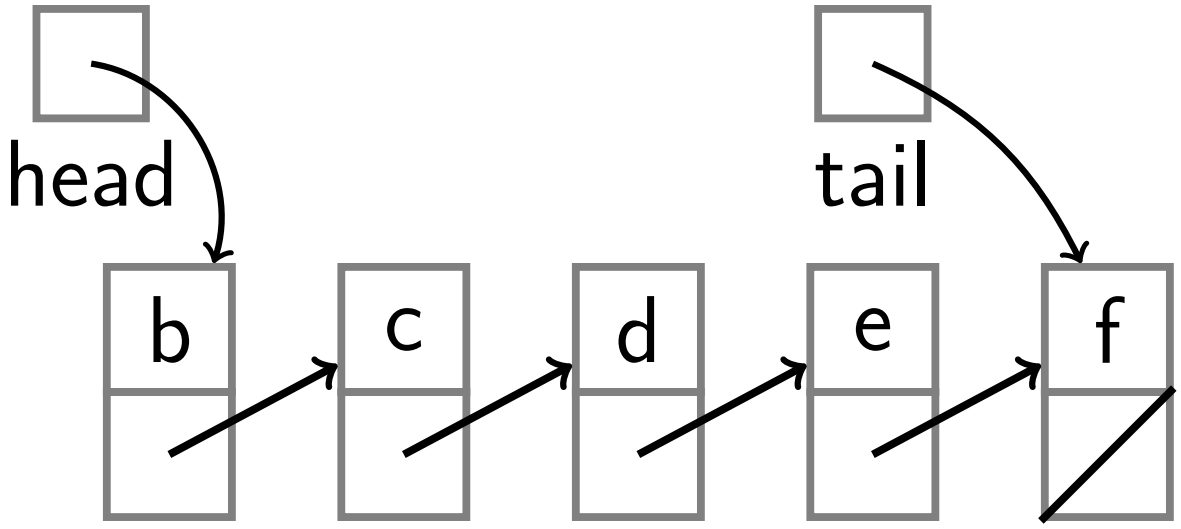
`Enqueue(f)`

Queue Implementation with Linked List

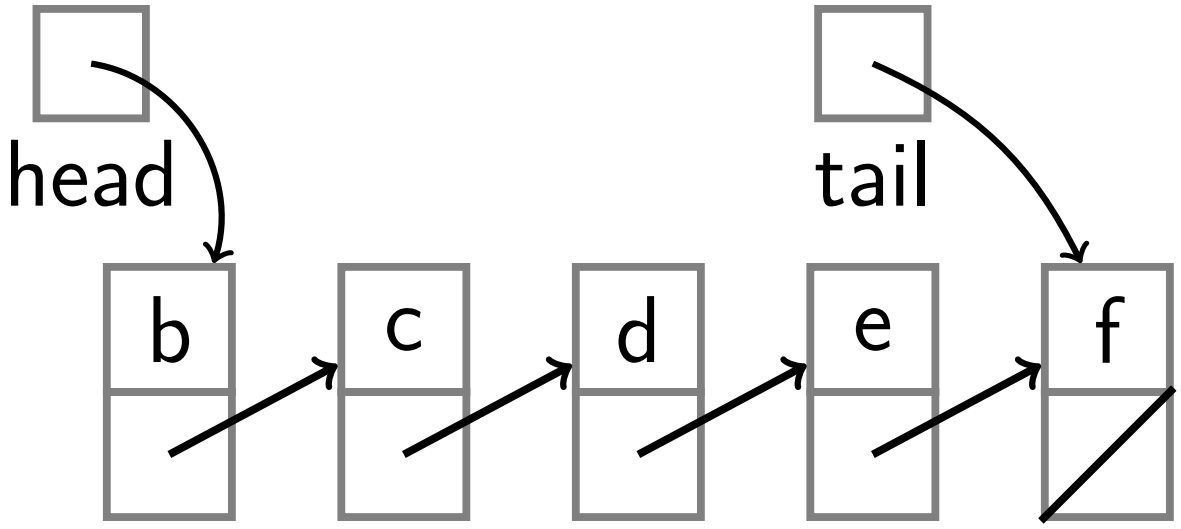


`Enqueue(f)`

Queue Implementation with Linked List

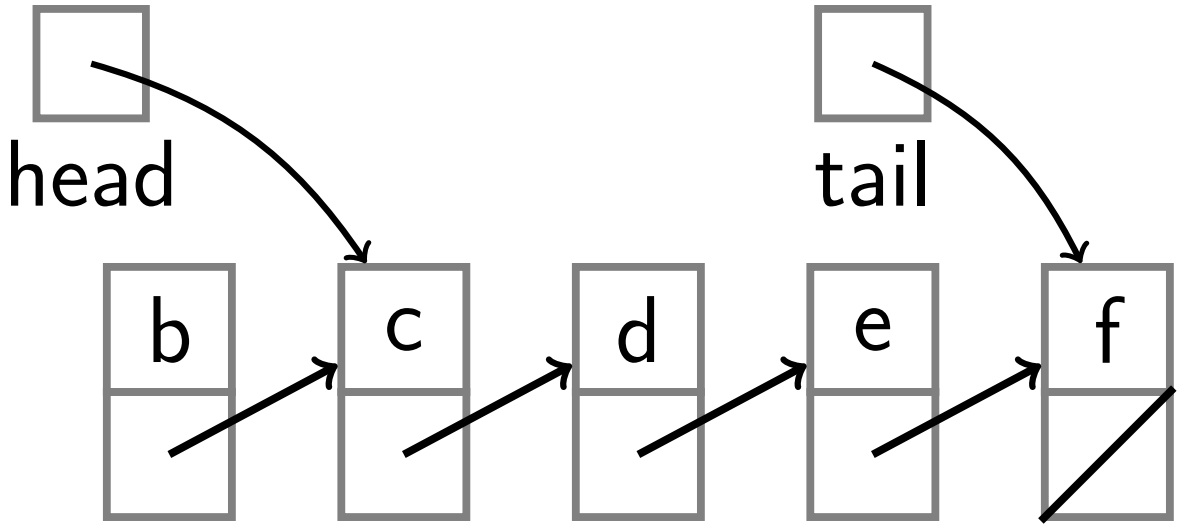


Queue Implementation with Linked List



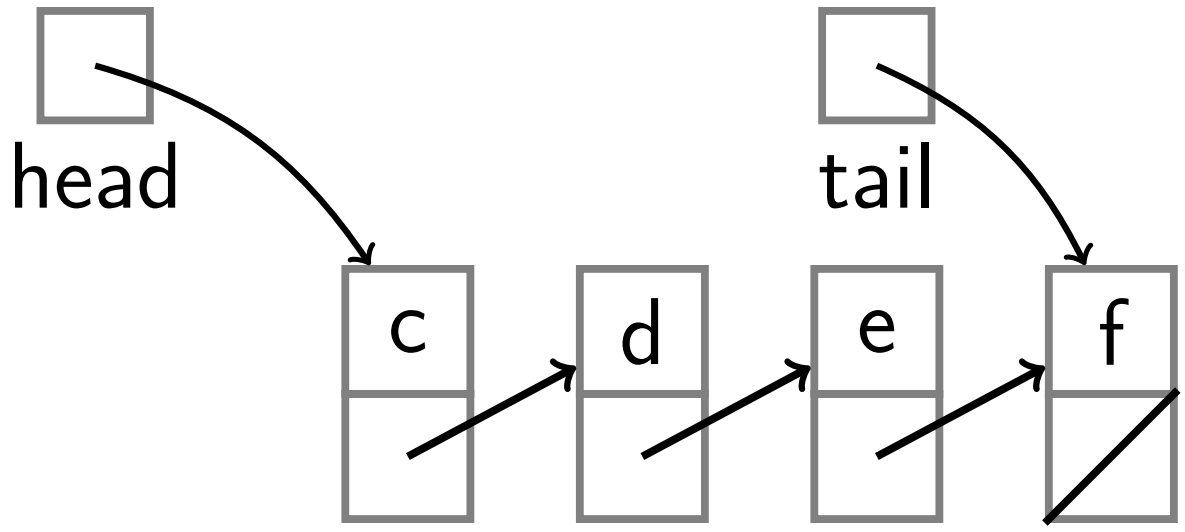
Dequeue()

Queue Implementation with Linked List

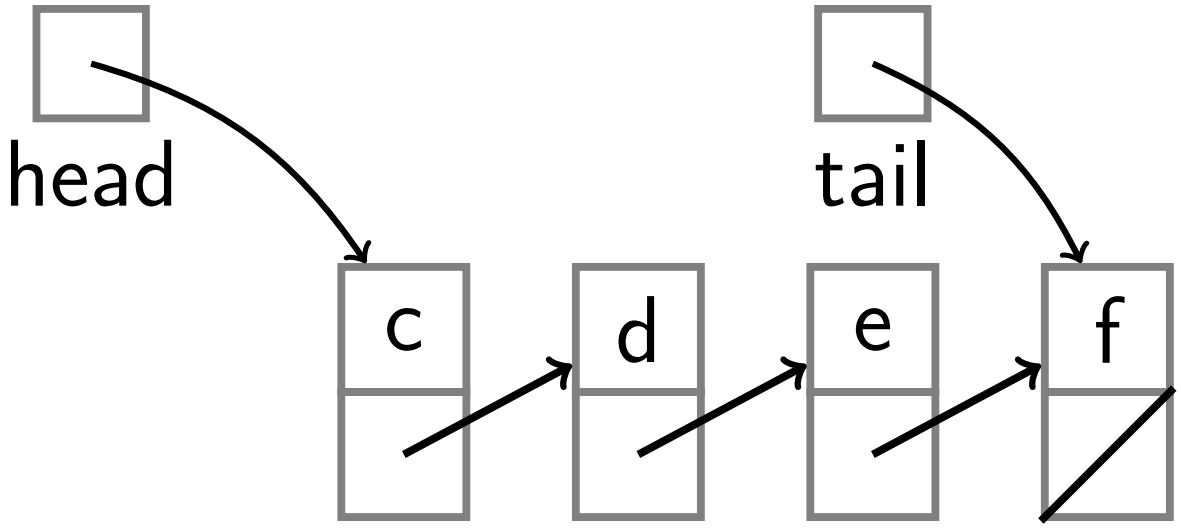


Dequeue()$\rightarrow$ b

Queue Implementation with Linked List

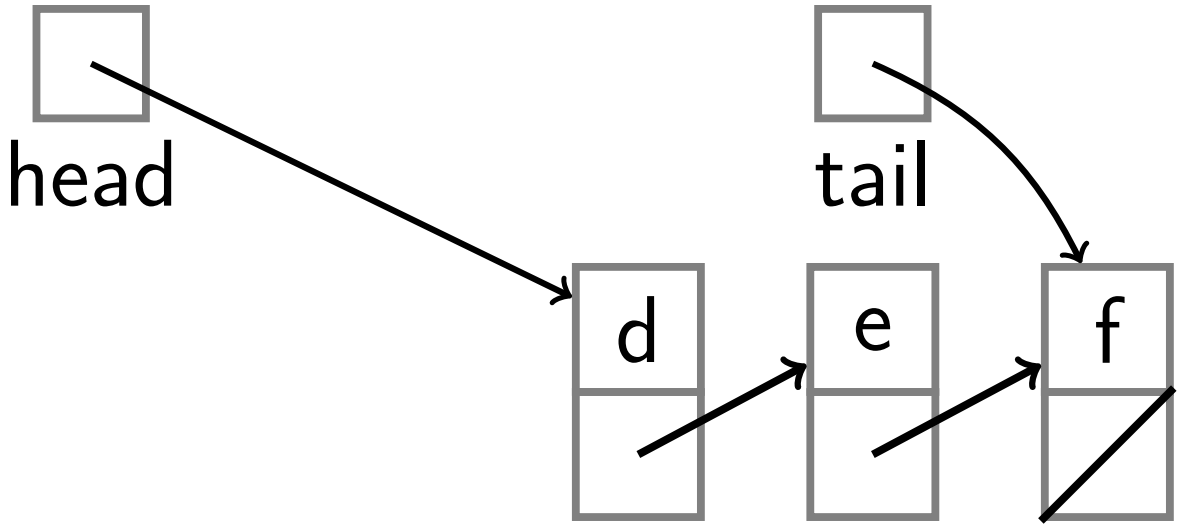


Queue Implementation with Linked List



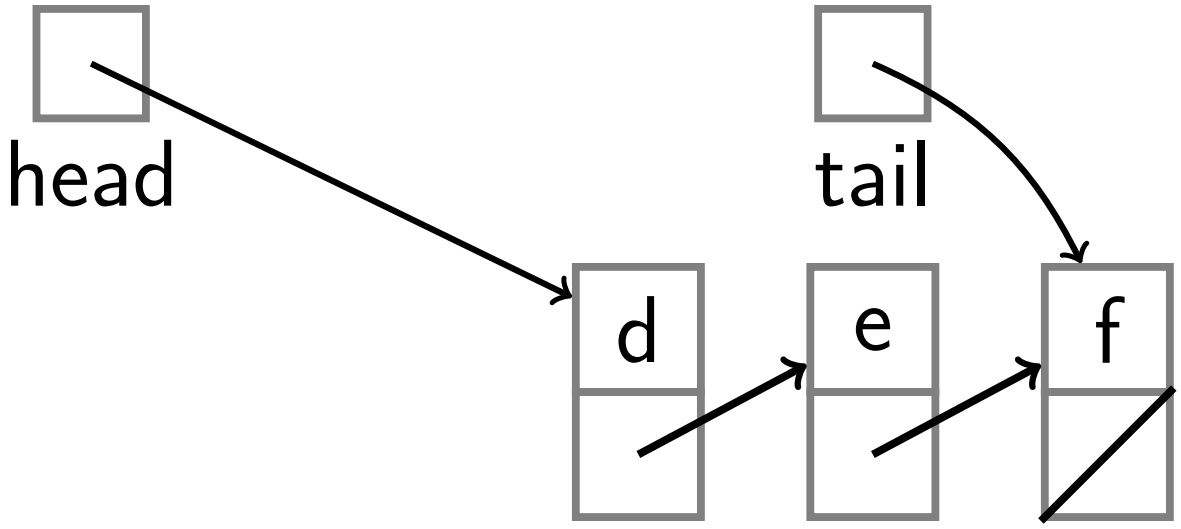
`Dequeue()`

Queue Implementation with Linked List

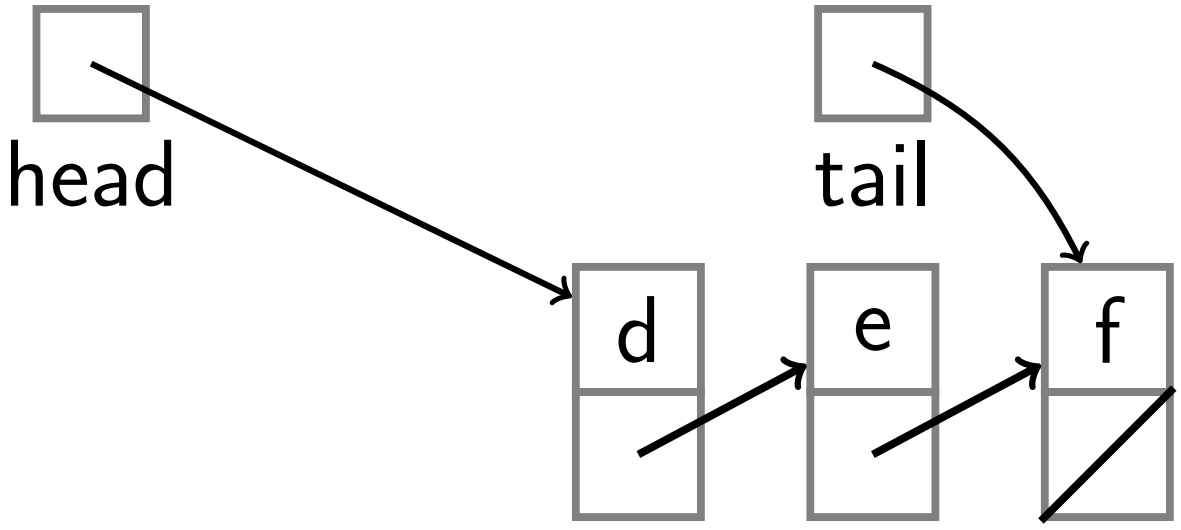


`Dequeue()` $\rightarrow$ `c`

Queue Implementation with Linked List

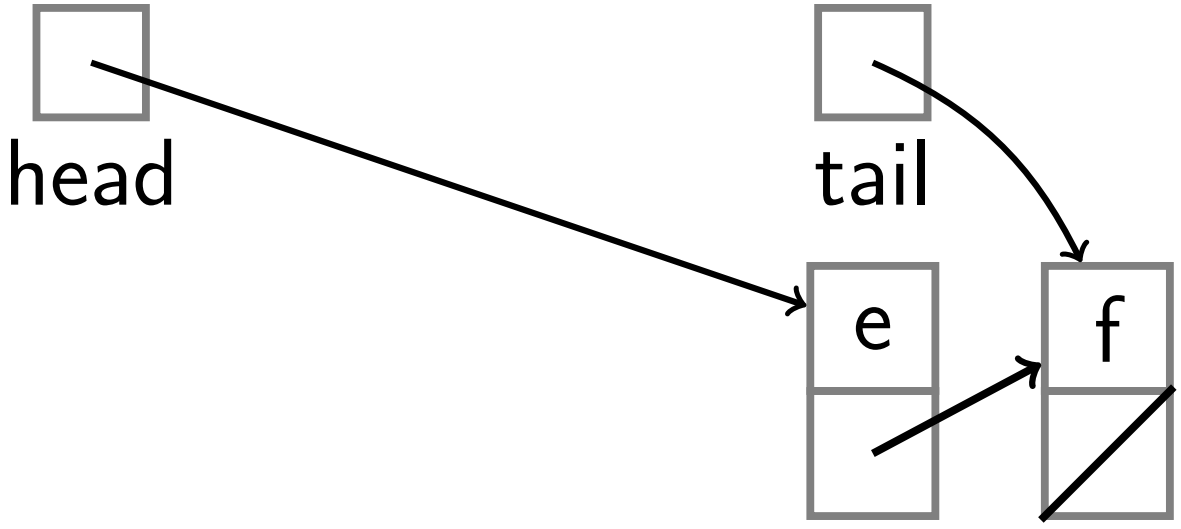


Queue Implementation with Linked List



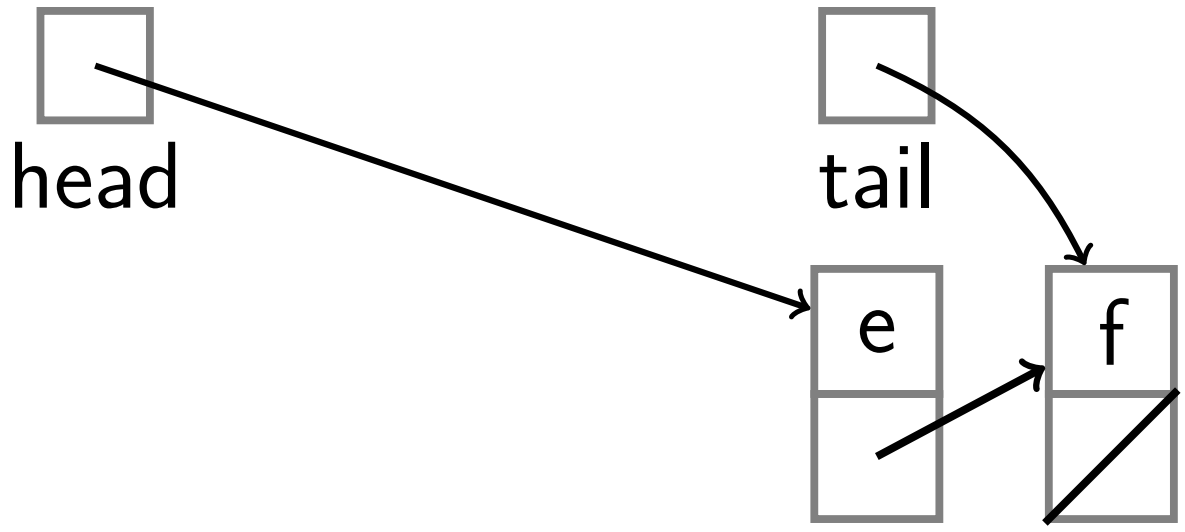
`Dequeue()`

Queue Implementation with Linked List

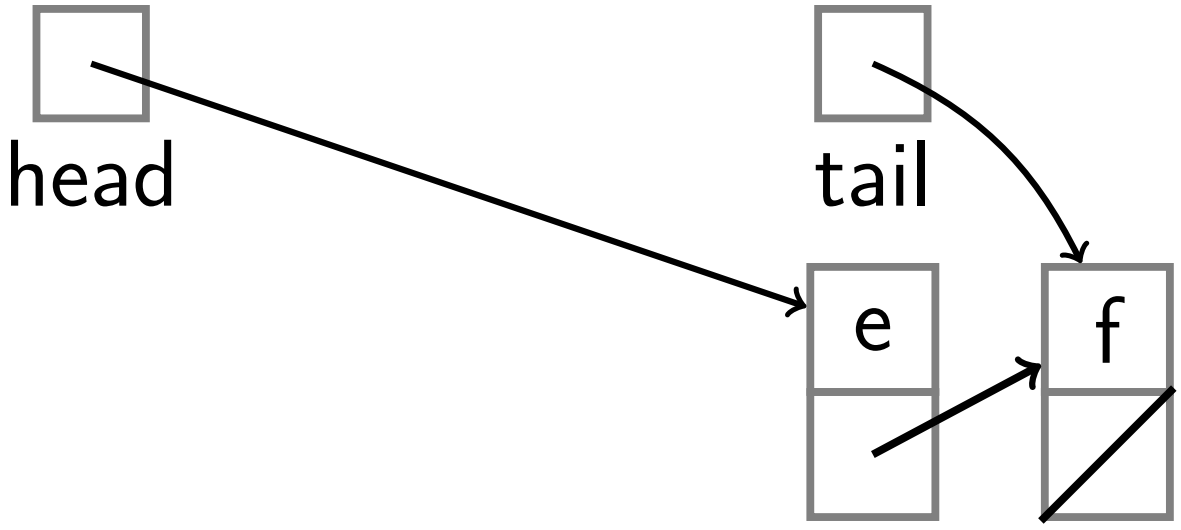


`Dequeue()` $\rightarrow$ `d`

Queue Implementation with Linked List

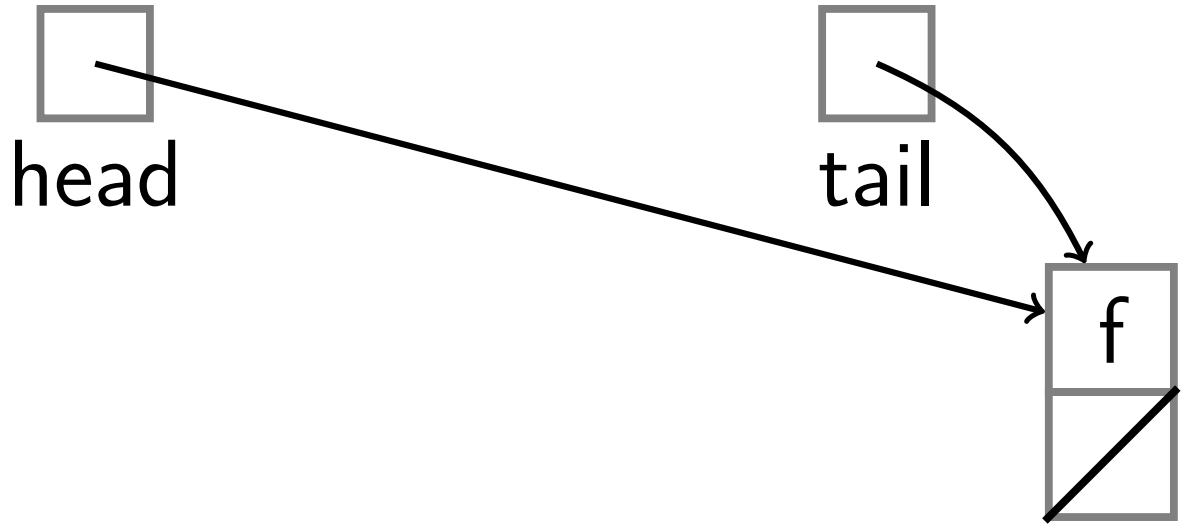


Queue Implementation with Linked List



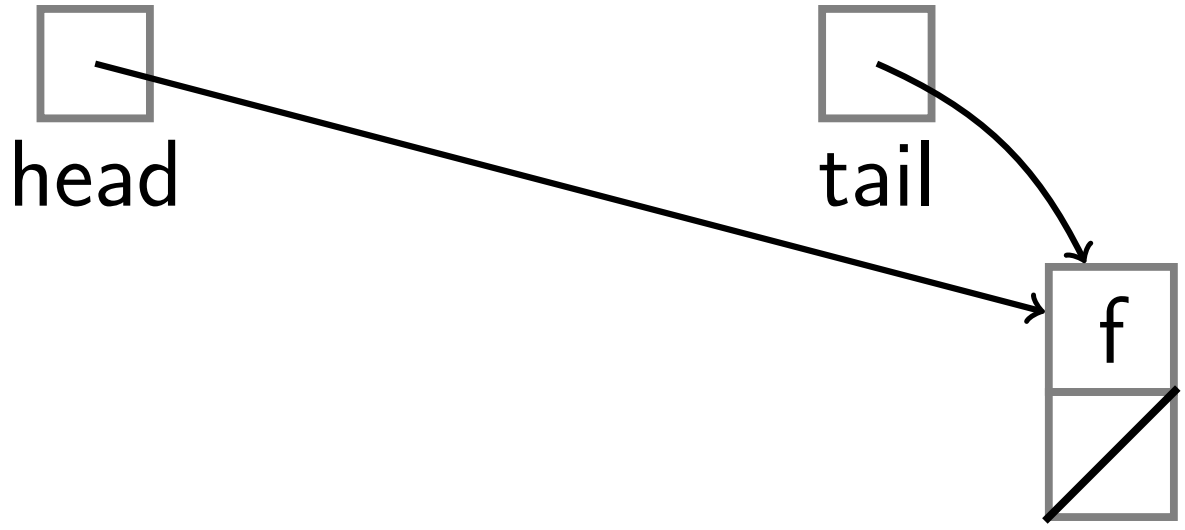
`Dequeue()`

Queue Implementation with Linked List

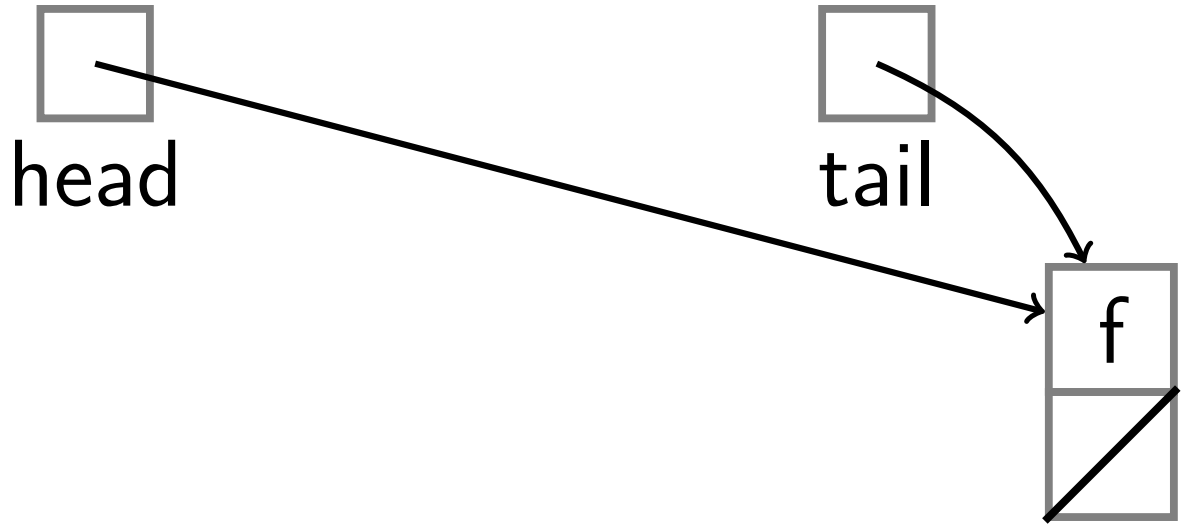


`Dequeue()` $\rightarrow$ `e`

Queue Implementation with Linked List

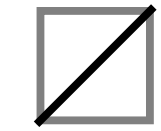


Queue Implementation with Linked List

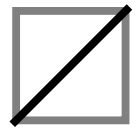


`Dequeue()`

Queue Implementation with Linked List



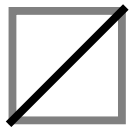
head



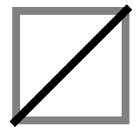
tail

`Dequeue()` $\rightarrow$ `f`

Queue Implementation with Linked List

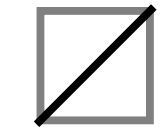


head

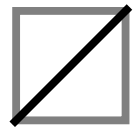


tail

Queue Implementation with Linked List



head

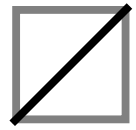


tail

`Empty()`

Queue Implementation with Linked List

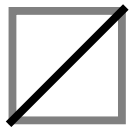
head



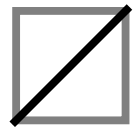
tail

`Empty()` $\rightarrow$ `True`

Queue Implementation with Linked List



head



tail

Queue Implementation with Linked List

- `Enqueue`: use `List.PushBack`

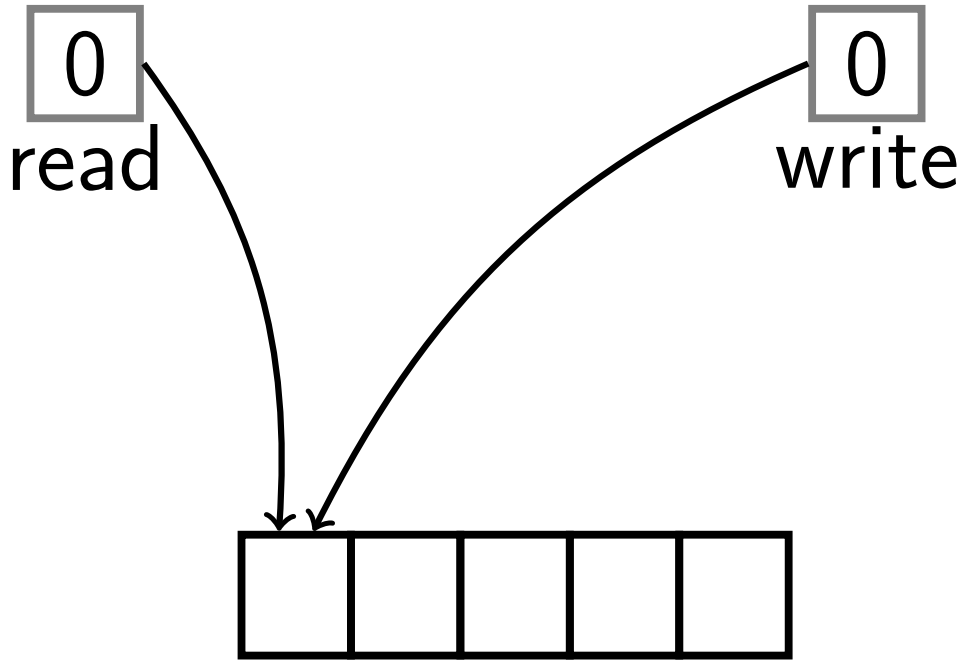
Queue Implementation with Linked List

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

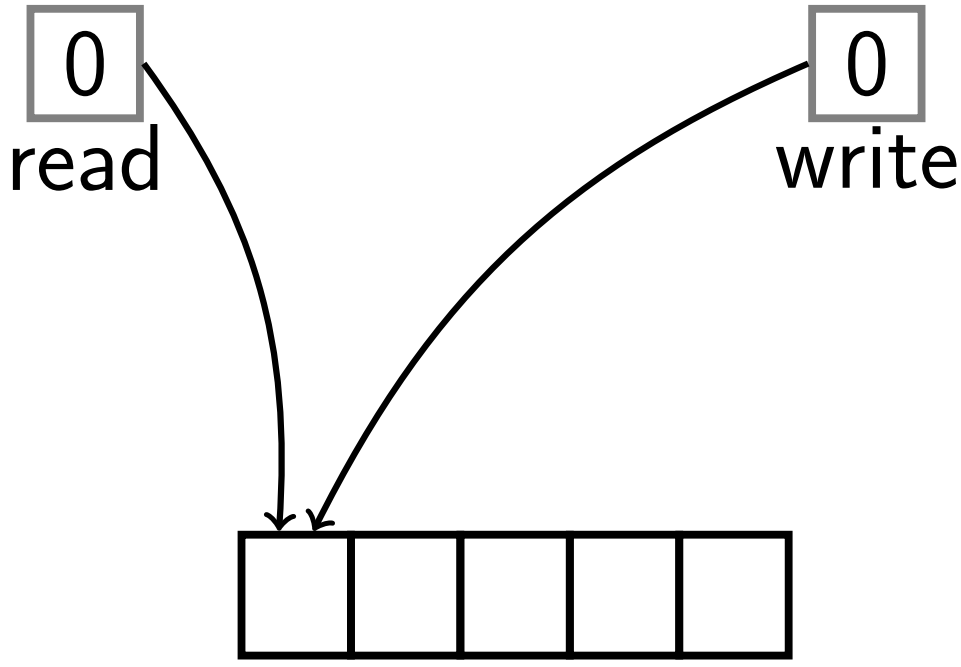
Queue Implementation with Linked List

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

Queue Implementation with Array

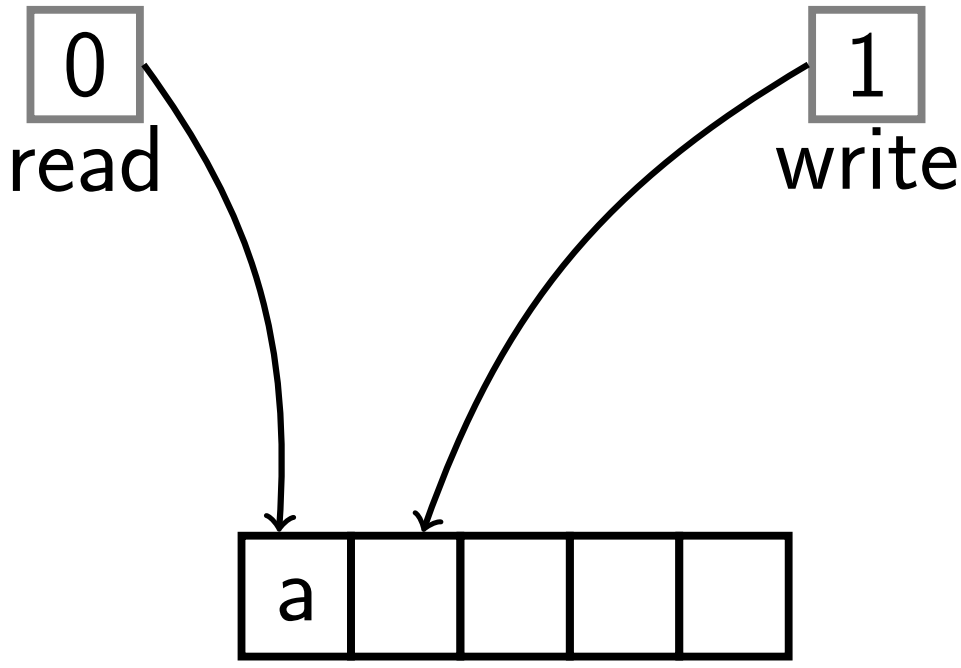


Queue Implementation with Array



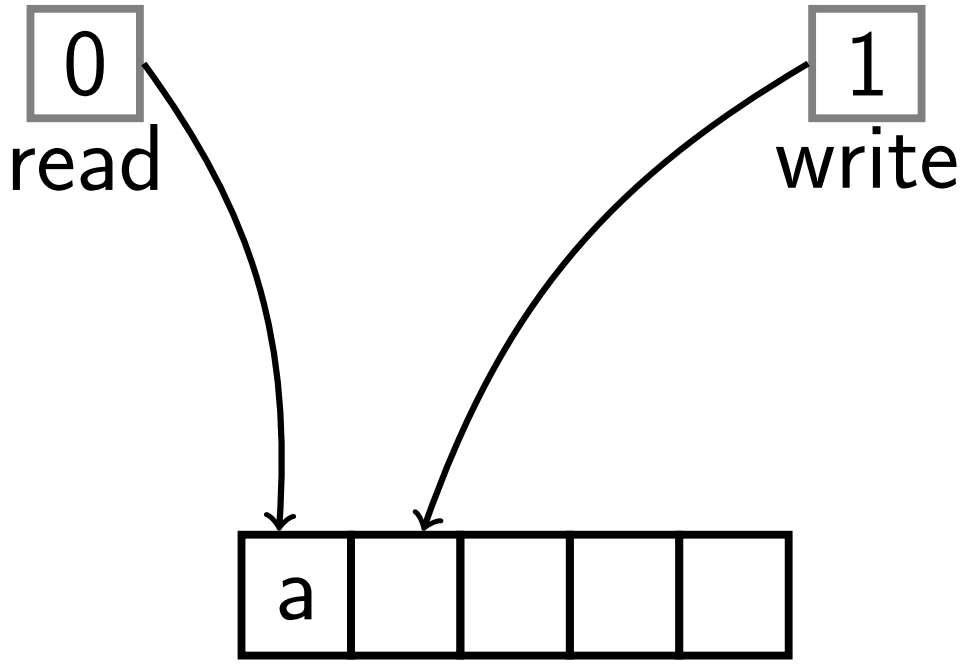
Enqueue(a)

Queue Implementation with Array

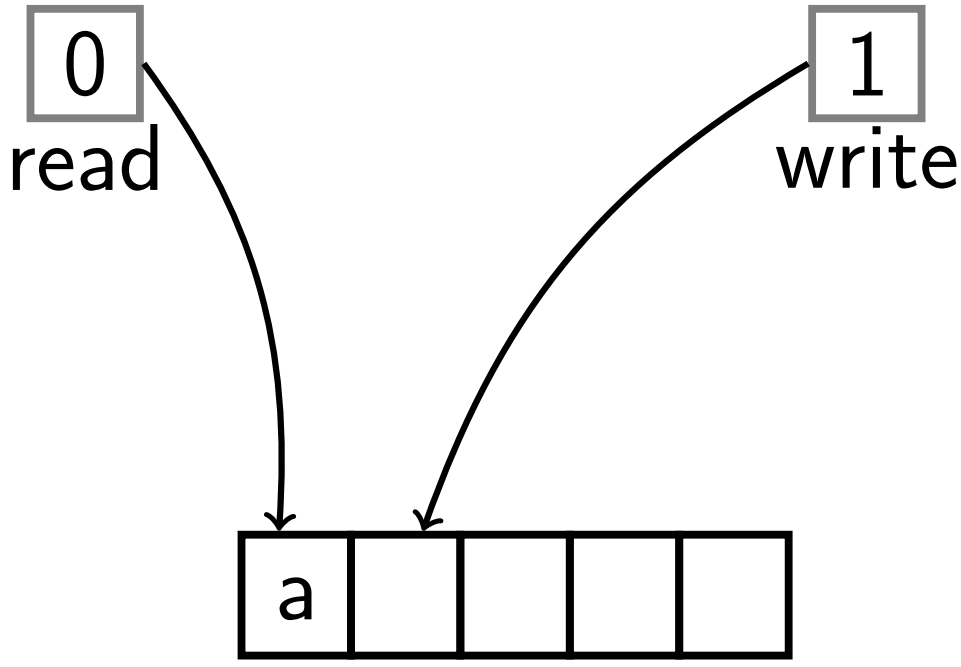


Enqueue(a)

Queue Implementation with Array

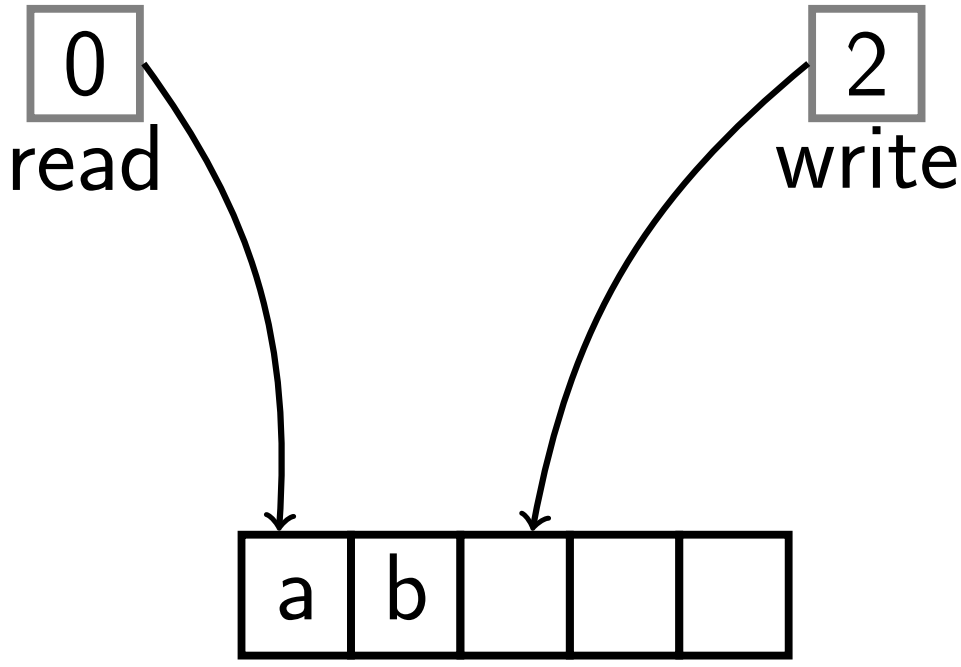


Queue Implementation with Array



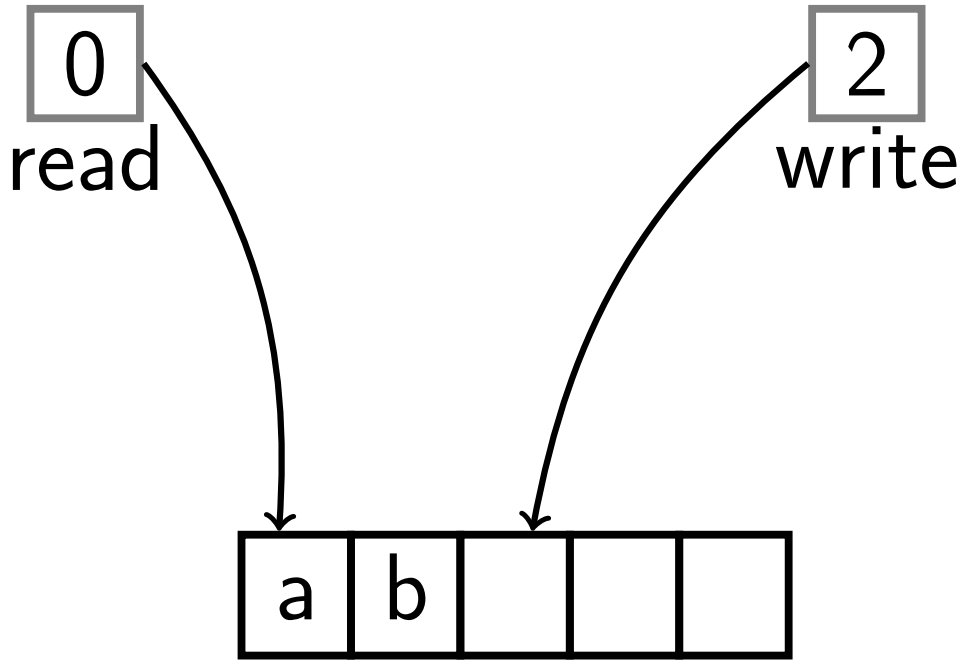
`Enqueue(b)`

Queue Implementation with Array

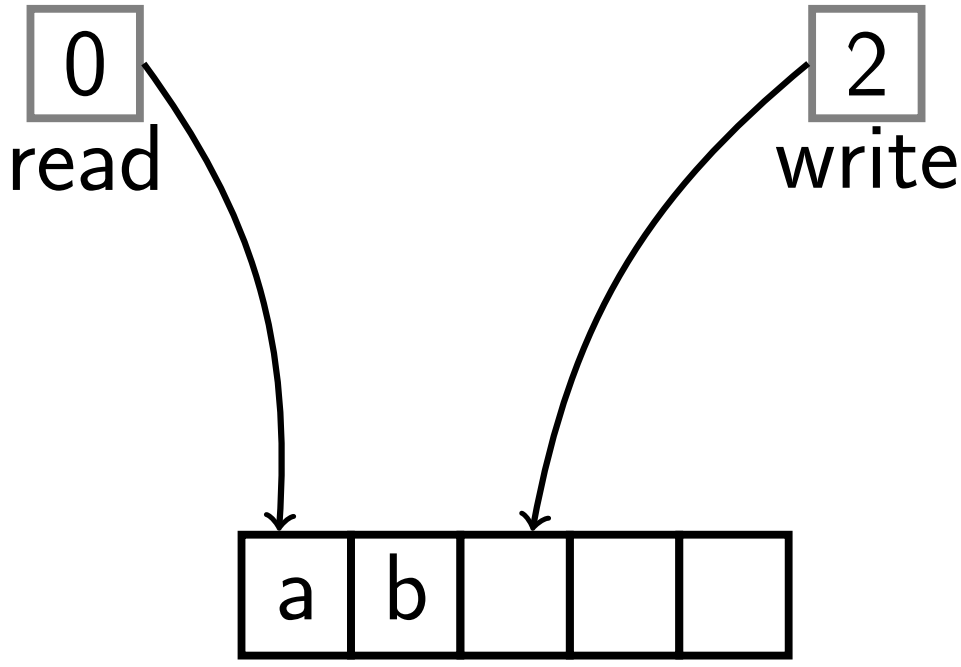


`Enqueue(b)`

Queue Implementation with Array

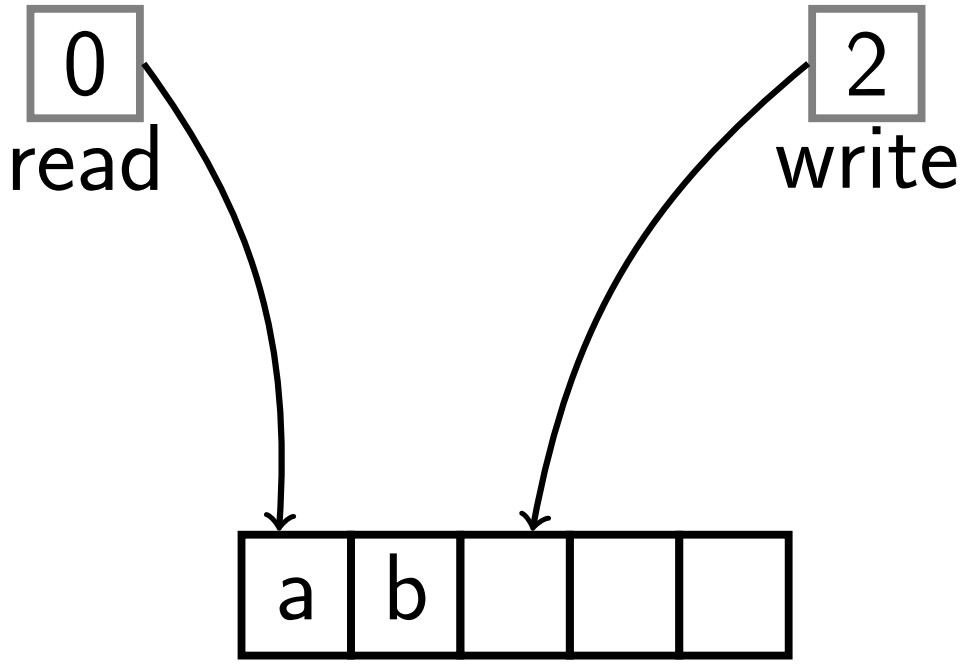


Queue Implementation with Array



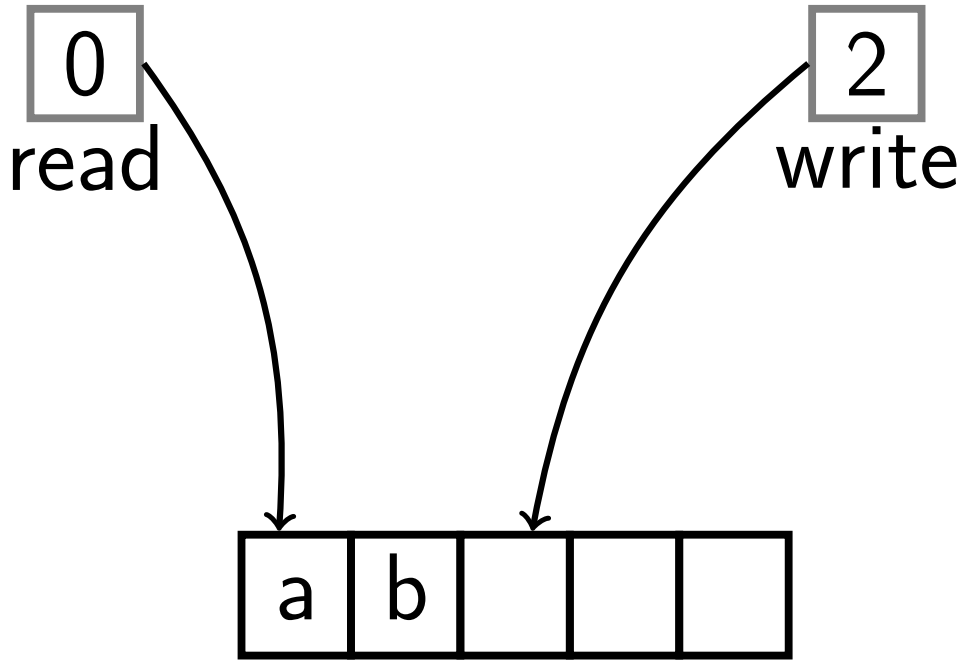
`Empty()`

Queue Implementation with Array

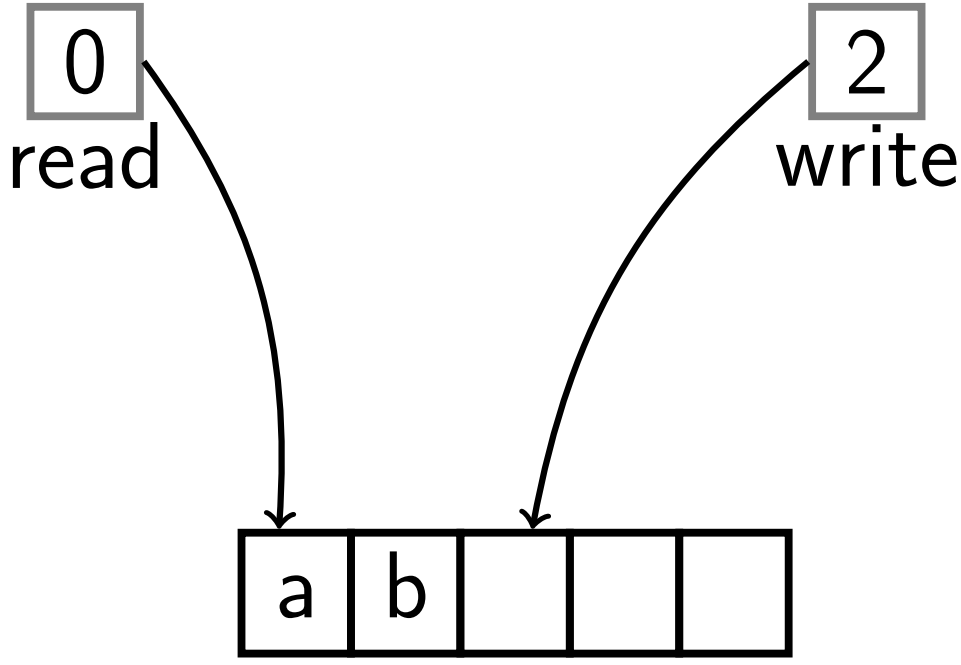


`Empty()` $\rightarrow$ `False`

Queue Implementation with Array

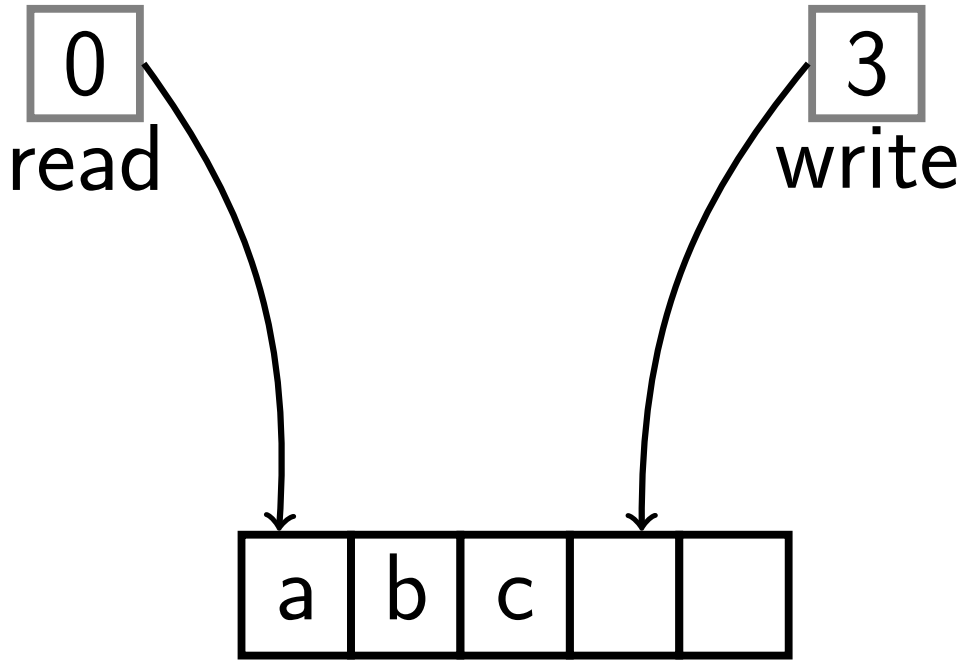


Queue Implementation with Array



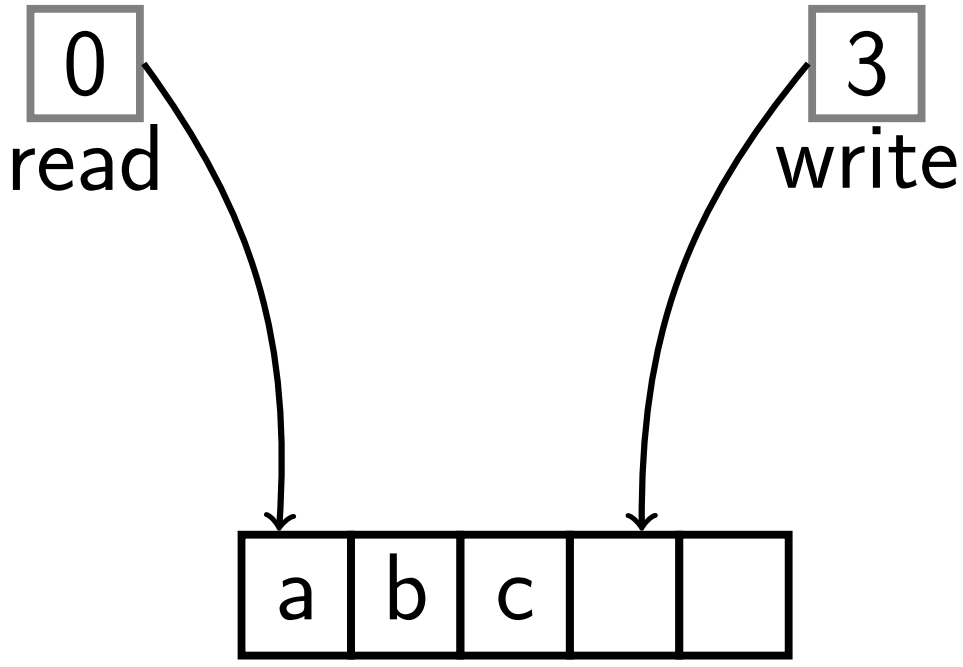
`Enqueue(c)`

Queue Implementation with Array

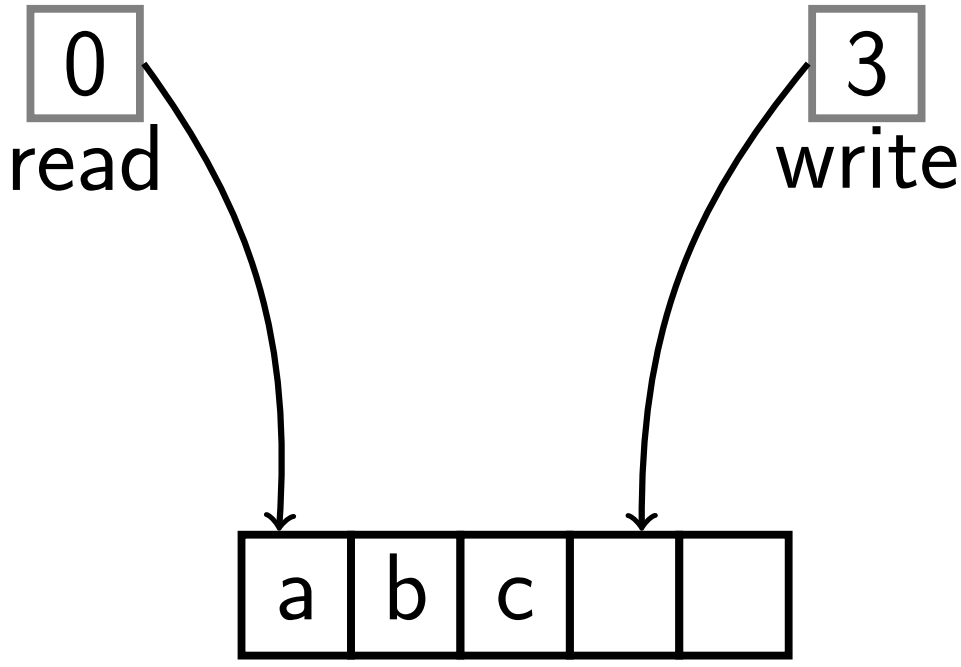


`Enqueue(c)`

Queue Implementation with Array

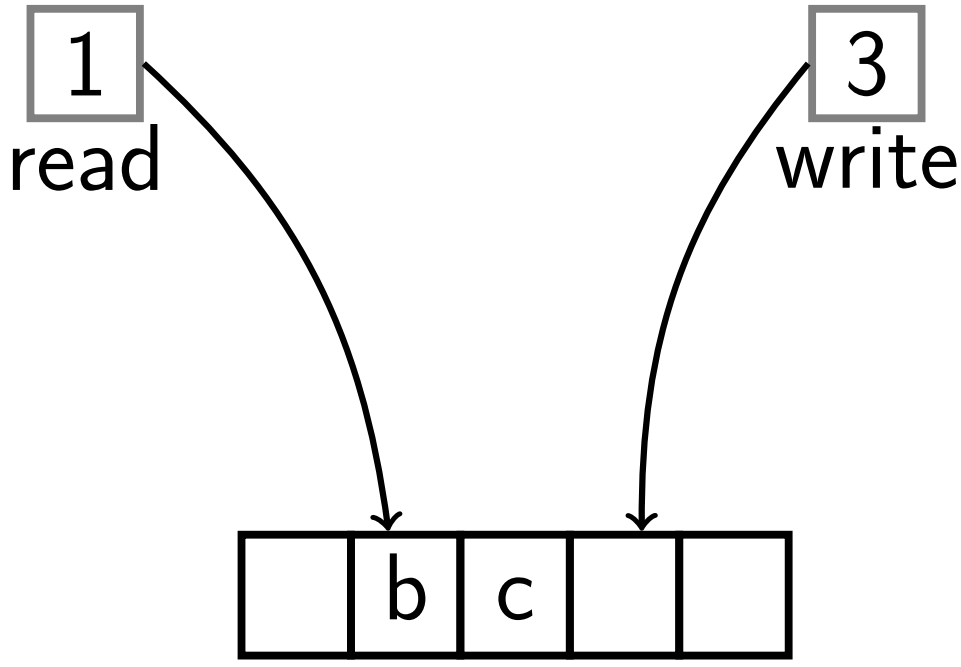


Queue Implementation with Array



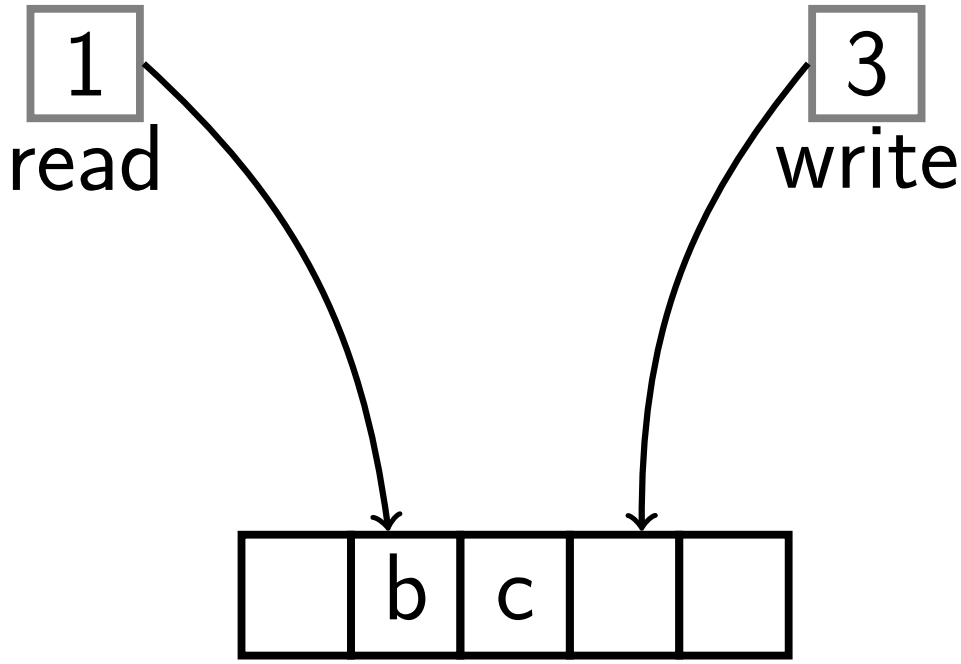
`Dequeue()`

Queue Implementation with Array

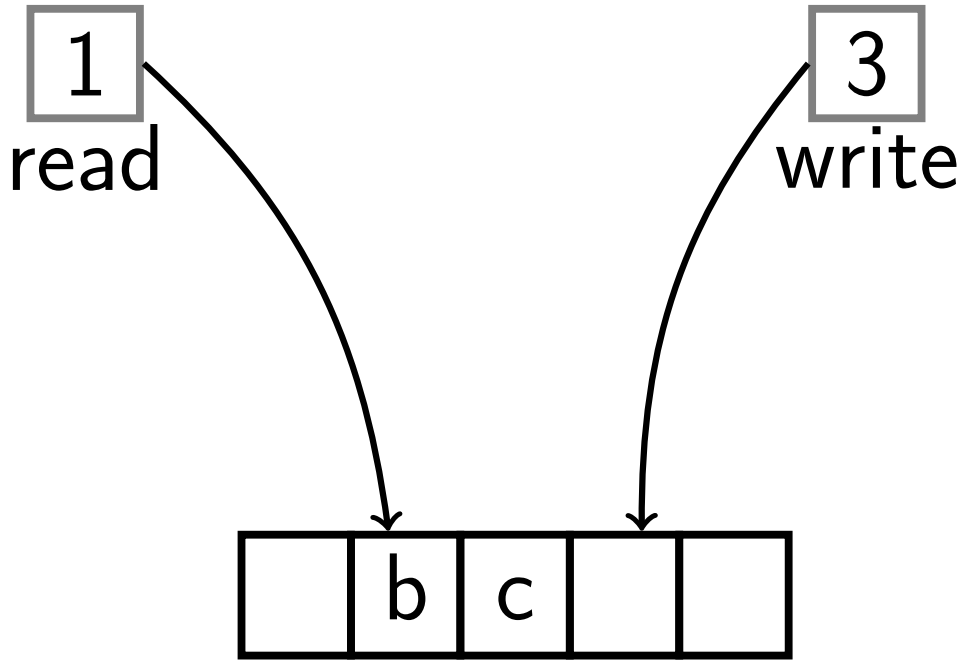


`Dequeue()` $\rightarrow$ `a`

Queue Implementation with Array

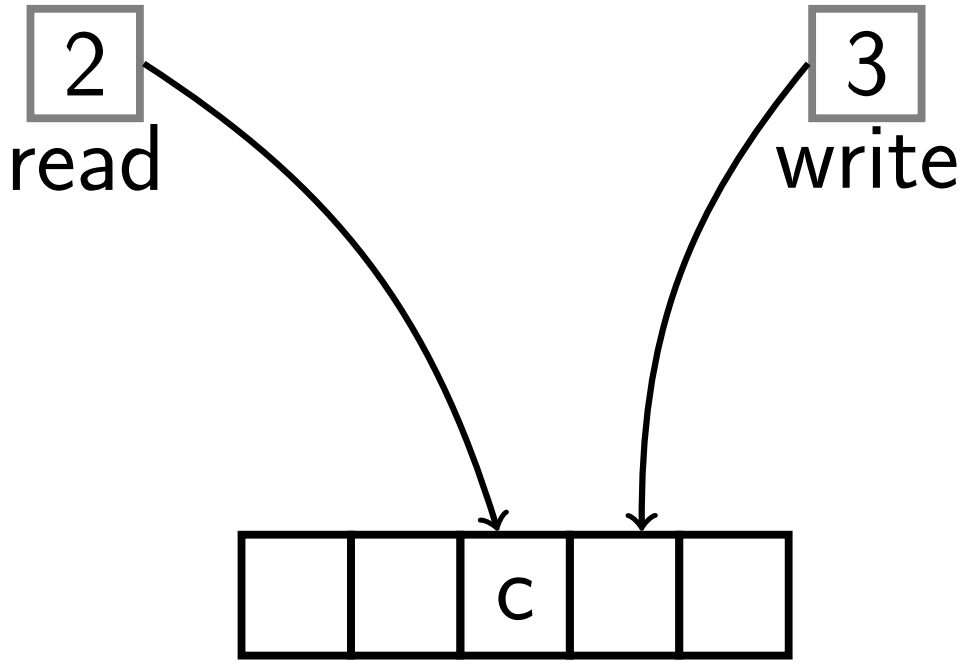


Queue Implementation with Array



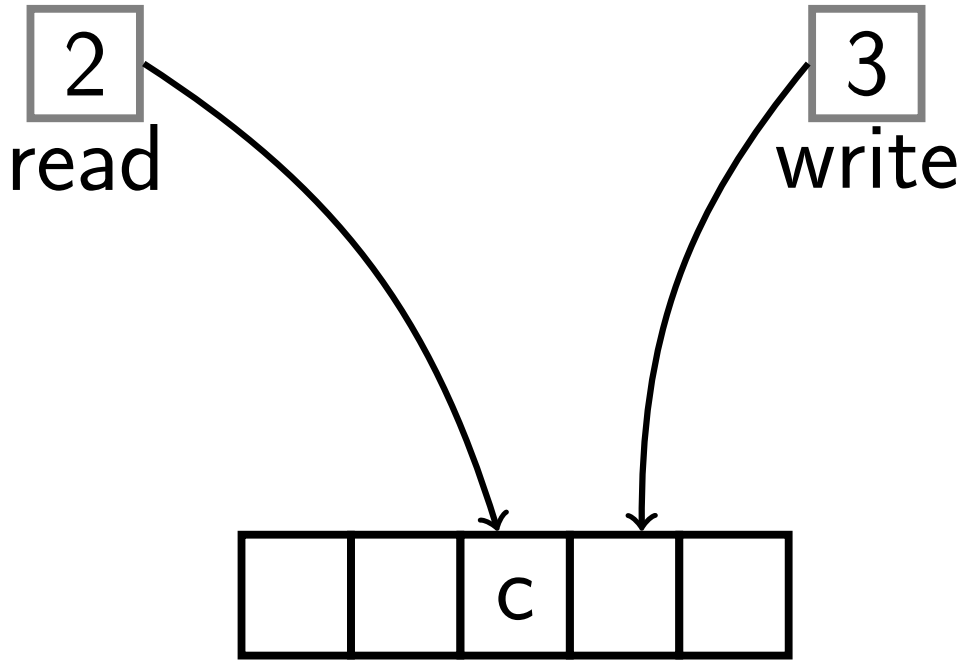
`Dequeue()`

Queue Implementation with Array

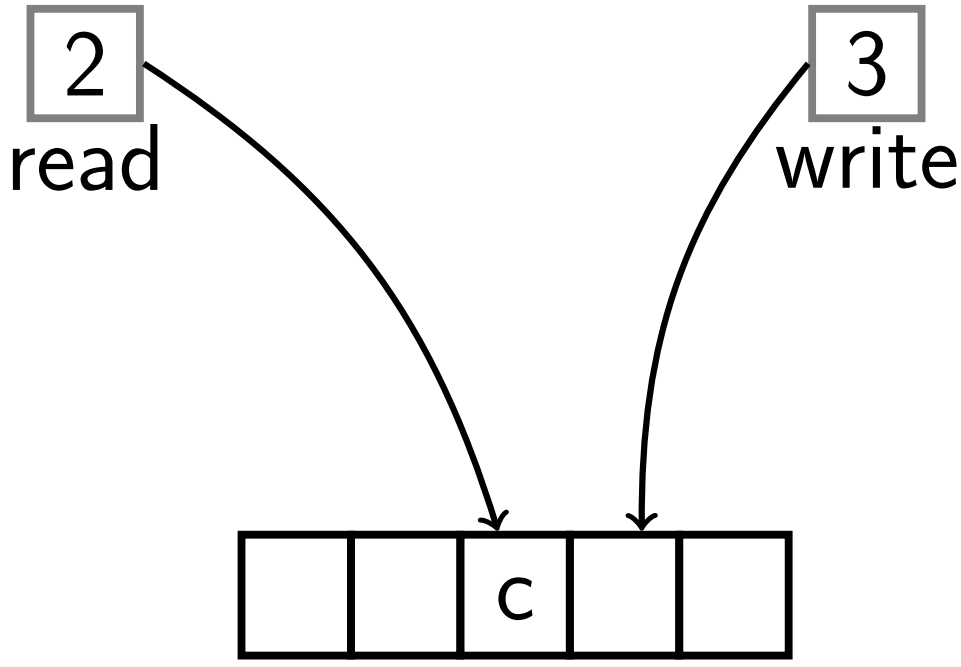


`Dequeue()` $\rightarrow$ `b`

Queue Implementation with Array

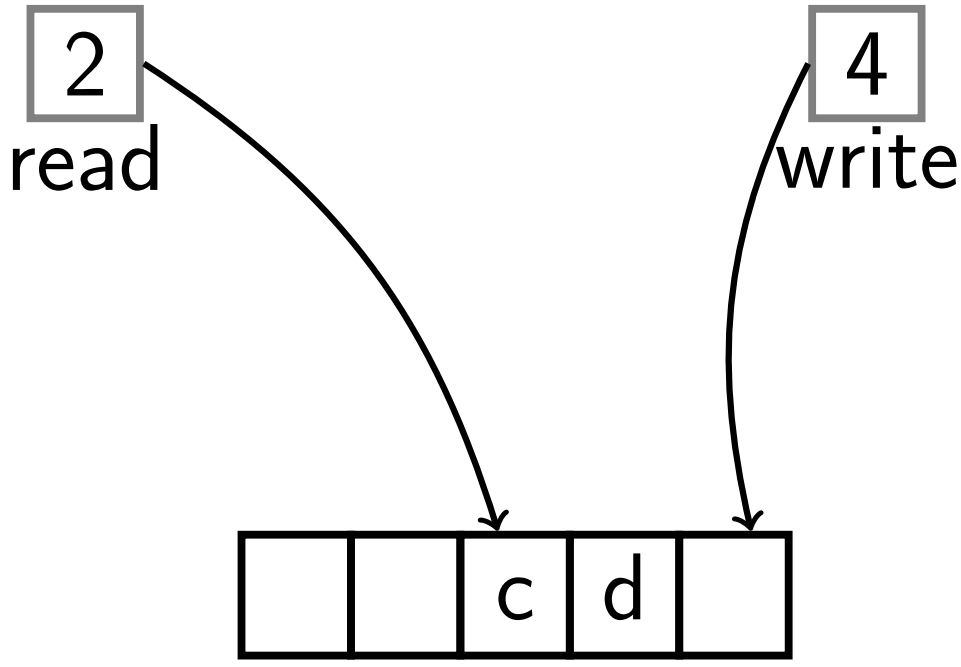


Queue Implementation with Array



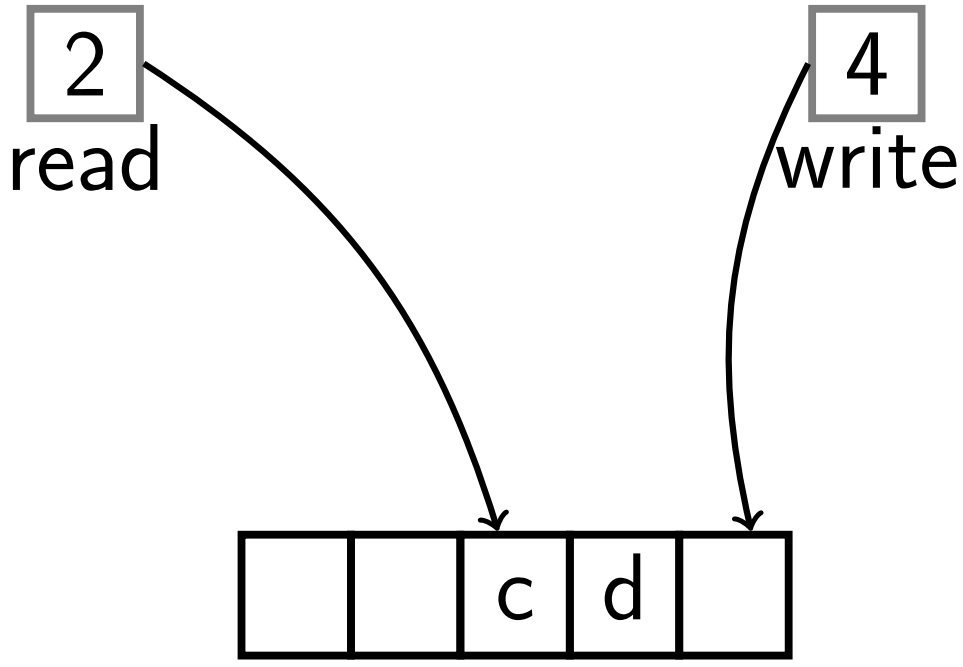
`Enqueue(d)`

Queue Implementation with Array

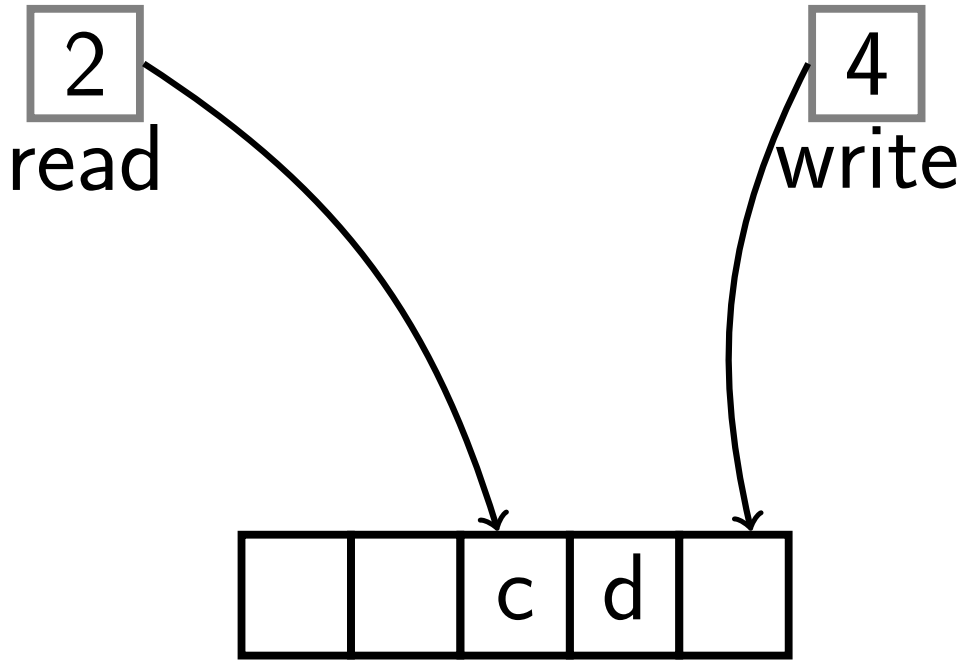


`Enqueue(d)`

Queue Implementation with Array

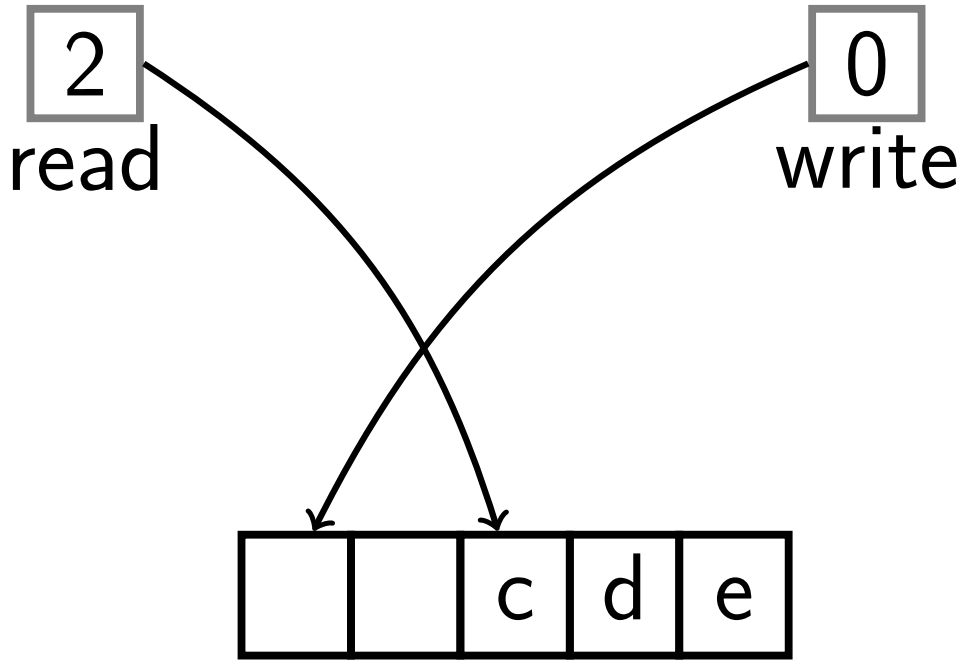


Queue Implementation with Array



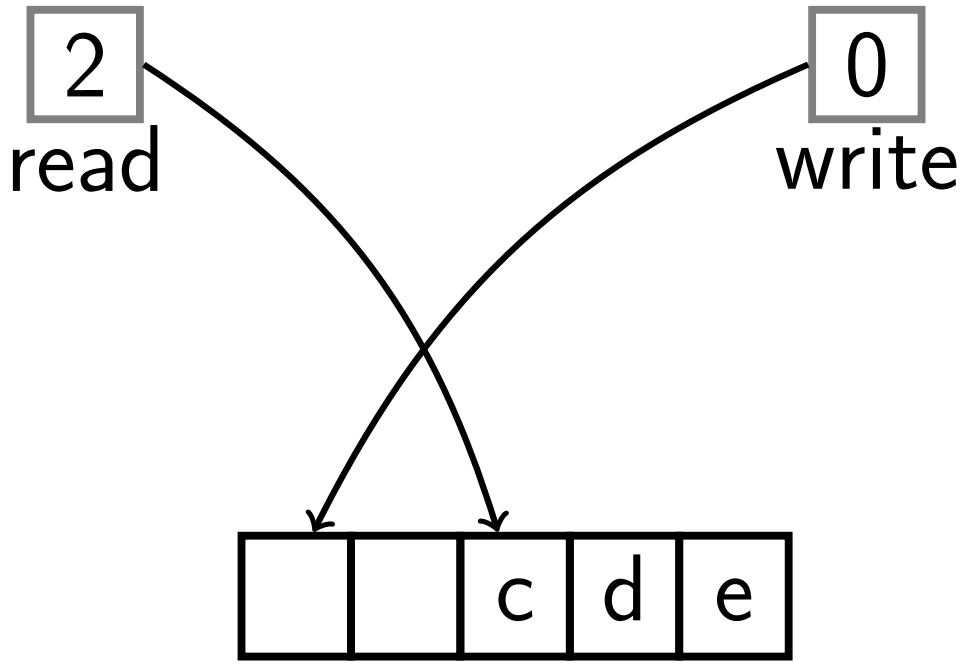
`Enqueue(e)`

Queue Implementation with Array

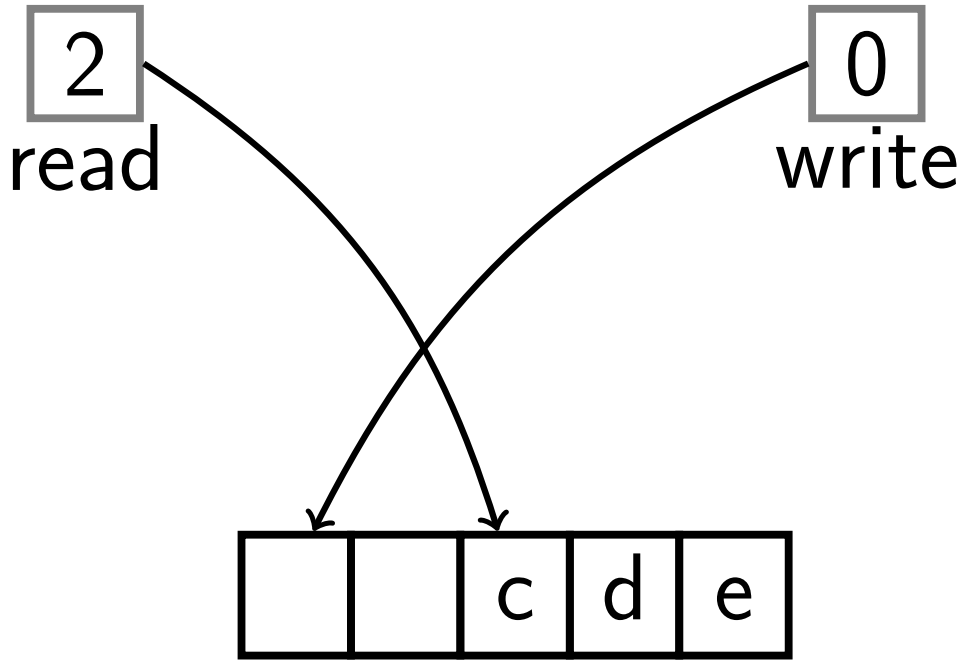


`Enqueue(e)`

Queue Implementation with Array

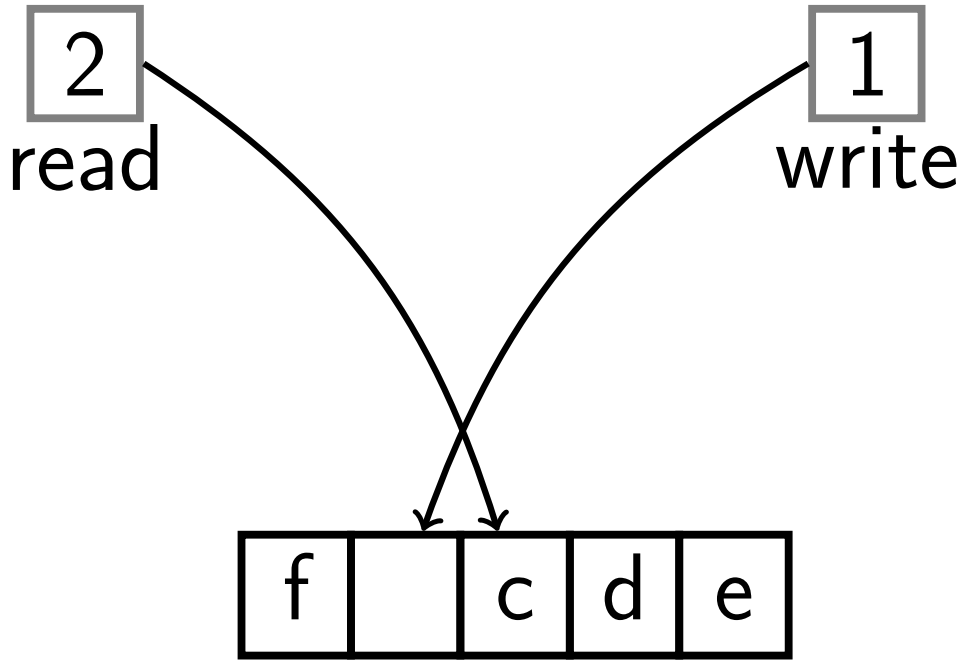


Queue Implementation with Array



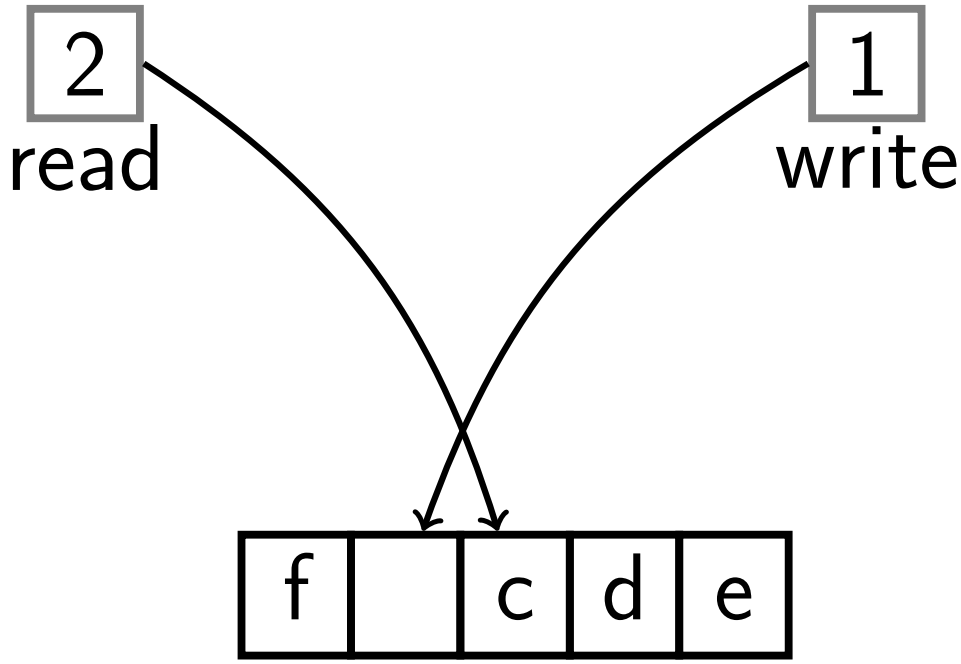
`Enqueue(f)`

Queue Implementation with Array

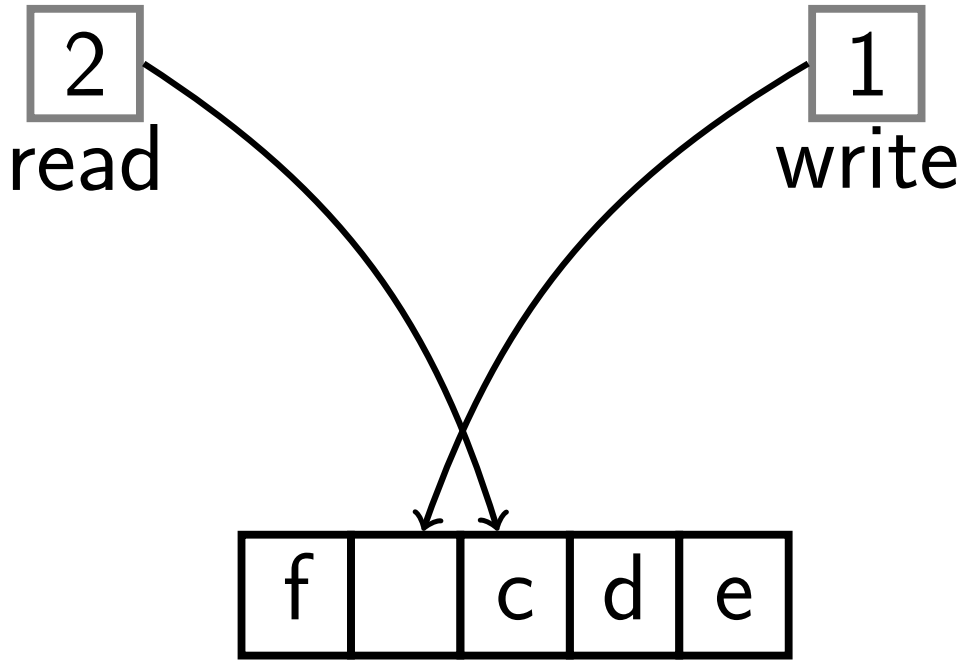


`Enqueue(f)`

Queue Implementation with Array

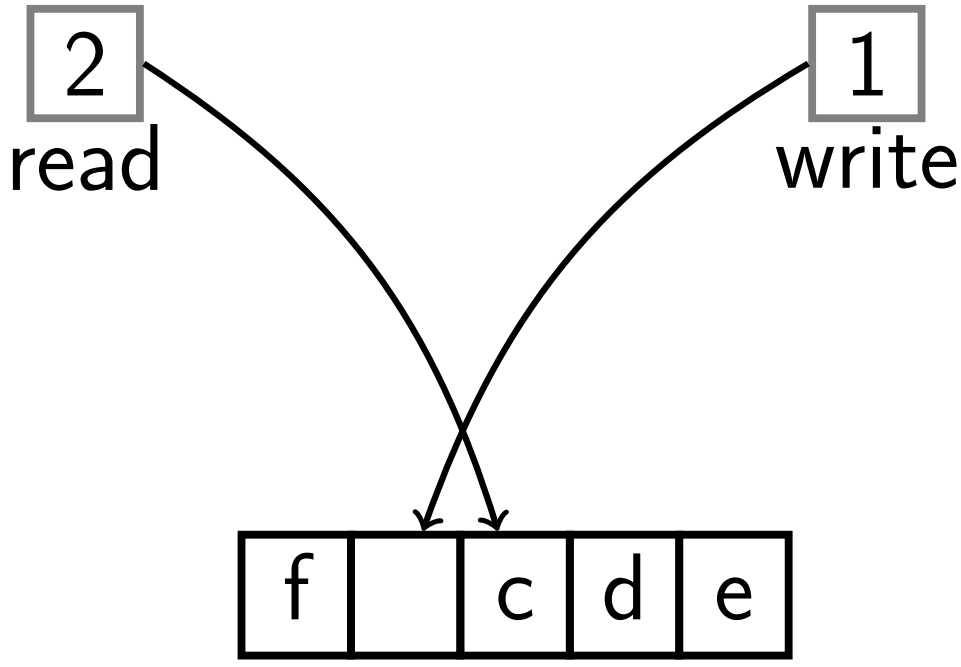


Queue Implementation with Array



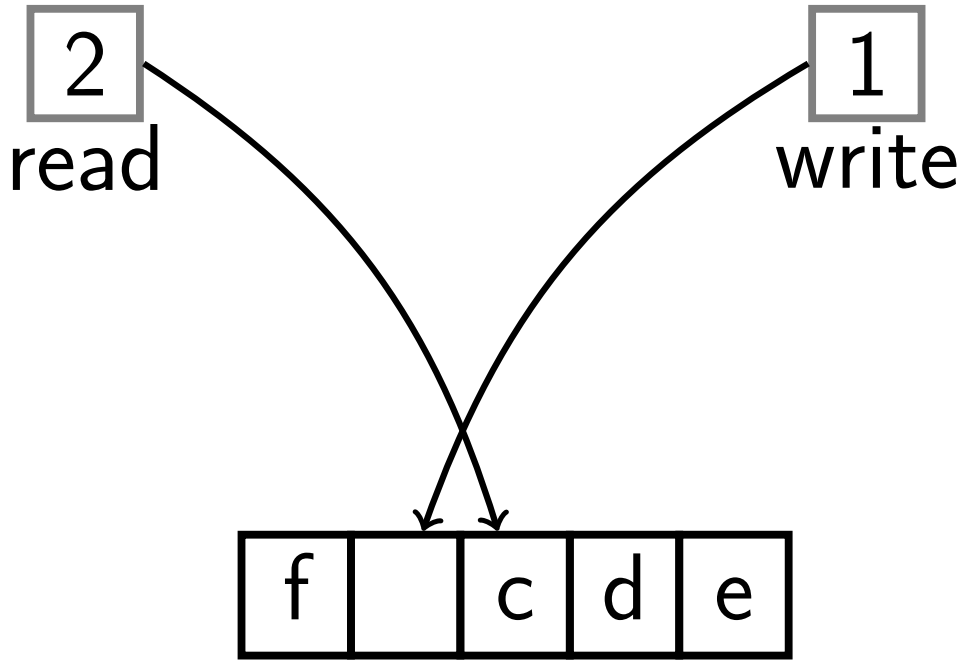
`Enqueue(g)`

Queue Implementation with Array

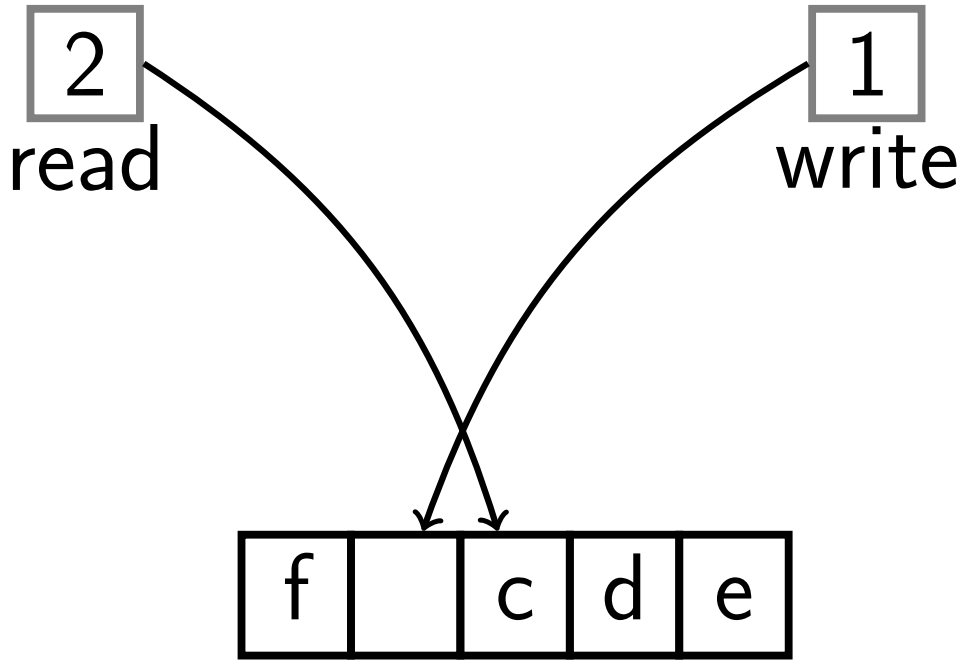


`Enqueue(g)` $\rightarrow$ `ERROR`

Queue Implementation with Array

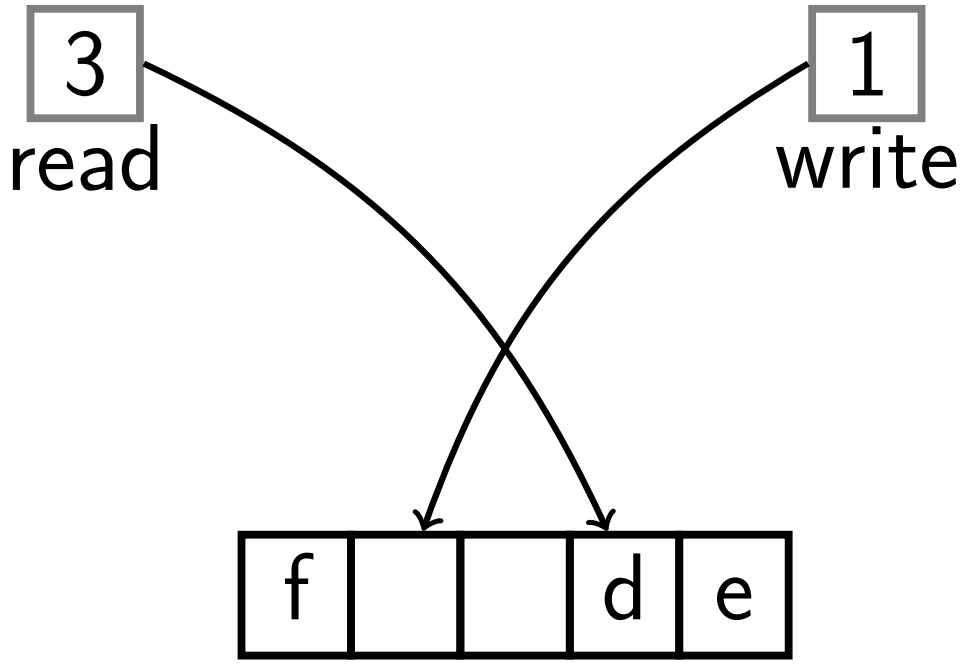


Queue Implementation with Array



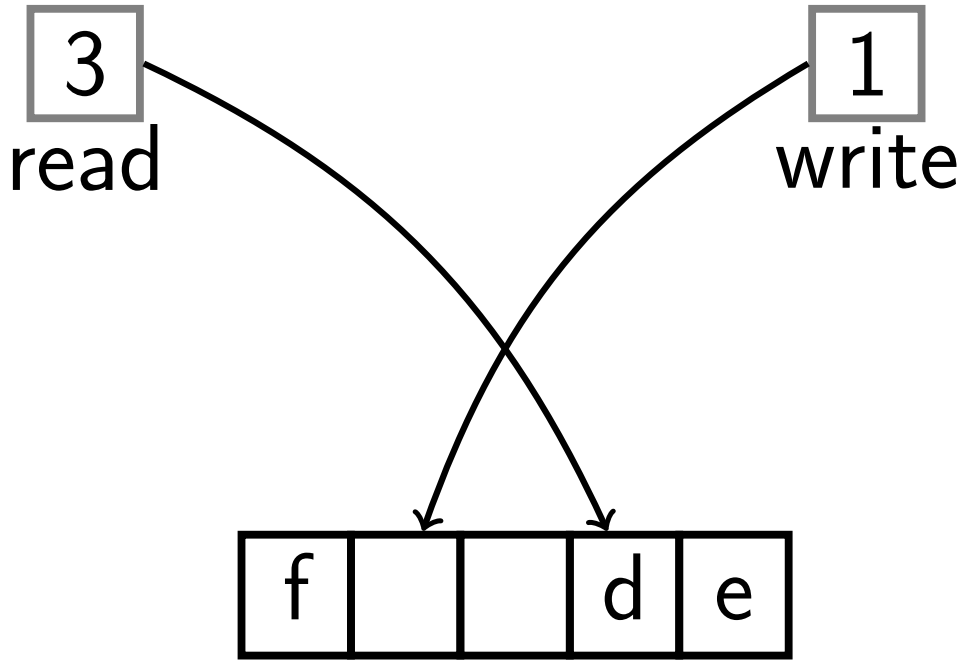

```
Dequeue()
```

Queue Implementation with Array

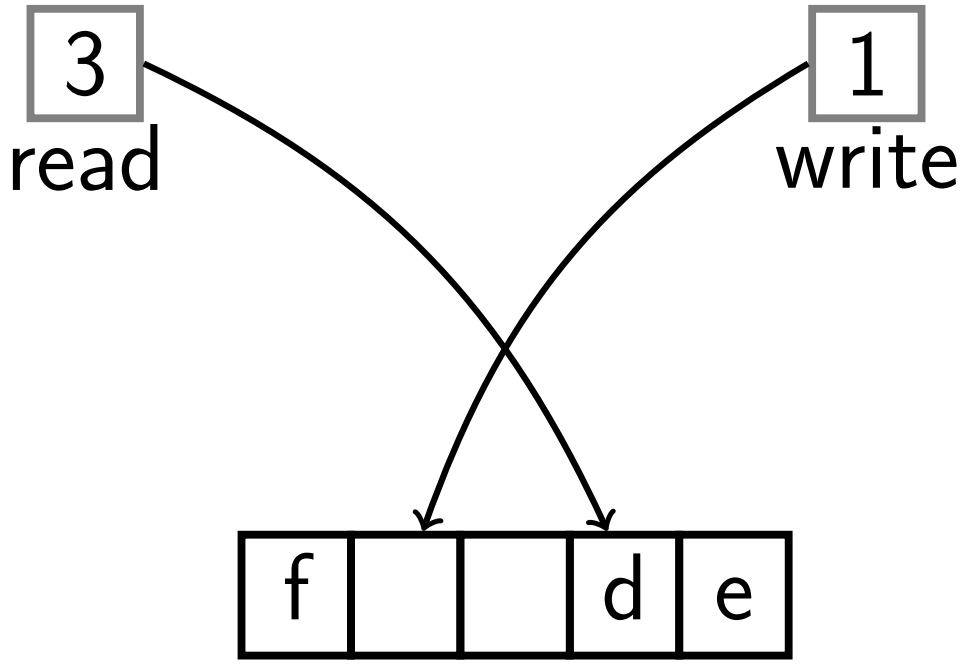


`Dequeue()` $\rightarrow$ `c`

Queue Implementation with Array

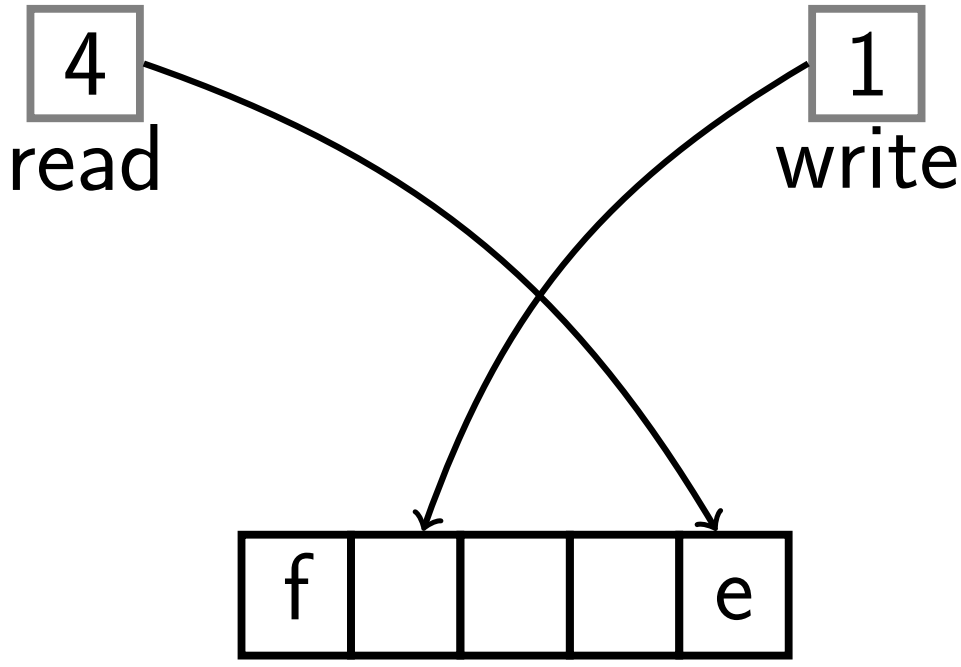


Queue Implementation with Array



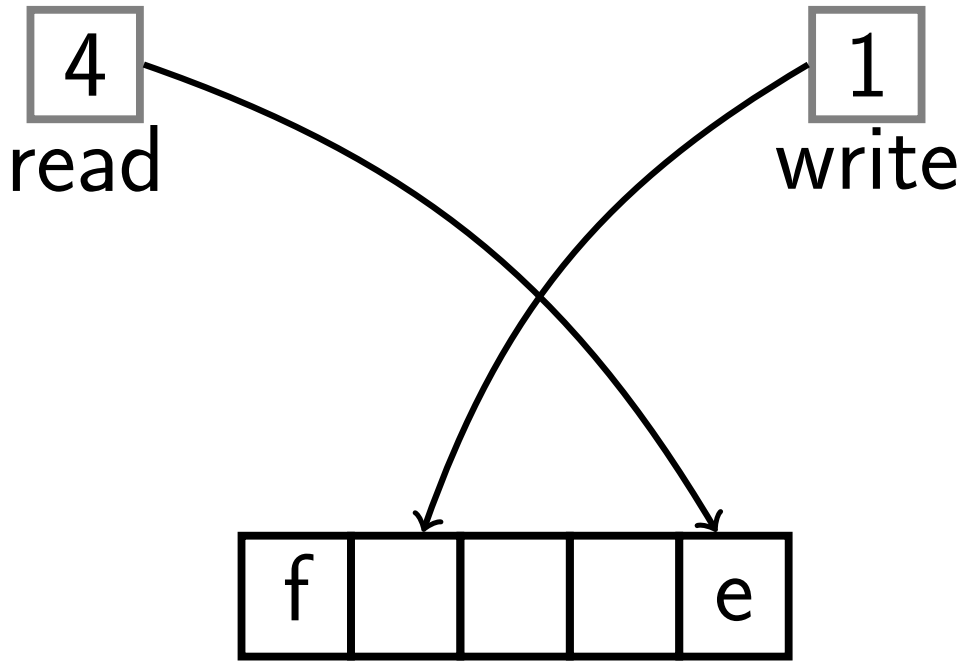
`Dequeue()`

Queue Implementation with Array

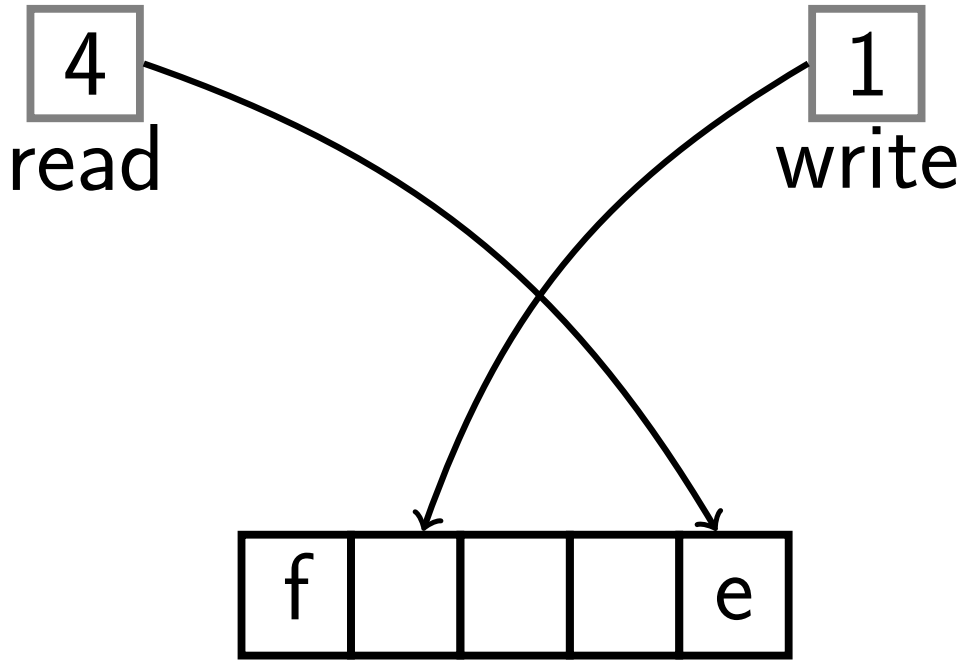


`Dequeue()` $\rightarrow$ `d`

Queue Implementation with Array

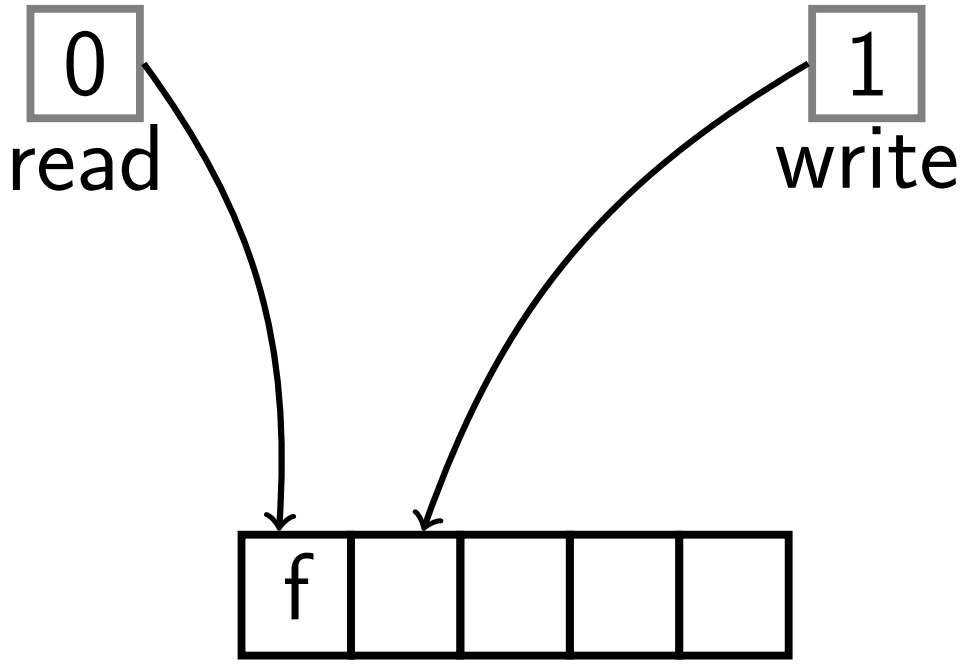


Queue Implementation with Array



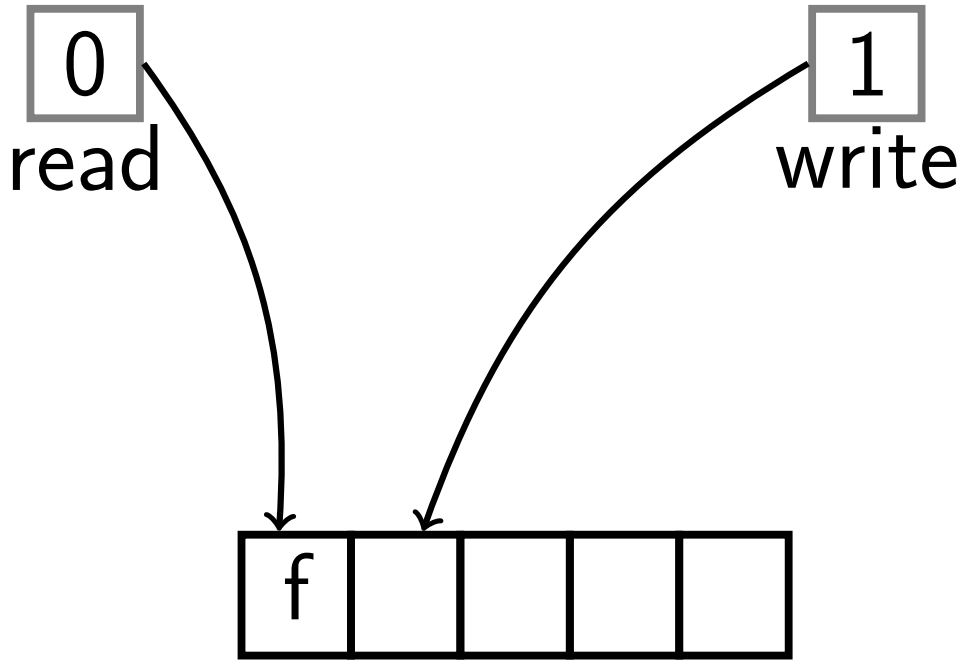

```
Dequeue()
```

Queue Implementation with Array

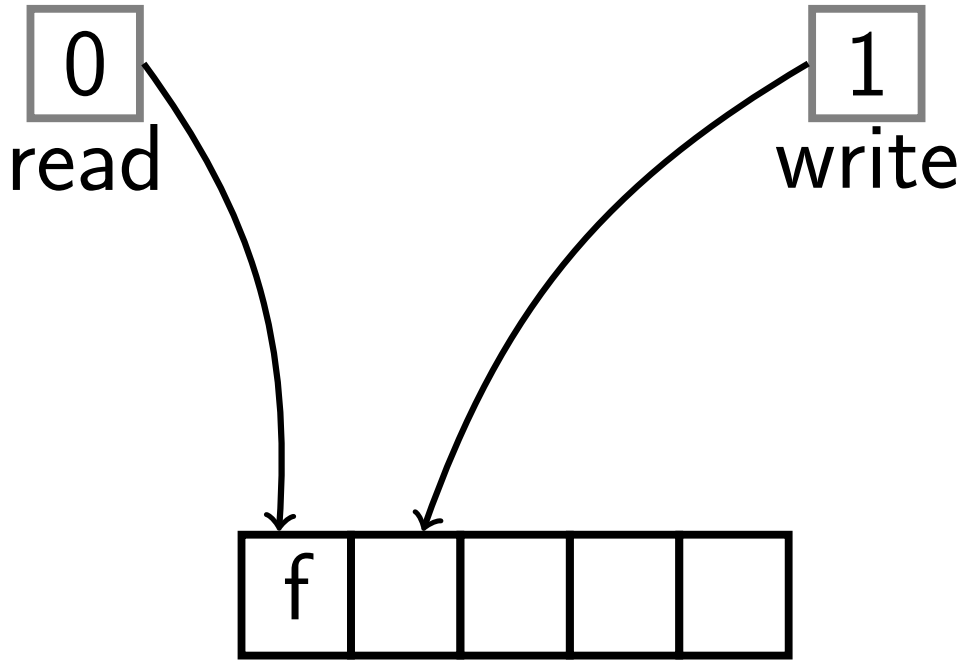


`Dequeue()` $\rightarrow$ `e`

Queue Implementation with Array

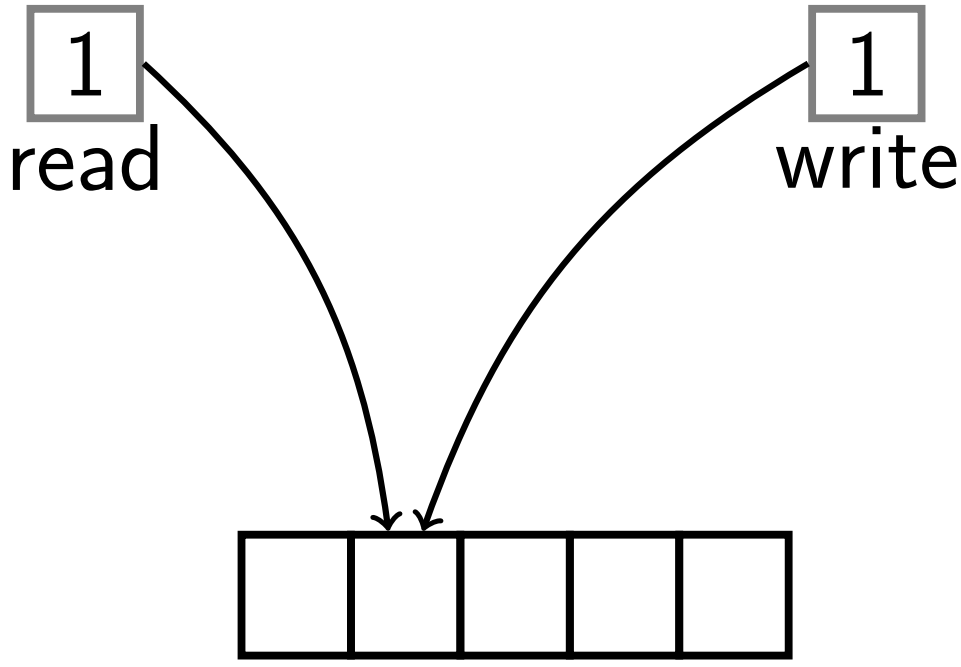


Queue Implementation with Array



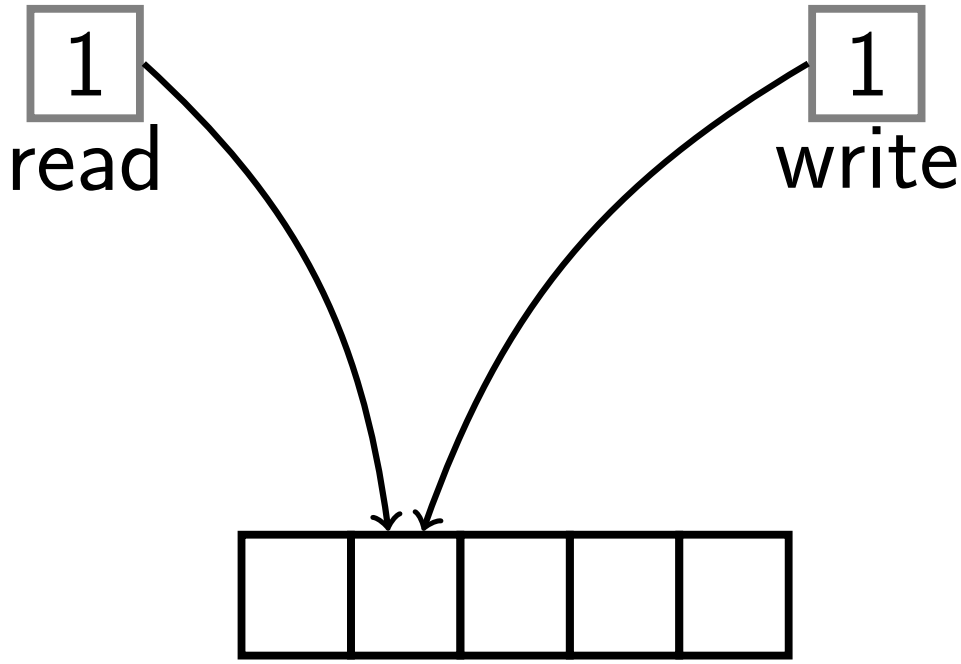
`Dequeue()`

Queue Implementation with Array

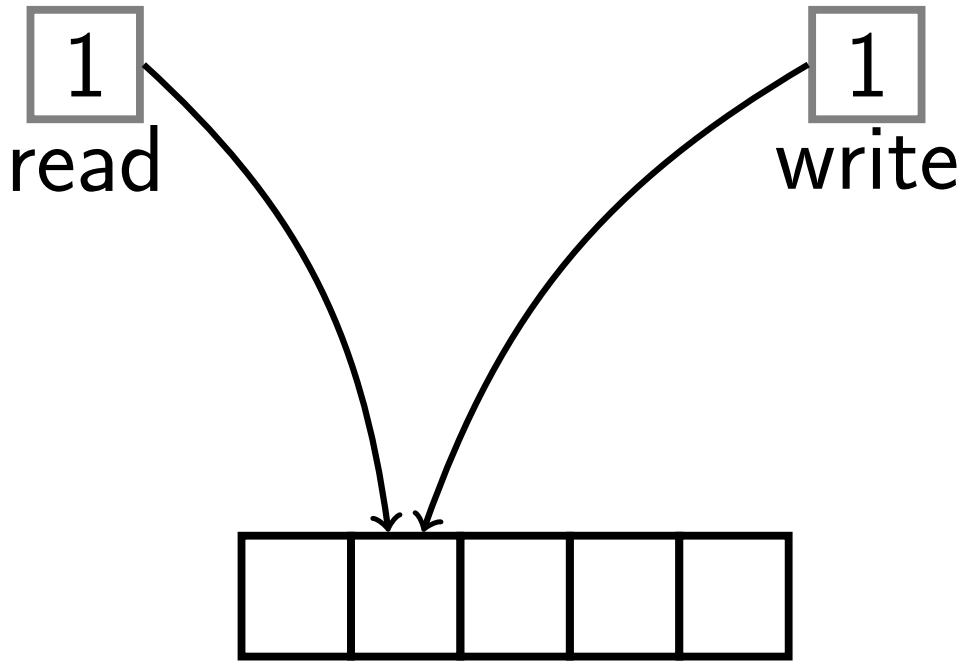


`Dequeue()` $\rightarrow$ `f`

Queue Implementation with Array

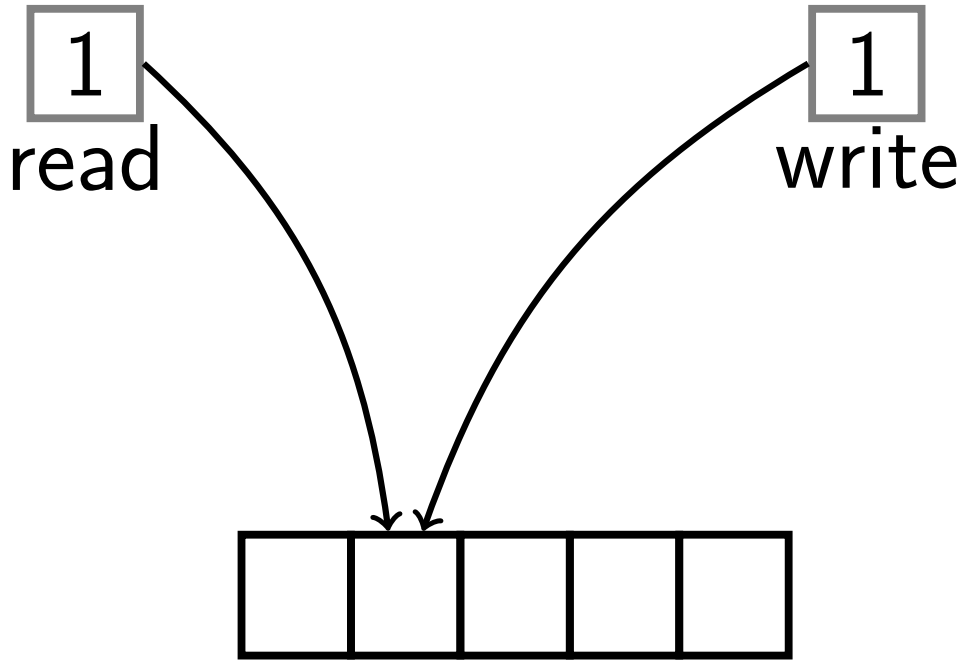


Queue Implementation with Array



`Empty()`

Queue Implementation with Array



`Empty()` $\rightarrow$ `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.