Courses Practice Roadmap Pro

7 - Queues

Mark Lesson Complete View Code Prev Next

Suggested Problems

Status	Star	Problem	Difficulty	Video Solution	Code
☐	☆	Number Of Students Unable To Eat Lunch	Easy		
☐	☆	Implement Stack Using Queues	Easy	📹	C++

Queues

Queues are similar to stacks, except they follow what is called a FIFO approach (First in First Out). A real world example would be a line at the bank. The first person to come in the line is the first person to be served. An example from the software world would be print jobs. For example, if multiple people are trying to print documents, it will be handled on a first come first serve basis.

Implementation and Operations

The most common implementation of a queue is using a Linked List.

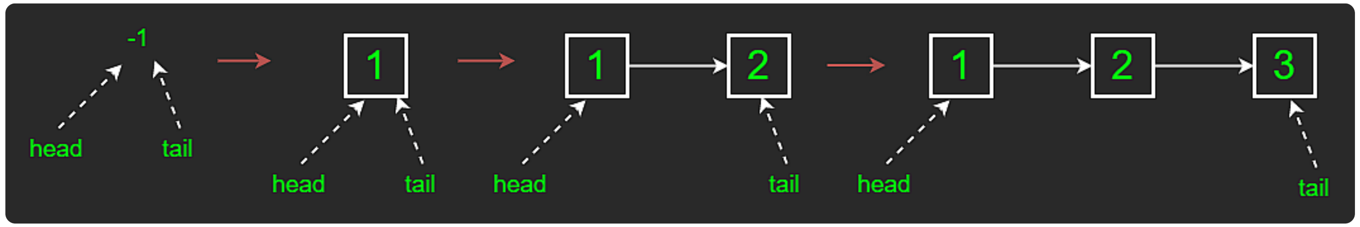
The two operations that queues support are `enqueue` and `dequeue` .

> *A queue is just an abstract interface, similar to a stack and can be implemented by multiple data structures, provided that they fulfill the contract of implementing enqueue and dequeue operations.*

Enqueue

The enqueue operation adds elements to the `tail` of the queue until the size of the queue is reached. Since adding to the end of the queue requires no shifting of the elements, this operation runs in $O(1)$. The following pseudocode and visual demonstrates this.

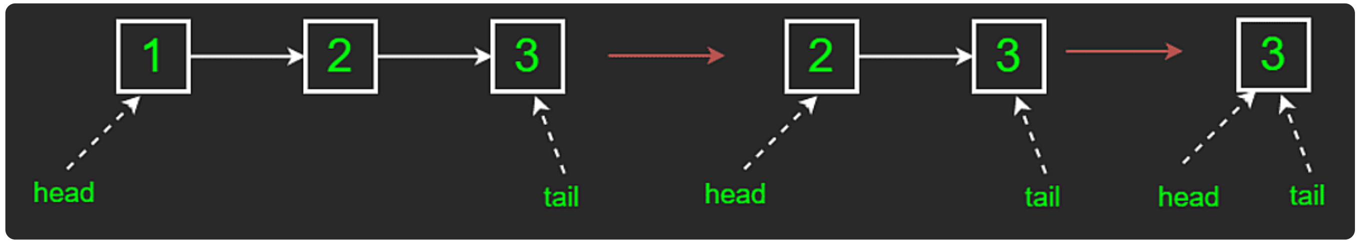
```
fn enqueue(val):
    newNode = ListNode(val)
    if queue is not empty:
        tail.next = newNode
        tail = tail.next
    else:
        // Queue is empty so head and tail both point to newNode
        head = newNode
        tail = newNode
```



Dequeue

The dequeue operation removes elements from the front of the queue and returns that element. The following pseudocode and the visual demonstrate this.

```
fn dequeue():
    if queue is empty:
        return -1
    // Otherwise remove from the head and update the head pointer
    else:
        val = head
        head = head.next
        if head is null:
            tail = null
        return val
```



Queues could also be implemented by using dynamic arrays, however, it gets a little bit trickier if you want to maintain efficiency of `enqueue` and `dequeue` opeartions. With the array implementation, `dequeue` would take $O(n)$ time due to shifting of the elements.

> *Similar to stacks, it is a good measure to check if the queue is empty before performing the dequeue opeartion.*

> *There is also a variation of the queue, a double-ended queue, called **deque** (pronounced "deck"). Deque allows you to add and remove elements from both the head and the tail.*

Closing Notes

Operation	Big-O Time Complexity
Enqueue	$O(1)$
Dequeue	$O(1)$

One of the most important use cases for the queue is when performing breadth-first search for trees and graphs, which we will see later in the course.