

CSCI 132:

Basic Data Structures and Algorithms

Stacks and Queues Conclusion, Priority Queue


Reese Pearsall & Iliana Castillon
Fall 2024

Announcements

Program 3 due **Friday** at 11:59 PM

Margot Lee Shetterly (author of hidden figures) talk
this Wednesday @ 6pm

MONTANA STATE UNIVERSITY





CROSSING BOUNDARIES SPEAKER SERIES

Margot Lee Shetterly
Author of

Hidden Figures:
The American Dream and the Untold Story of the Black Women
Mathematicians Who Helped Win the Space Race

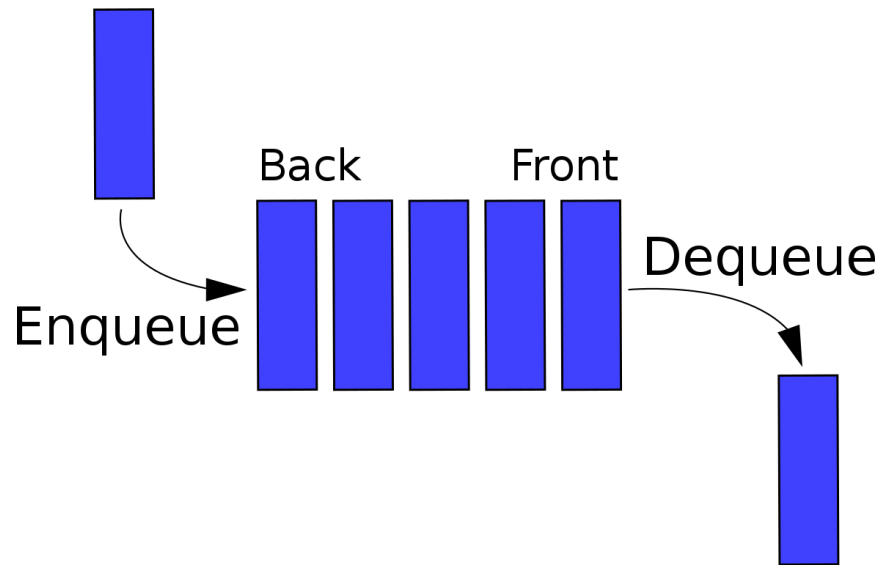
WEDNESDAY
OCTOBER 30, 2024
6PM / SUB BALLROOMS
FREE EVENT OPEN TO ALL, NO TICKETS REQUIRED




MONTANA
STATE UNIVERSITY

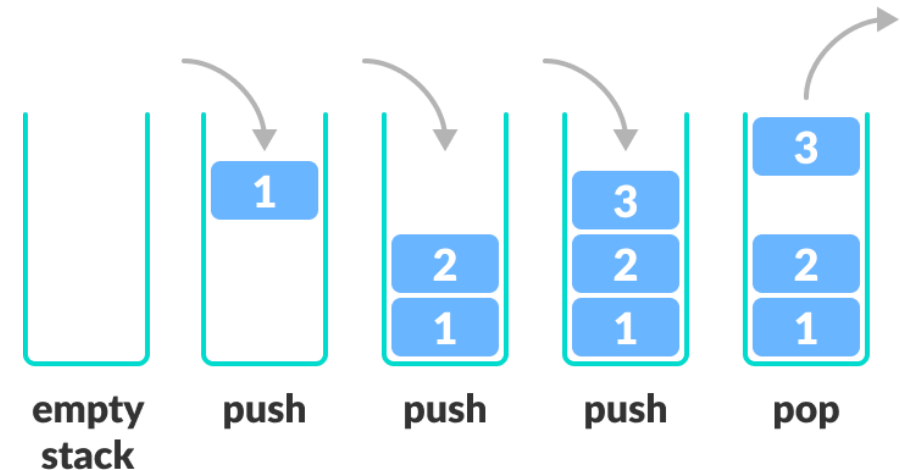
Co-sponsored by the
Office of the President
and the Office of
Diversity & Inclusion

A **Queue** is a data structure that holds data, but operates in a First-in First-out (**FIFO**) fashion



`enqueue()` , `dequeue()`

A **stack** is a data structure that can hold data, and follows the **last in first out (LIFO)** principle



`push()` , `pop()`

We implemented both data structures using an Array or a Linked List

Takeaway: Adding and removing elements from a **queue** runs in constant time ($O(1)$)

(FIFO)

Takeaway: Adding and removing elements from a **stack** runs in constant time ($O(1)$)

(LIFO)

Queue Runtime Analysis

	Linked List	Array
Creation	$O(1)$	$O(n)$
Enqueue	$O(1)$	$O(1)$
Dequeue	$O(1)$	$O(1)$
Peek	$O(1)$	$O(1)$
Print Queue	$O(n)$	$O(n)$

Stack Runtime Analysis

	w/ Array	w/ Linked List
Creation	$O(n)$	$O(1)$
Push()	$O(1)$	$O(1)$
Pop()	$O(1)$	$O(1)$
peek()	$O(1)$	$O(1)$
Print()	$O(n)$	$O(n)$

Which data structure should *you* use?

it depends

Data structures always have tradeoffs.

With stacks and queues, the important thing to consider is **the order** of how you want your data to be read

Stacks → LIFO
Queues → FIFO*

Queue Runtime Analysis

	Linked List	Array
Creation	$O(1)$	$O(n)$
Enqueue	$O(1)$	$O(1)$
Dequeue	$O(1)$	$O(1)$
Peek	$O(1)$	$O(1)$
Print Queue	$O(n)$	$O(n)$

Stack Runtime Analysis

	w/ Array	w/ Linked List
Creation	$O(n)$	$O(1)$
Push()	$O(1)$	$O(1)$
Pop()	$O(1)$	$O(1)$
peek()	$O(1)$	$O(1)$
Print()	$O(n)$	$O(n)$

Queue Runtime Analysis

	Linked List	Array
Creation	$O(1)$	$O(n)$
Enqueue	$O(1)$	$O(1)$
Dequeue	$O(1)$	$O(1)$
Peek	$O(1)$	$O(1)$
Print Queue	$O(n)$	$O(n)$

Applications of Queue Data Structures

- Online waiting rooms
- Operating System task scheduling
- Web Server Request Handlers
- Network Communication
- CSCI 232 Algorithms

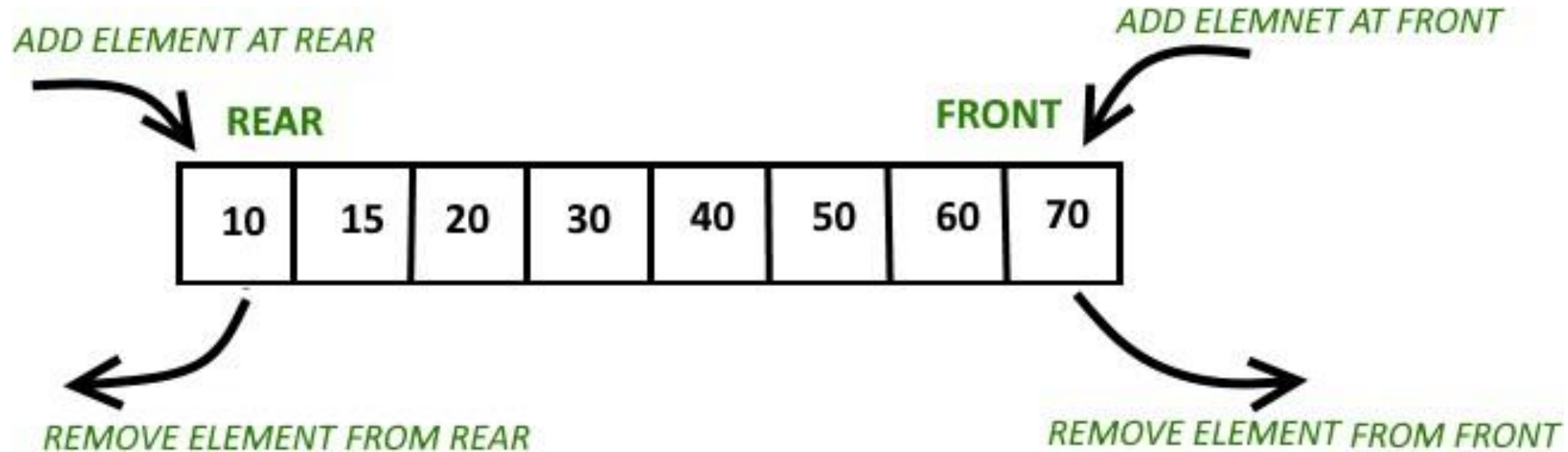
Stack Runtime Analysis

	w/ Array	w/ Linked List
Creation	$O(n)$	$O(1)$
Push()	$O(1)$	$O(1)$
Pop()	$O(1)$	$O(1)$
peek()	$O(1)$	$O(1)$
Print()	$O(n)$	$O(n)$

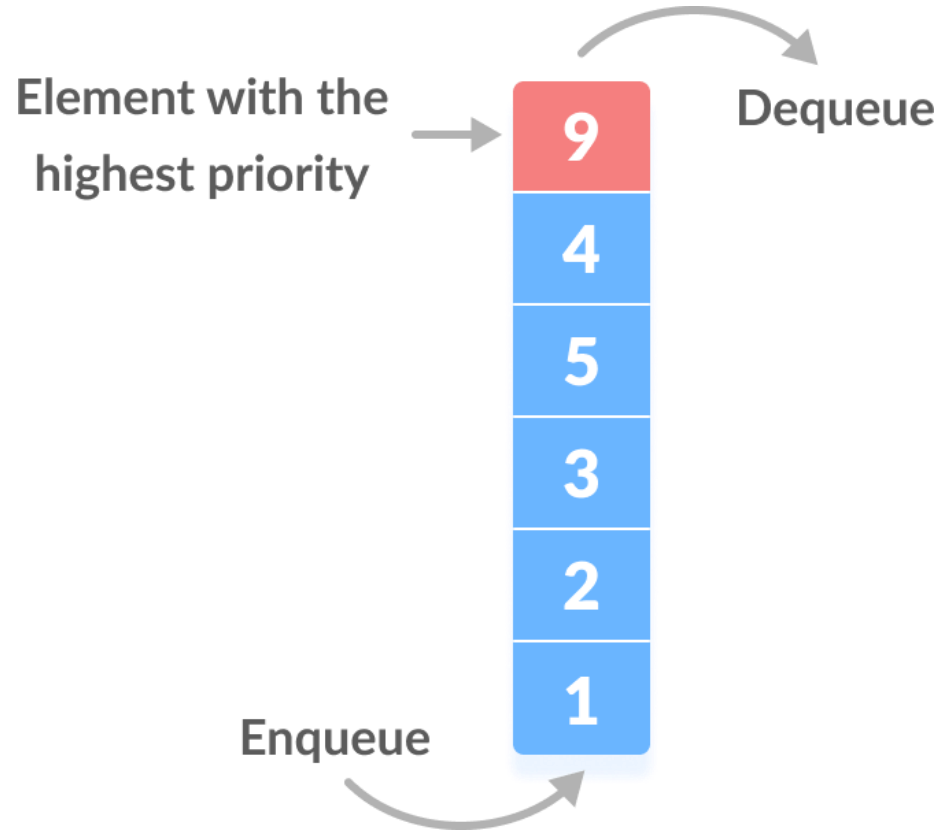
Applications of Stack Data Structures

- Tracking function calls in programming
- Web browser history
- Undo/Redo buttons
- Recursion/Backtracking
- CSCI 232 Algorithms

A double-ended queue, or a **deque** (deck) is a type of queue in which insertion and removal of elements can either be performed from the front or the rear



Most of the time, queues will operate in a FIFO fashion, however there may be times we want to dequeue the item with the **highest priority**



Priority queue in a data structure is an extension of a linear queue that possesses the following properties: Every element has a certain priority assigned to it

When we enqueue something, we might need to “shuffle” that item into the correct spot of the priority queue

Interface Comparable<T>

Implementing the comparable interface will allow us to compare objects to each other

compareTo(T o)

Compares this object with the specified object for order.

Should return:

- A *negative number* if this object is less than the other
- Zero if they are equal
- A *positive number* if this object is greater

In the real world, when you want to use a Queue, Stack, Deque, or a Priority Queue, you will likely import this data structure

```
import.java.util.Stack
```

```
import.java.util.Queue
```

java.util.Queue is an interface. We cannot create a Queue object.
Instead, we create an instance of an object *that implements* this interface

Some of the Classes that implement the Queue interface:

1. PriorityQueue (java.util.PriorityQueue)
2. Linked List (java.util.LinkedList)

(If you need a FIFO queue, Linked List is the way to go...)