App Academy

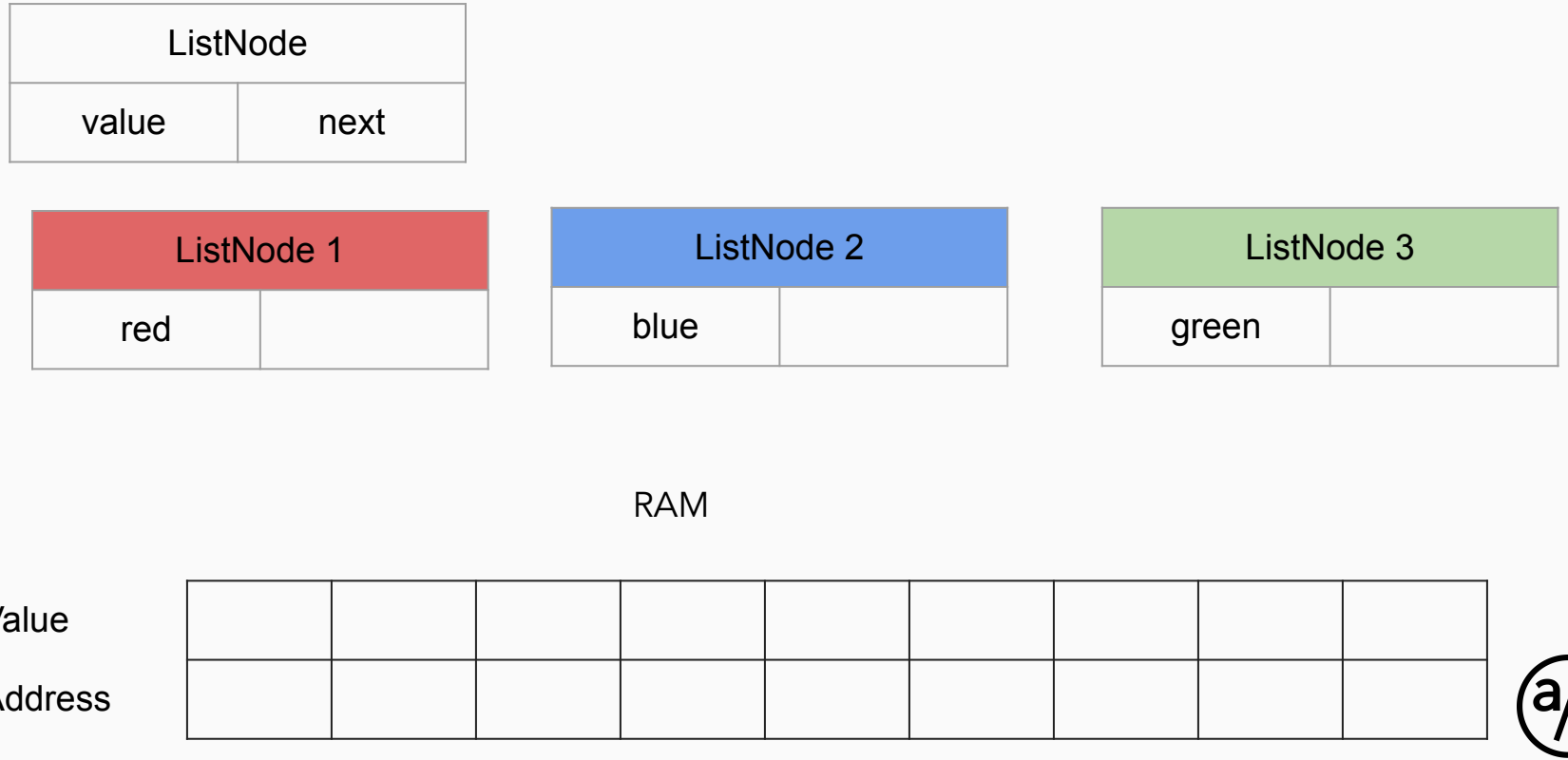
Linked Lists

Linked Lists

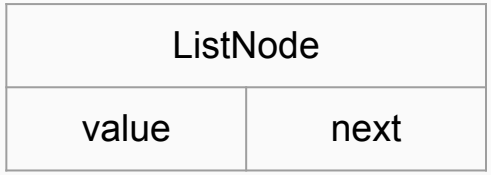
- Linked Lists are made up of “list nodes”
- Each of these nodes encapsulate at minimum 2 components
  - Value
  - Pointers to other nodes
    - Next pointer
    - Prev pointer (for doubly linked list)
- We can represent these nodes using objects with key-value pairs
  - Example:

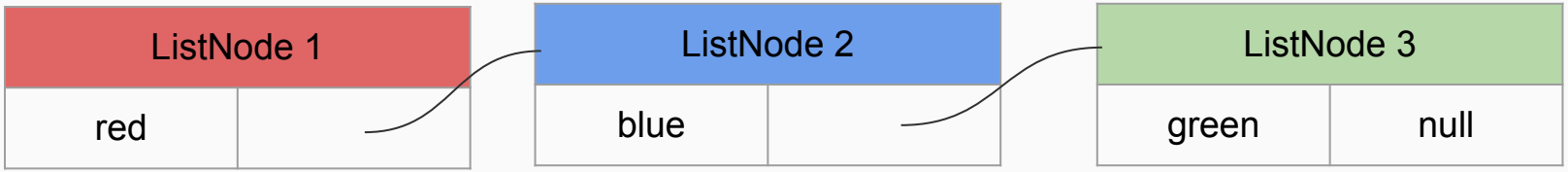
```
{
    value: 0,
    next: NODE,
    prev: NODE
}
```

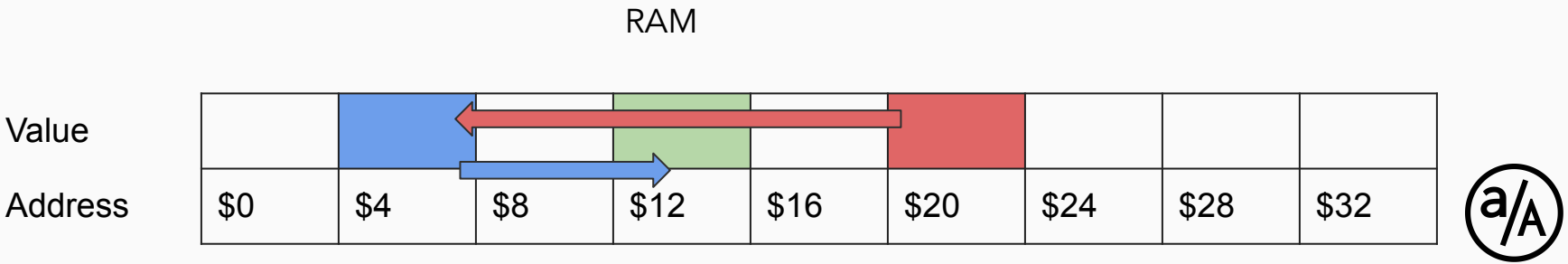
Singly Linked Lists



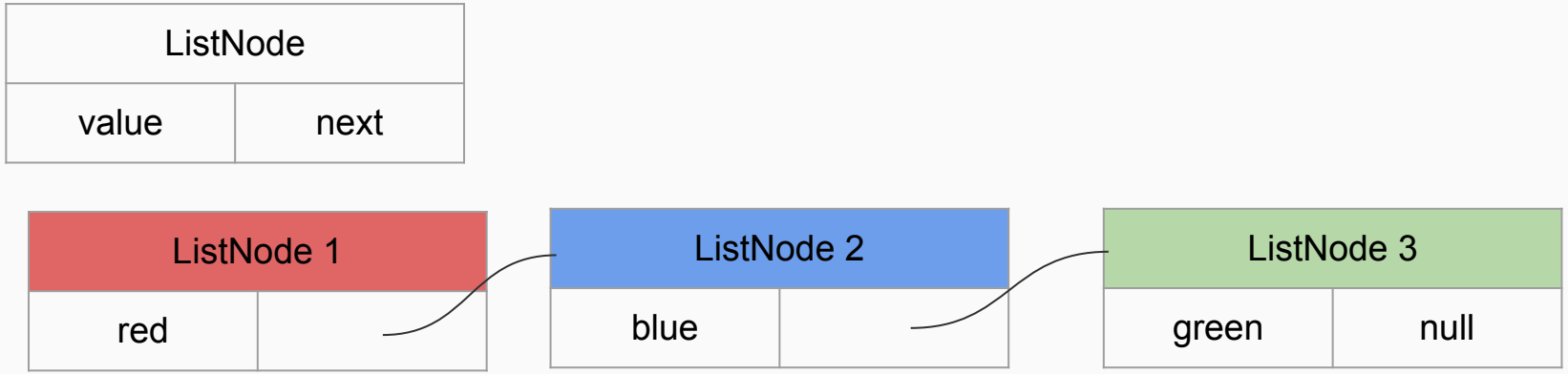
Singly Linked Lists





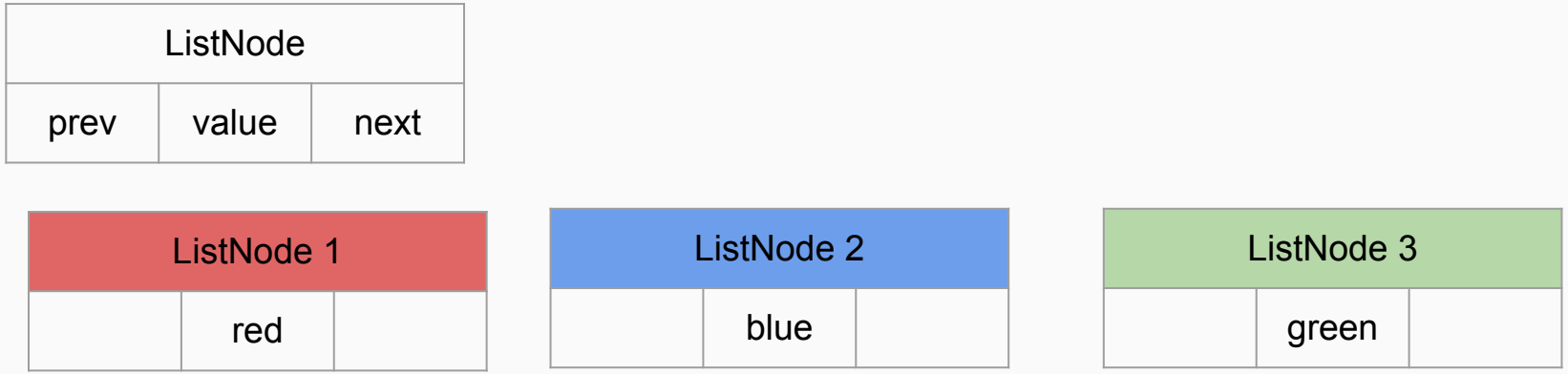


Chaining .next

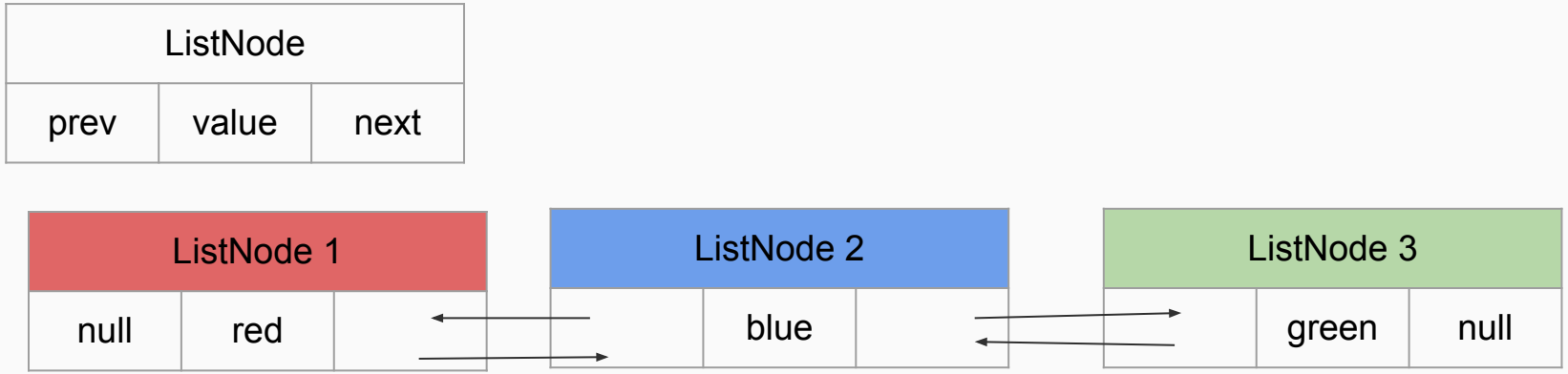


- From ListNode1, we can access ListNode2 by using ListNode1.next
- We can access ListNode3 from both ListNode1 or ListNode2 with the following:
  - ListNode1.next.next
  - ListNode2.next

Doubly Linked Lists



Doubly Linked Lists



How do we traverse a linked list?

Iterative:

```
def traverse(head):
    current = head
    while (current) :
        current = current.next;
    return None
```

Recursive:

```
def traverse(head):
    if not head:
        return None
    return traverse(head.next)
```

Linked List Runtimes

Operations	Big-O Time
Read/ Write ith element	
Insert / Remove End	
Insert Middle or Beginning***	
Remove Middle or Beginning	

Comparing with Array Runtimes

Arrays

Operations	Big-O Time
Read/ Write ith element	O(1)
Insert / Remove End	O(1)
Insert Middle or Beginning	O(n)
Remove Middle or Beginning	O(n)

Linked Lists

Operations	Big-O Time
Read/ Write ith element	O(n)
Insert / Remove End	O(1)
Insert Middle or Beginning***	O(1)
Remove Middle or Beginning	O(1)

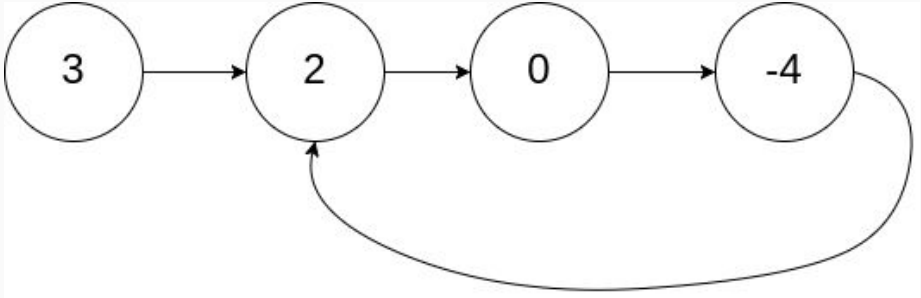
***Note that the act of inserting or removing a node from the middle of a linked list by itself is O(1), but this assumes you already found the location for insertion / removal.

Demos

- Middle of the Linked List
- Reverse Linked List

Cycle Detection

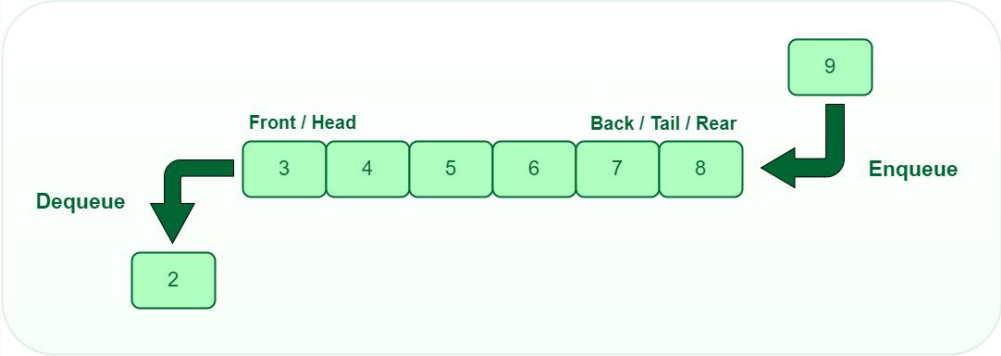
- Fast and slow pointers can also be used with Linked Lists to detect loops within the list.
- Demo:
  - Linked List Cycle



Queues

- FIFO (first in, first out)
- Because we can add/remove nodes at the beginning of a Linked List in O(1) time, it is actually more efficient to create a queue this way versus an array.
- In python, we can use deque from collections library

Operations	Big-O Time
Enqueue	O(1)
Dequeue	O(1)



***There aren't a lot of problems where a queue shines on its own, which is why this slide is lackluster. Don't worry though, in trees & graphs, you'll see how queues are used to implement BFS - an extremely powerful algorithm.

Questions?

Let's practice!

- Review - try doing these both iteratively and recursively!
  - Merge Two Sorted Lists
  - Remove Nth Node From End of List
- Bonus
  - Design Browser History
  - LRU Cache