Data Structures and Algorithms

Khazhak Galstyan

Session 5:
Basic Data Structures

What we will cover today

- Singly / Doubly Linked Lists
- Stacks
- Queues
- Deques

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List								
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)							
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)						
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)					
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)				
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)			
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)		
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

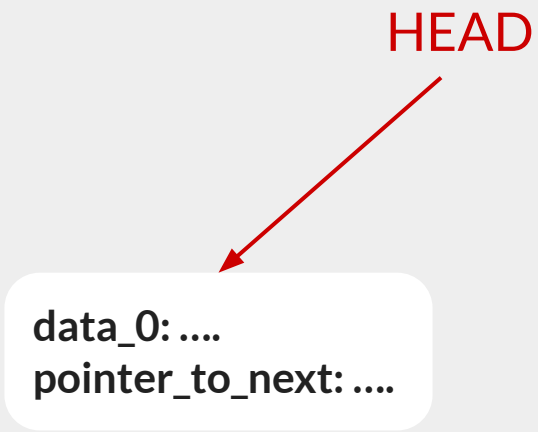
	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

Linked Lists

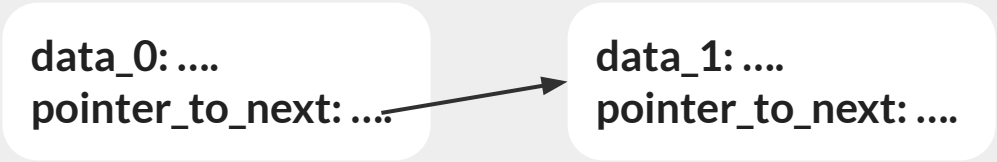
Linked List

data_0: ....
pointer_to_next: ....

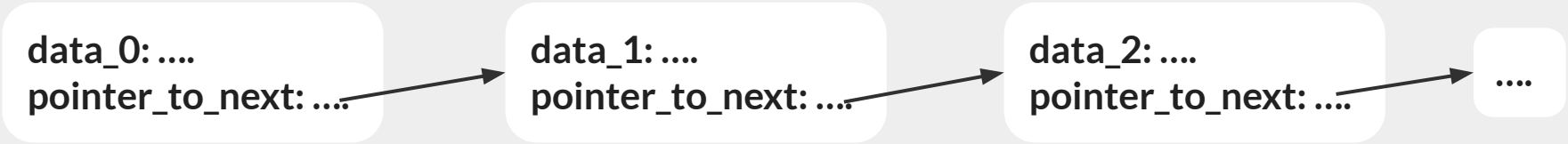
Linked List



Linked List



Linked List


data_0: ....
pointer_to_next: ....
data_1: ....
pointer_to_next: ....
data_2: ....
pointer_to_next: ....
....

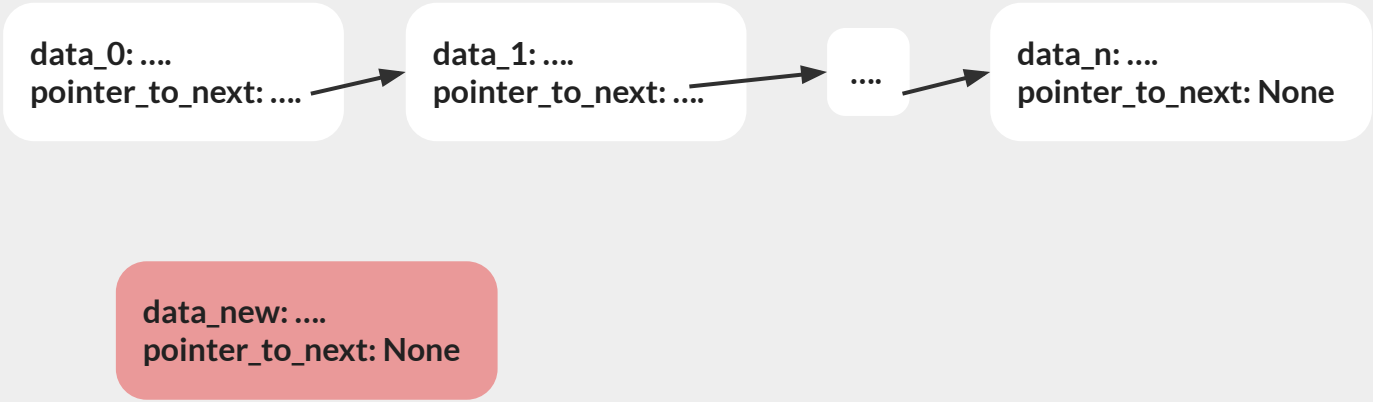
Linked List: Implementation

```
class LinkedList {
    Node head;

    class Node {
        int data;
        Node next;

        Node(int d) { data = d; }
    }
}
```

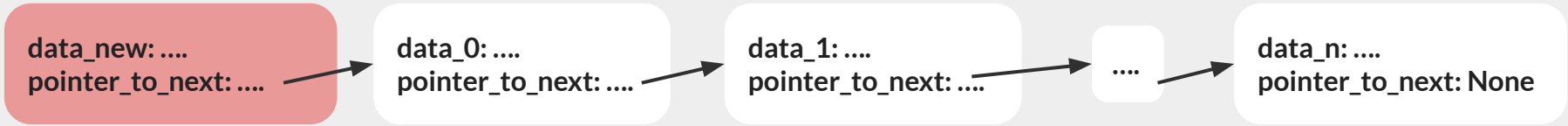
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

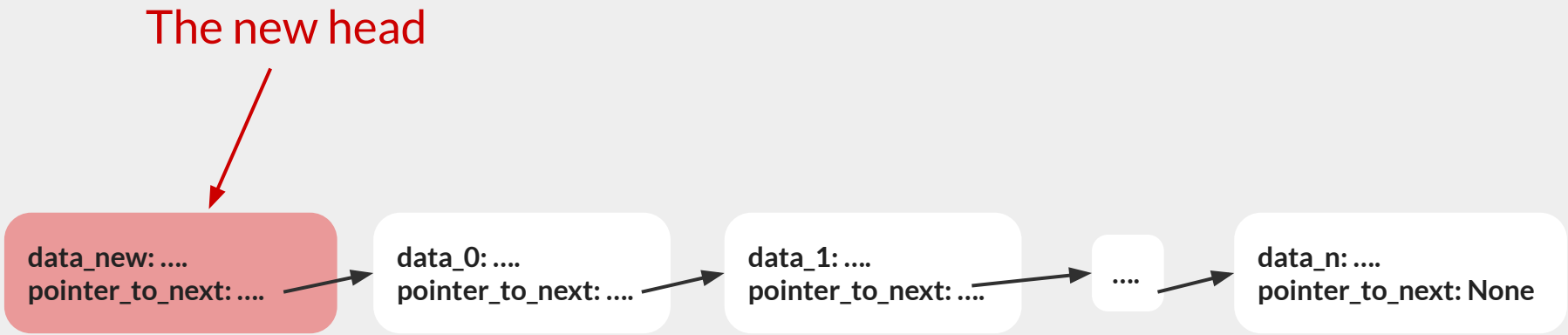
Linked List: Append



Linked List: Append



Linked List: Append



Linked List: Append Implementation

```
class Node:
   def init(self, data=None):
      self.data = data
      self.next_pointer = None

class LinkedList:
   def init(self):
      self.head = None

   def Append(self, newdata):
      new_node = Node(newdata)
      new_node.next_pointer = self.head
      self.head = new_node
```

```
class LinkedList {
    Node head;

    class Node {
        int data;
        Node next;

        Node(int d) { data = d; }
    }
}
```

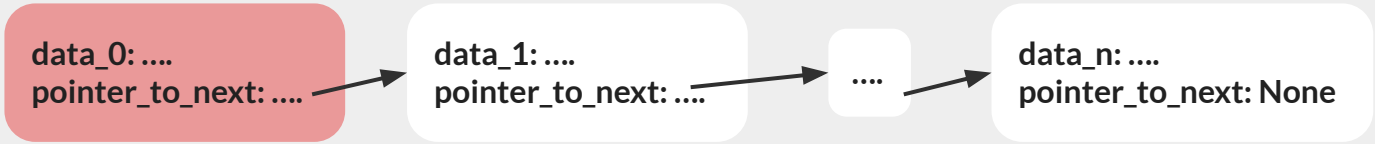
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single								
Linked List Double								
Stack								
Queue								
Deque								

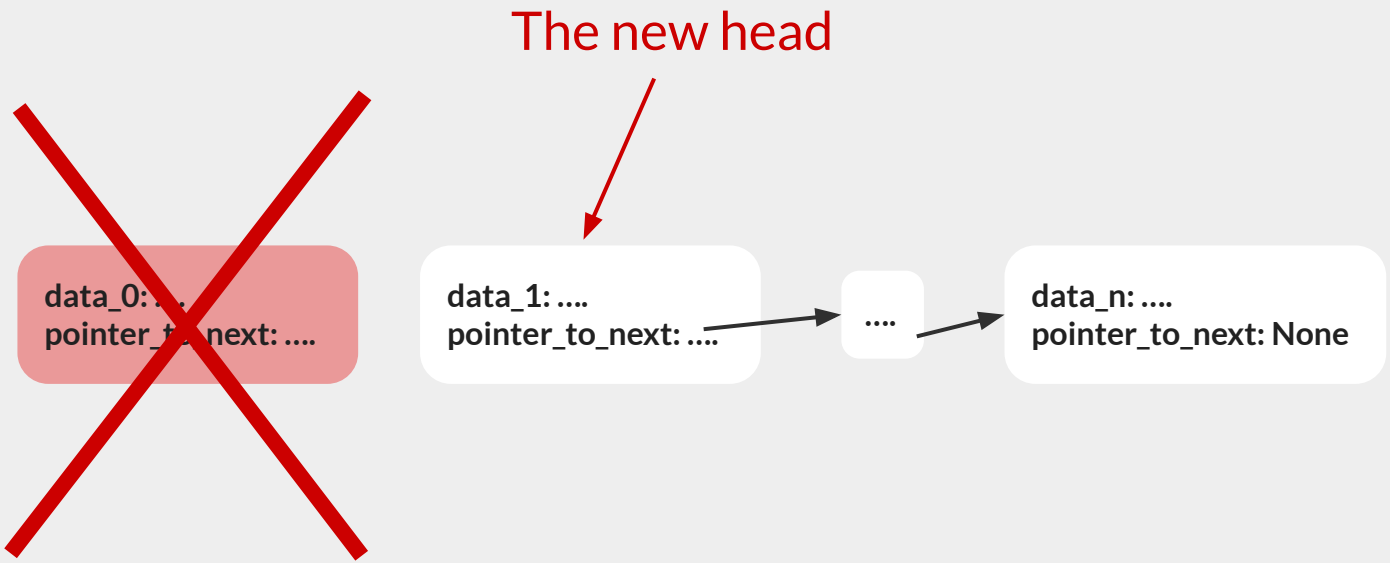
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)							
Linked List Double								
Stack								
Queue								
Deque								

Linked List: Pop



Linked List: Pop



Linked List: Pop Implementation

```
class Node:
   def init(self, data=None):
      self.data = data
      self.next_pointer = None

class LinkedList:
   def init(self):
      self.head = None

   def PopHead(self):
      if self.head is not None:
          head = self.head
          self.head = self.head.next_pointer
          return head
      else:
          raise RuntimeError('Trying to pop from empty list')
```

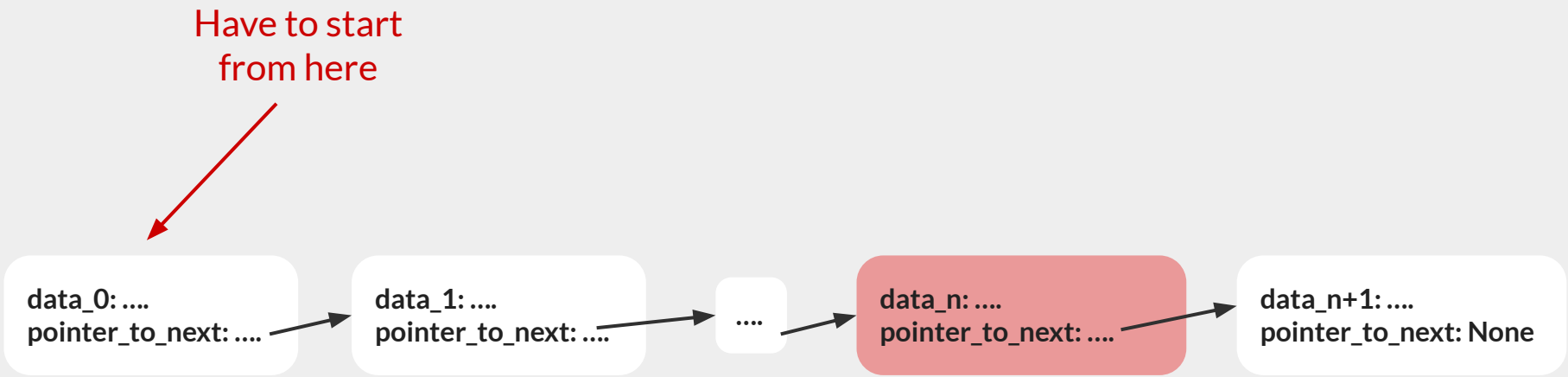
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)							
Linked List Double								
Stack								
Queue								
Deque								

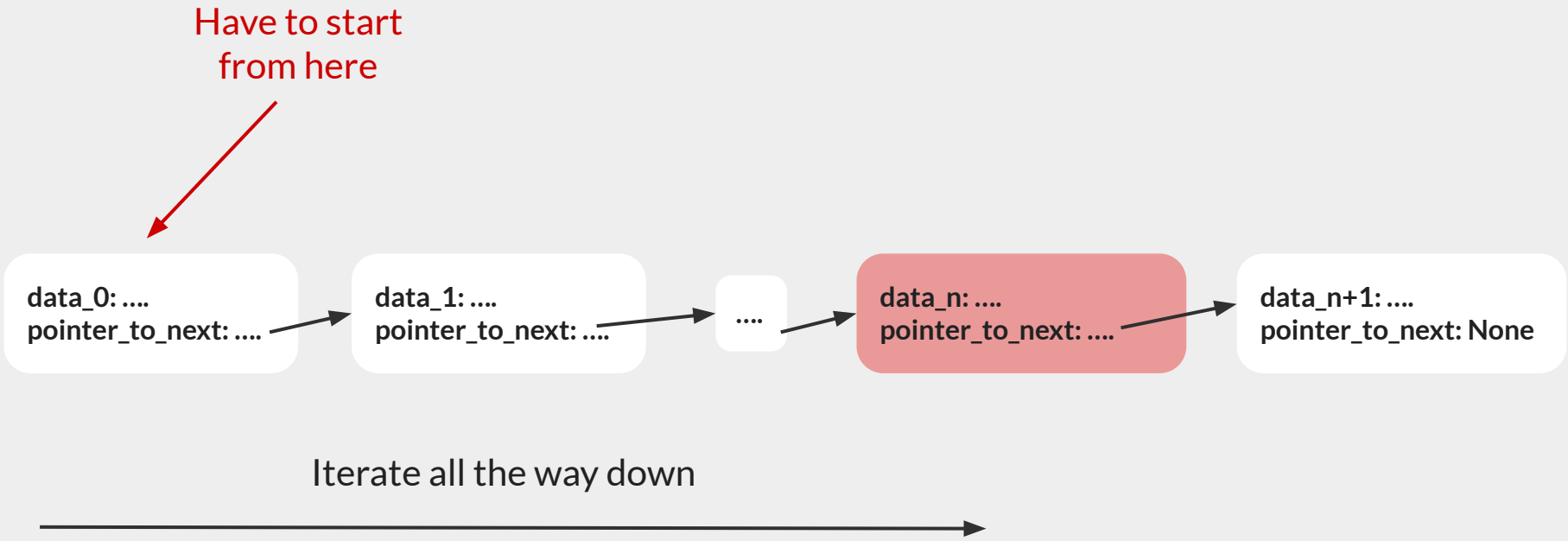
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)		O(1)					
Linked List Double								
Stack								
Queue								
Deque								

Linked List: Get N-th Item



Linked List: Get N-th Item



Linked List: Get Implementation

```
class LinkedList:
   def init(self):
      self.head = None

   def Get(self, n):
      current_node = self.head

      for i in range(0, n):
          if current_node is None:
               raise Exception(....)
          current_node = current_node.next_pointer

      return current_node.data
```

Linked List: Set Implementation

```
class LinkedList:
   def init(self):
      self.head = None

   def Set(self, n, data):
      current_node = self.head

      for i in range(0, n):
          if current_node is None:
               raise Exception(....)
          current_node = current_node.next_pointer

      current_node.data = data
```

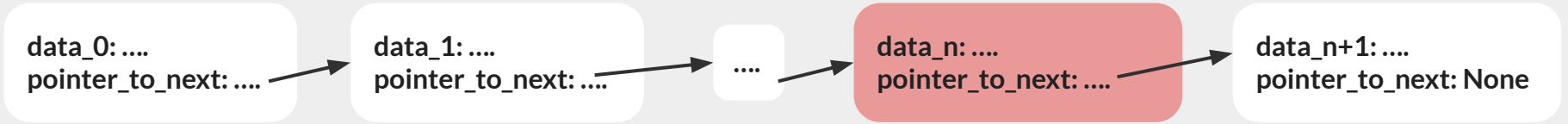
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)		O(1)					
Linked List Double								
Stack								
Queue								
Deque								

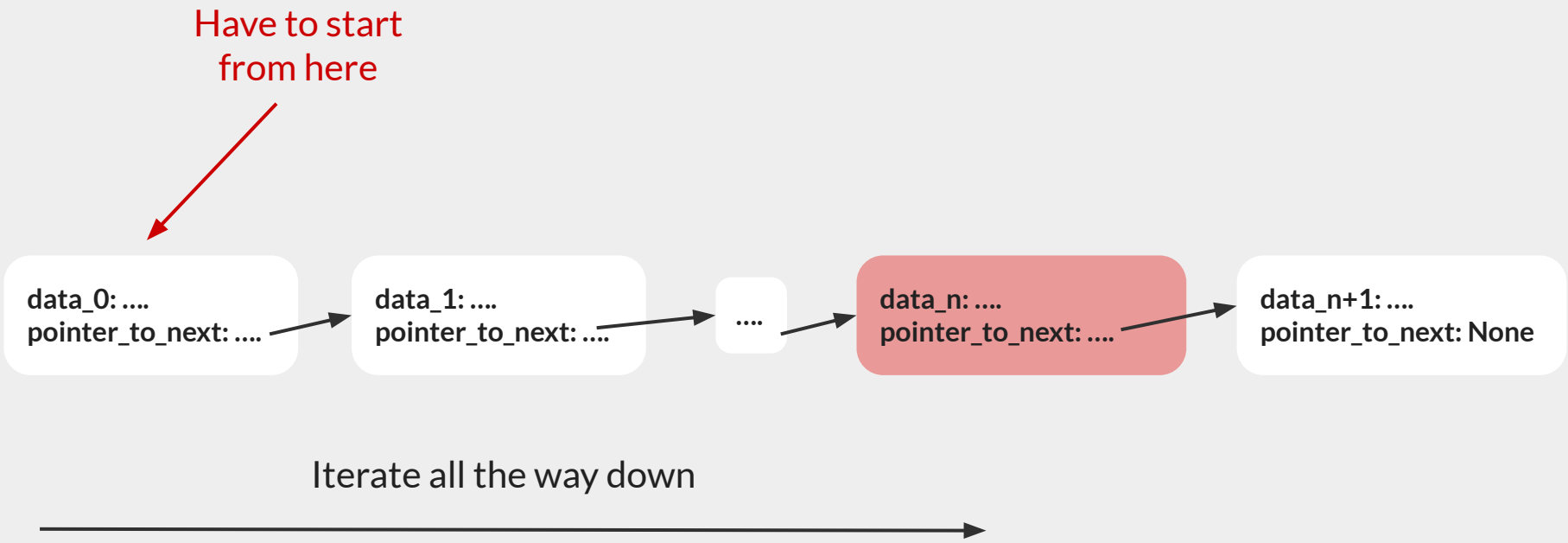
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)		O(1)				O(n)	O(n)
Linked List Double								
Stack								
Queue								
Deque								

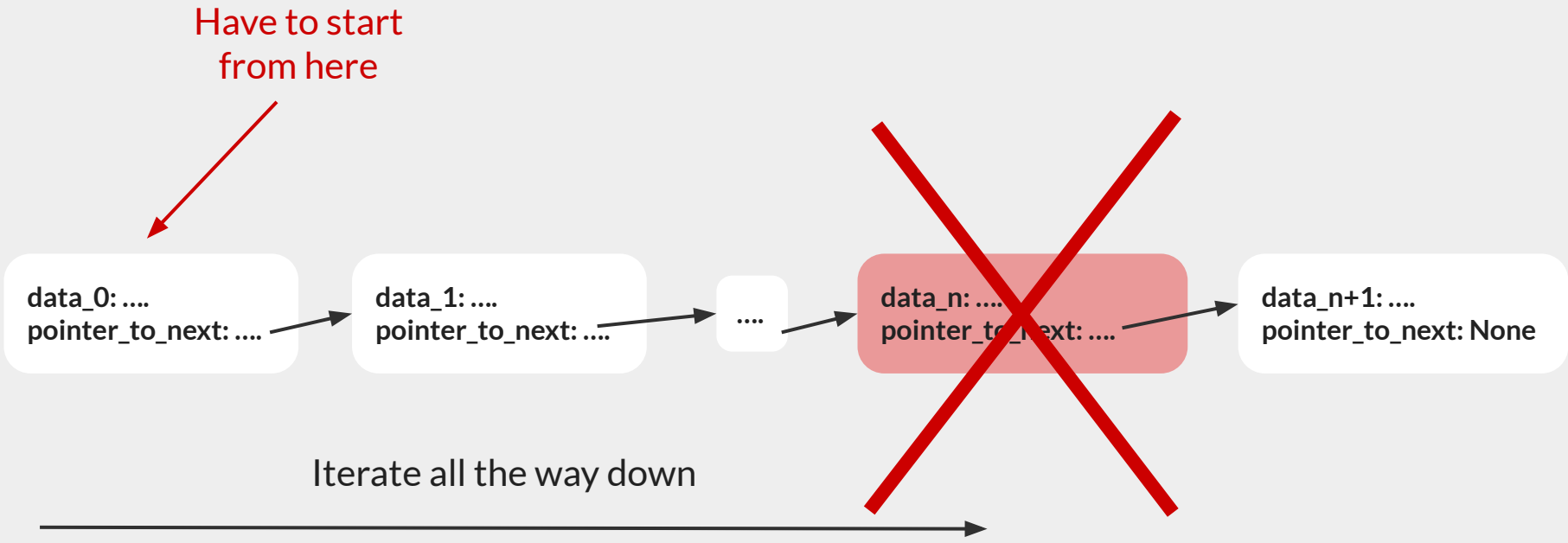
Linked List: Remove N-th Item



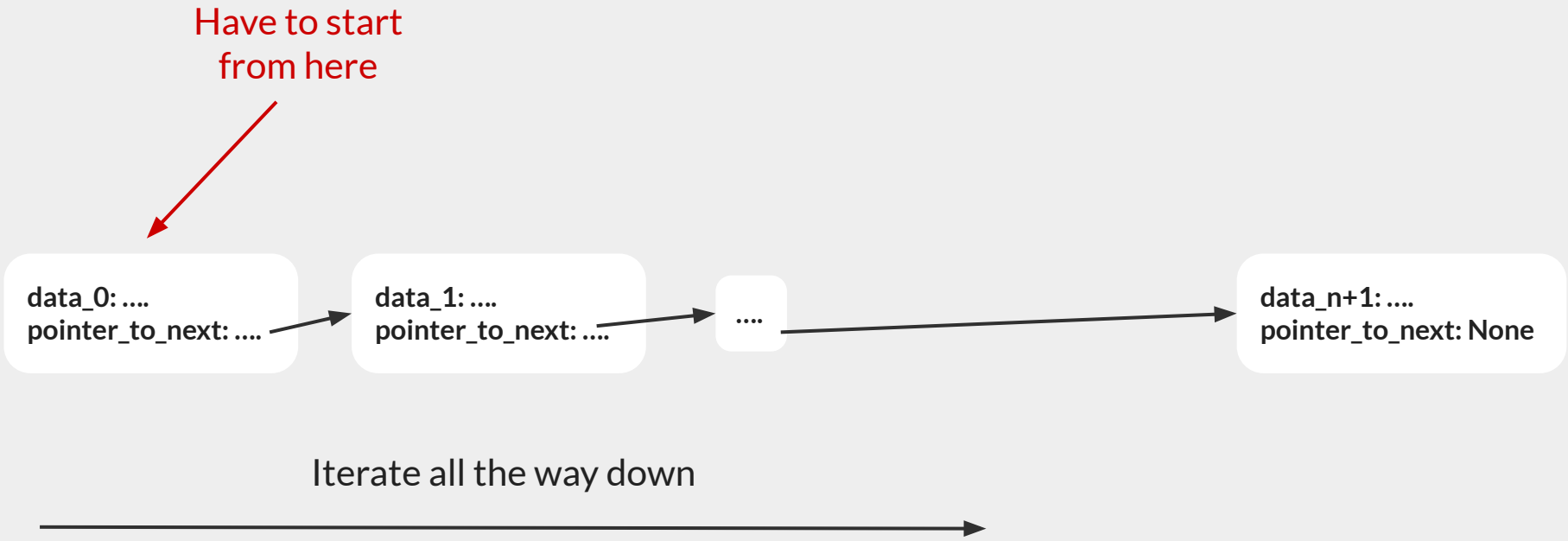
Linked List: Remove N-th Item



Linked List: Remove N-th Item



Linked List: Remove N-th Item



Linked List: Remove Implementation

```
class LinkedList:
   def init(self):
      self.head = None

   def Remove(self, n):
      current_node = self.head

      for i in range(0, n-1):
          if current_node is None:
               raise Exception(....)
          current_node = current_node.next_pointer

      to_pop = current_node.next_pointer
      current_node.next_pointer = to_pop.next_pointer
      return to_pop
```

Linked List: Insert Implementation

```
class LinkedList:
   def init(self):
      self.head = None

   def Insert(self, n, node):
      current_node = self.head

      for i in range(0, n-1):
          if current_node is None:
               raise Exception(....)
          current_node = current_node.next_pointer

      node.next_pointer = current_node.next_pointer
      current_node.next_pointer = node
```

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)		O(1)				O(n)	O(n)
Linked List Double								
Stack								
Queue								
Deque								

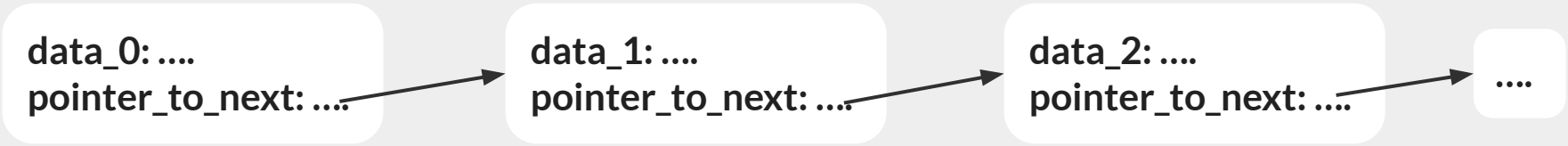
Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)		O(1)		O(n)	O(n)	O(n)	O(n)
Linked List Double								
Stack								
Queue								
Deque								

Worst-Case Complexities

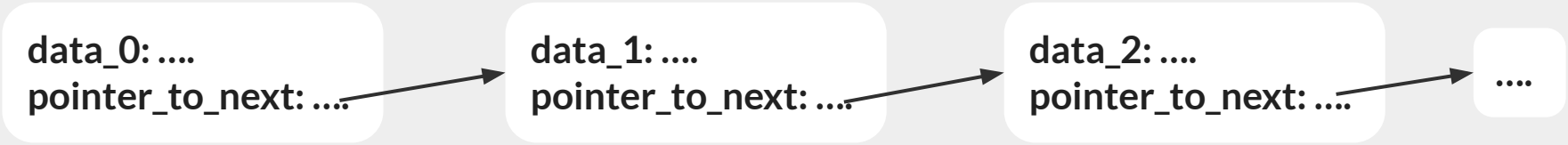
	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double								
Stack								
Queue								
Deque								

Singly Linked List

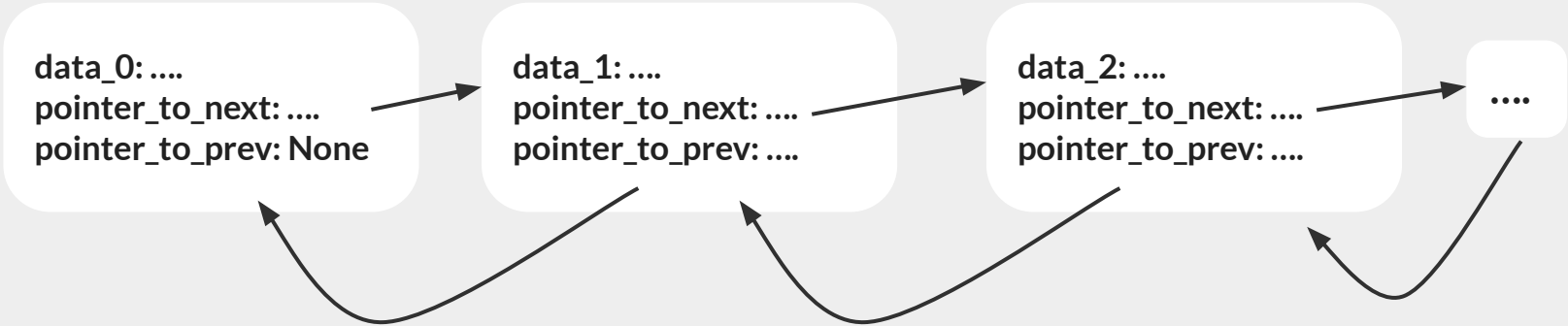


Doubly Linked List

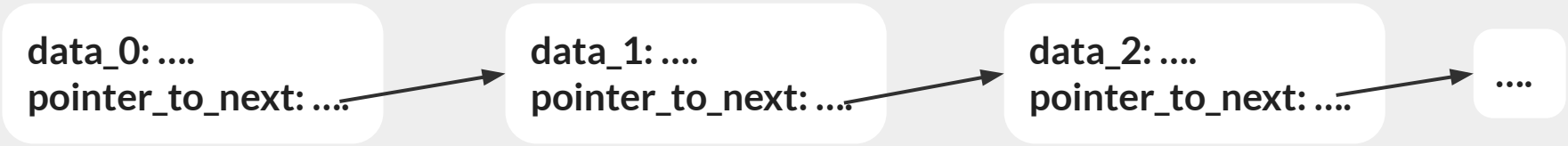
Singly Linked List



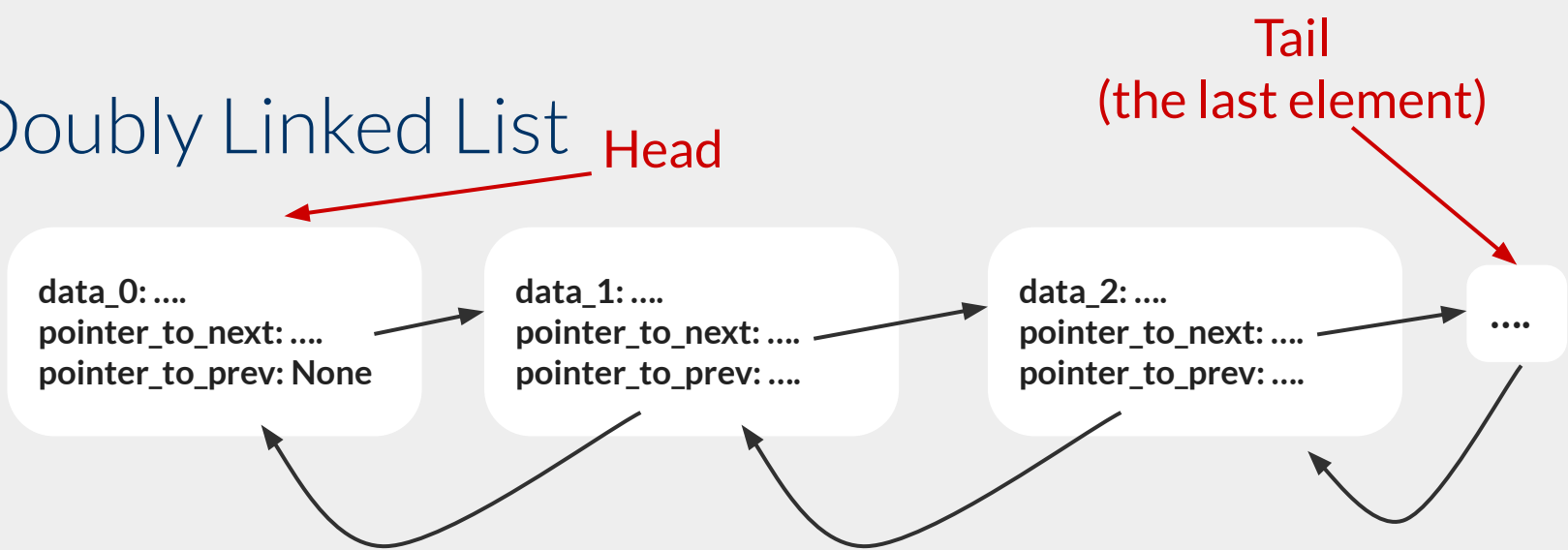
Doubly Linked List



Singly Linked List



Doubly Linked List



Worst-Case Complexities

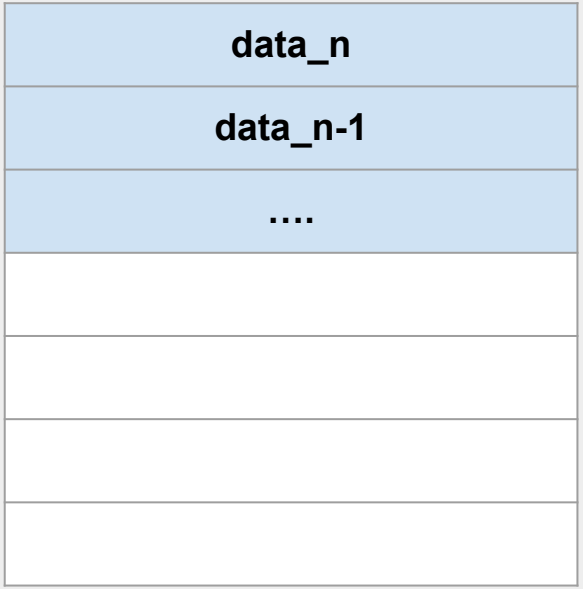
	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack								
Queue								
Deque								

Linked List: Real-Life Use Cases

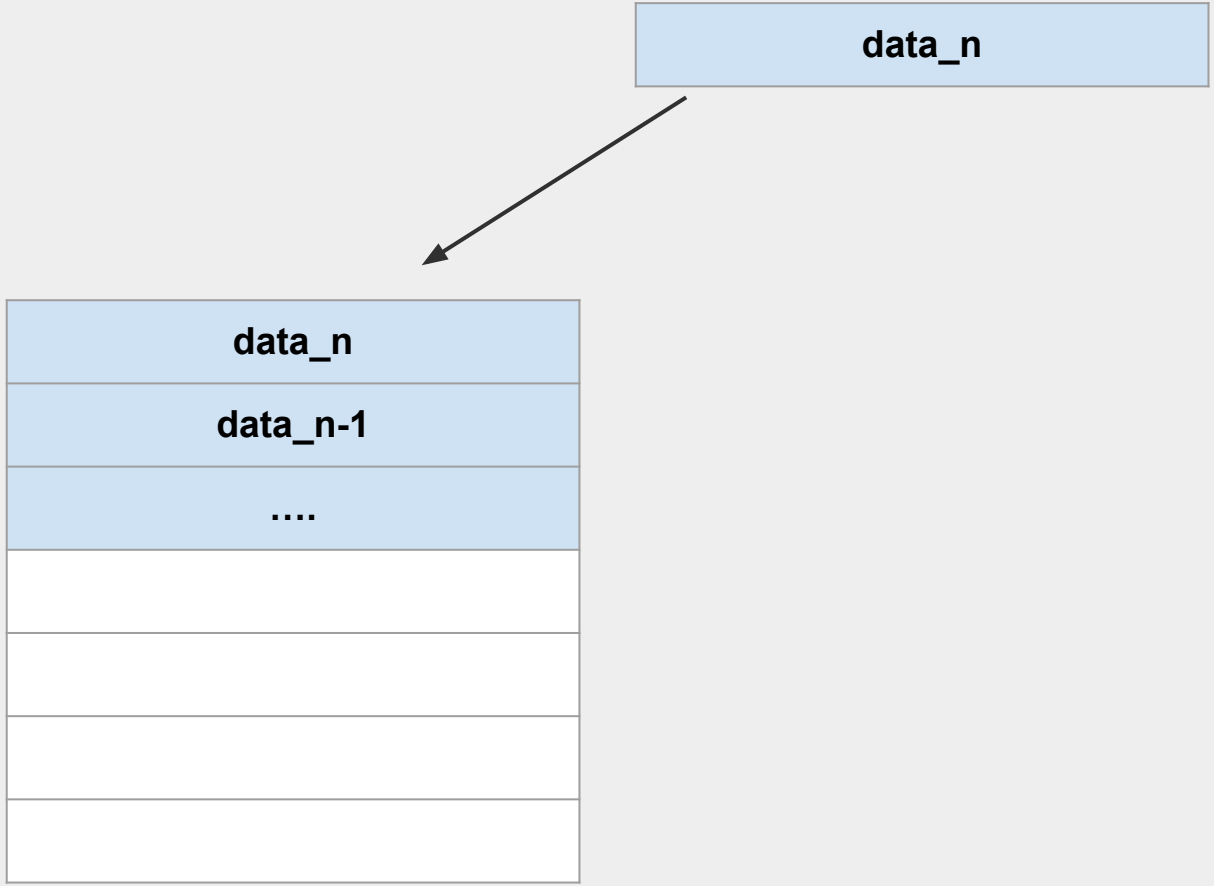
- Low-level memory management
- Databases / File Systems
- Implementing Stacks / Queues
- ….

Stacks: Last In First Out (LIFO)

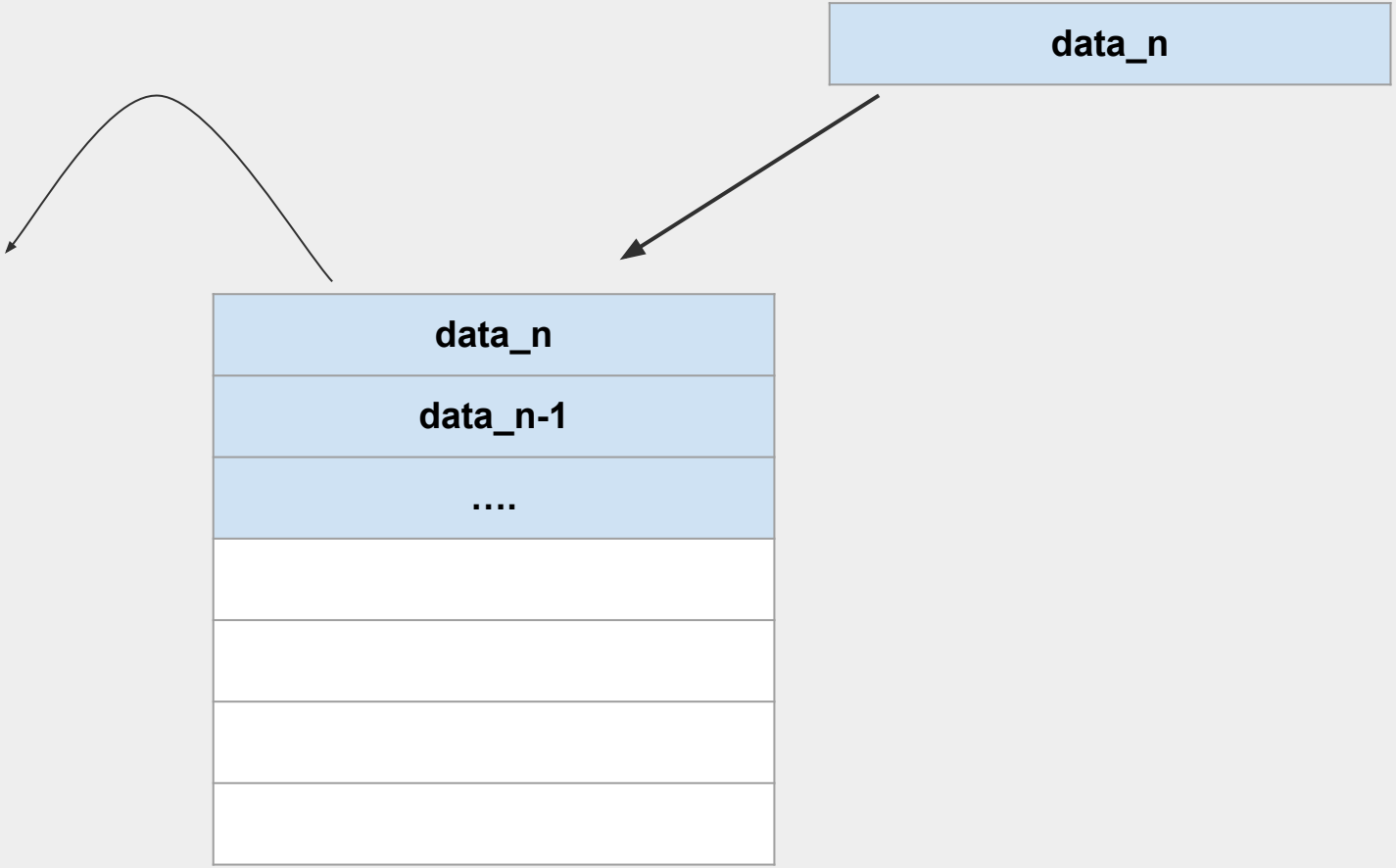
Stack



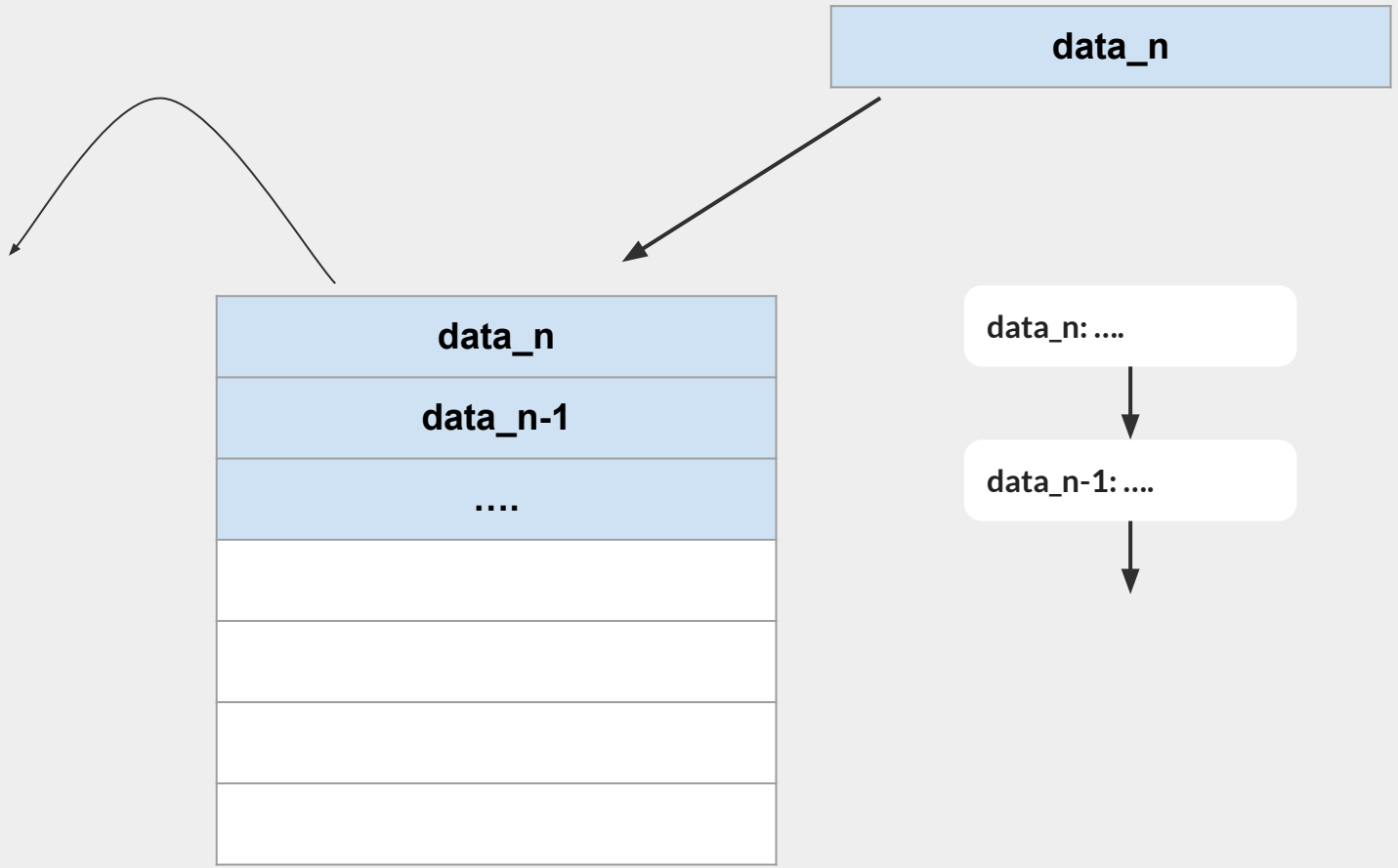
Stack



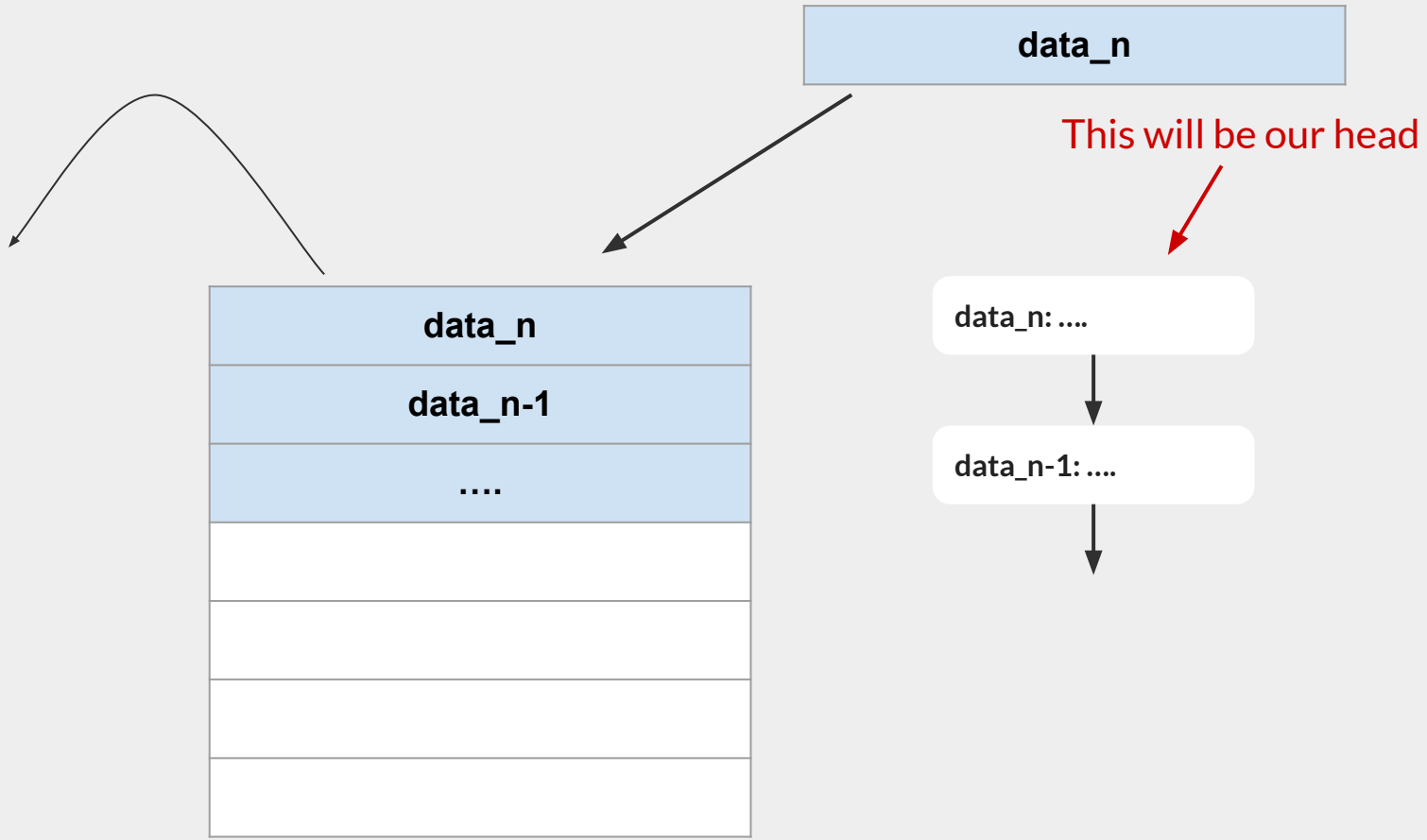
Stack



Stack



Stack



Stack: Implementation

```
class Stack:
      def init(self):
            self.linked_list = LinkedList()

      def push(item):
            self.linked_list.Append(item)

      def pop():
            return self.linked_list.PopHead()

      def peek():
            return self.linked_list.head.data
```

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack								
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)							
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)	N/A						
Queue								
Deque								

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)	N/A	O(1)					
Queue								
Deque								

Worst-Case Complexities

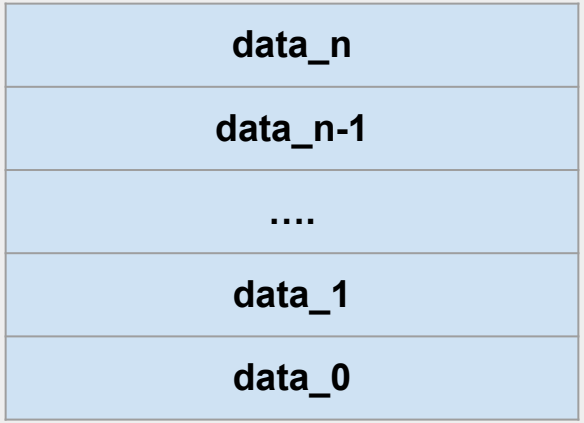
	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)	N/A	O(1)	N/A	N/A	N/A	N/A	N/A
Queue								
Deque								

Stack: Real-Life Use Cases

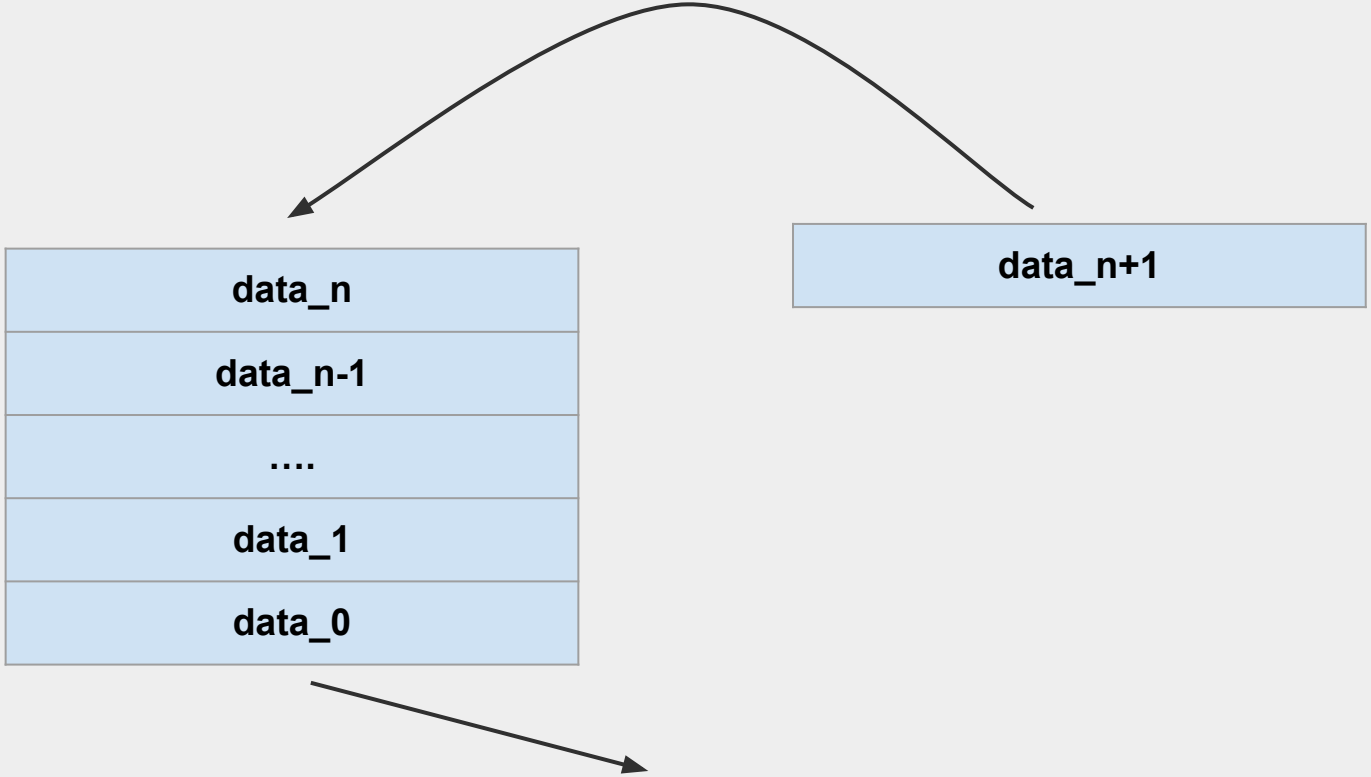
- Programming Language Interpreters / Compilers
- Syntax Parsing / Expression Evaluation
- Virtual Machines
- Anywhere you do undo / redo
- ….

Queue: First In First Out (FIFO)

Queue



Queue



Queue: Implementation

```
class Queue:
    def init(self):
        self.head = None
        self.tail = None

    def push(value):
        item = Item(value)
        if self.head is None:
            self.head = item
            self.tail = item
        else:
            self.tail.next = item
    def pop():
        item = self.head
        self.head = item.next
        return item.value

    def peek():
        return self.head.value
```

```
class Item:
    def init(self, value):
        self.value = value
        self.next = None
```

Queue: Implementation using DLL

```
class Queue:
      def init(self):
            self.linked_list = DoublyLinkedList()

      def push(value):
            self.linked_list.AppendLeft(value)

      def pop():
            return self.linked_list.PopRight()

      def peek():
            return self.linked_list.head.value
```

Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)	N/A	O(1)	N/A	N/A	N/A	N/A	N/A
Queue								
Deque								

Worst-Case Complexities

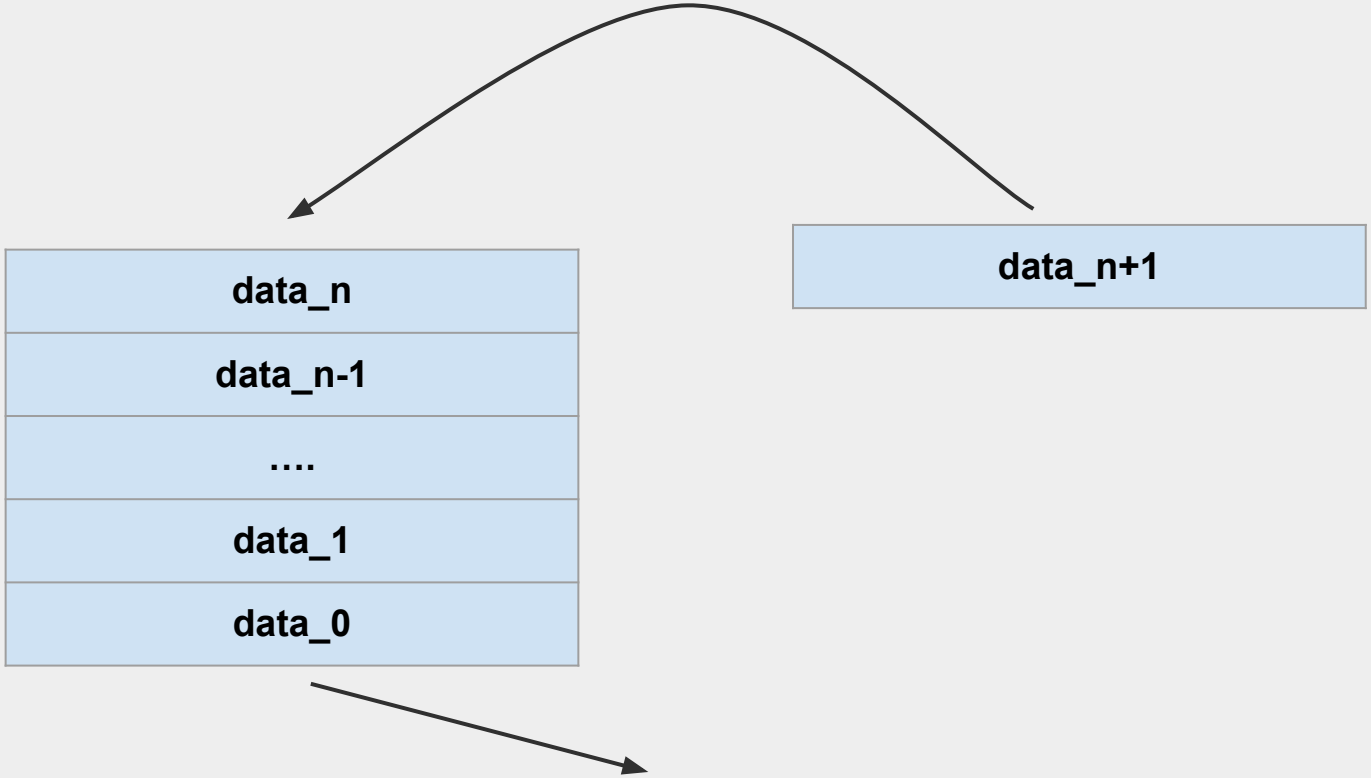
	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)	N/A	O(1)	N/A	N/A	N/A	N/A	N/A
Queue	O(1)	N/A	N/A	O(1)	N/A	N/A	N/A	N/A
Deque								

Queue: Real-Life Use Cases

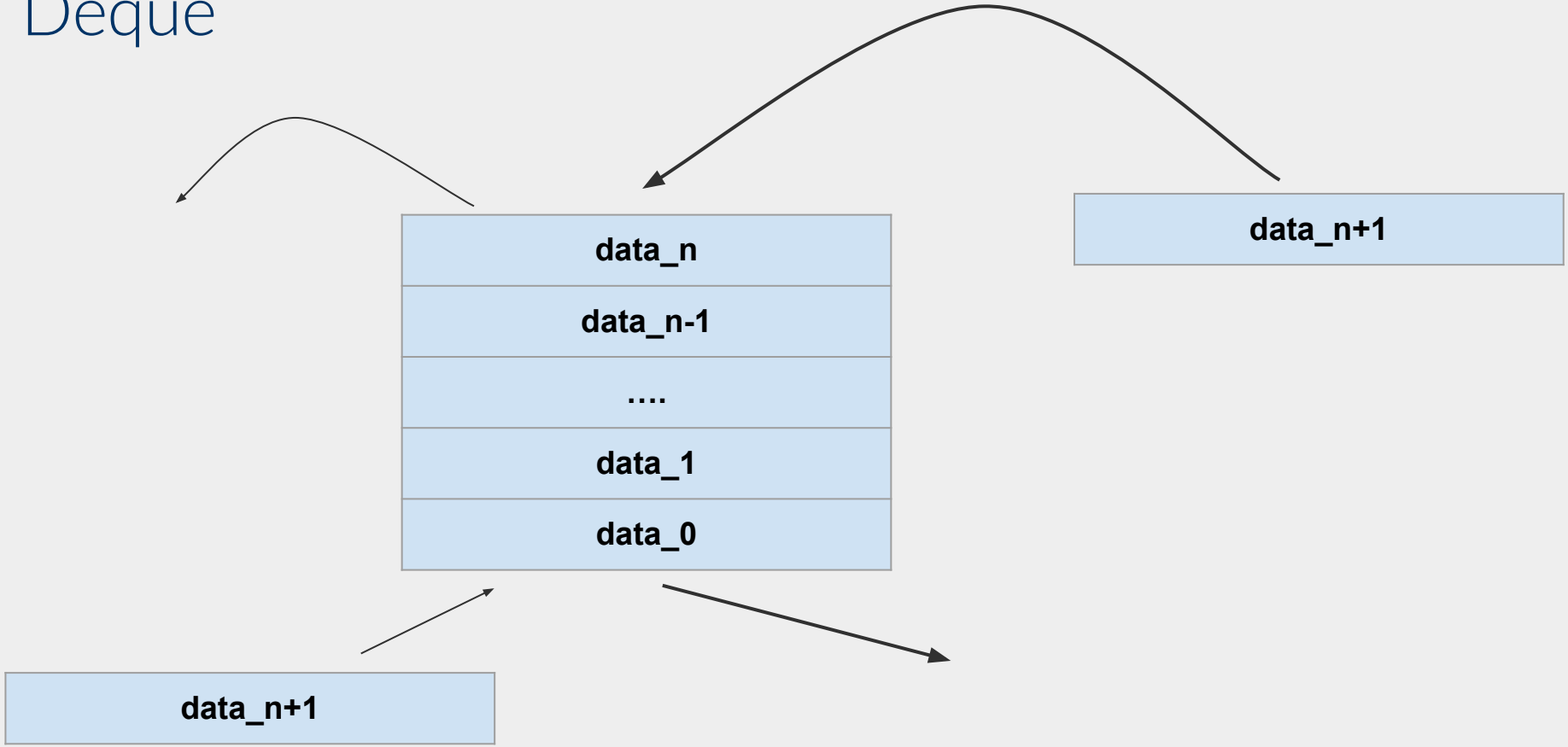
- Simulations
- Operating Systems
- Networks
- ….

Deque

Deque



Deque



Worst-Case Complexities

	Append Left	Append Right	Pop Left	Pop Right	Delete k-th	Insert k-th	Get k-th	Set k-th
List	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(1)	O(1)
Linked List Single	O(1)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
Linked List Double	O(1)	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)	N/A	O(1)	N/A	N/A	N/A	N/A	N/A
Queue	O(1)	N/A	N/A	O(1)	N/A	N/A	N/A	N/A
Deque	O(1)	O(1)	O(1)	O(1)	N/A	N/A	N/A	N/A

Thanks!