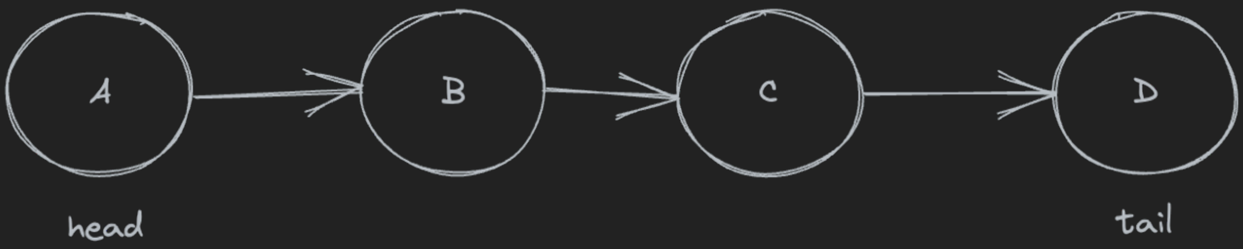
Operation	**Queue**
Enqueue	O(1)
Dequeue	O(1)
Peeking	O(1)
Contains	O(n)
Removal	O(n)
Is Empty	O(1)

Definition

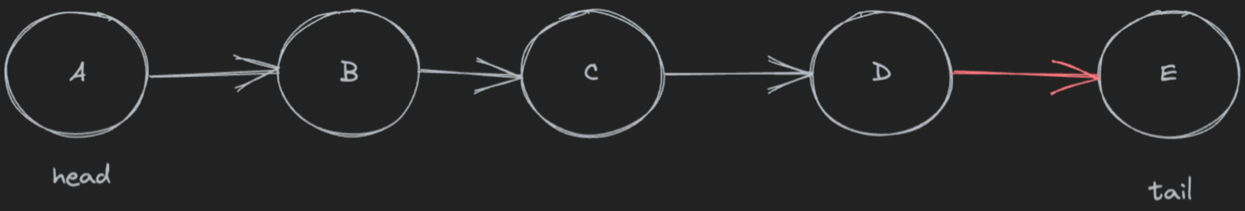
A queue is a linear data structure, having two primary operations, enqueue and dequeue. **FIFO** (First in first out)



Every queue has a front and a back—you can choose, allowing access to both ends of the queue at any moment.

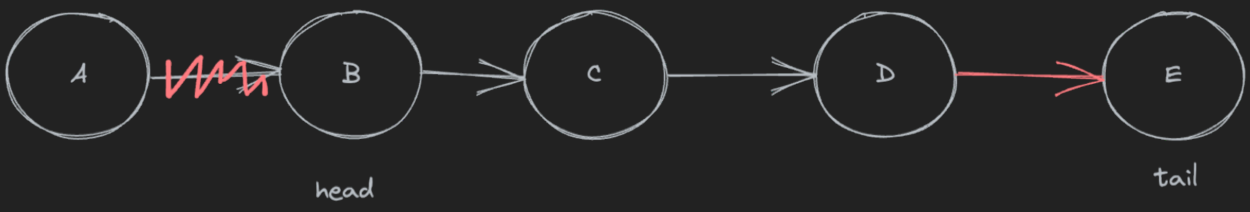
Enqueue (Adding or Offering)

When you add elements to the back



Dequeue (Removing or Polling)

When you remove elements to the front



Peek

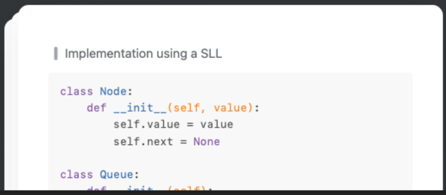
Check what the first value of the Queue is, without removing it

Use Cases

- Modeling any sort of line, such as a movie theatre
- Can be used to efficiently keep track of the x most recent elements added
- Web server request management where you want first come first serve
- Breathe First Search (BFS) graph traversal

Implementation

Implementation using a SLL



Implementation using a SLL

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None

class Queue:
```

↑ Queue

Implementation

Implementation using a SLL

```python
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None

class Queue:
    def __init__(self):
        self.length = 0
        self.head = None
        self.tail = None

    def enqueue(self, item):
        node = Node(item)
        self.length += 1

        if self.length == 0:
            self.head = self.tail = node
            return

        self.tail.next = node
        self.tail = node

    def dequeue(self):
        if self.length == 0:
            return None

        self.length -= 1
        head = self.head
        self.head = self.head.next

        # dealloc
        head.next = None

        return head.value

    def peek(self):
        return self.head.value if self.head is not None else None
```