iOS Dev Accelerator Week 2 Day 5

- Linked Lists
- Week Wrap up

Linked Lists

A Linked List is…

A group of nodes linked together in a sequence.

Can think of it as a line of railroads cars

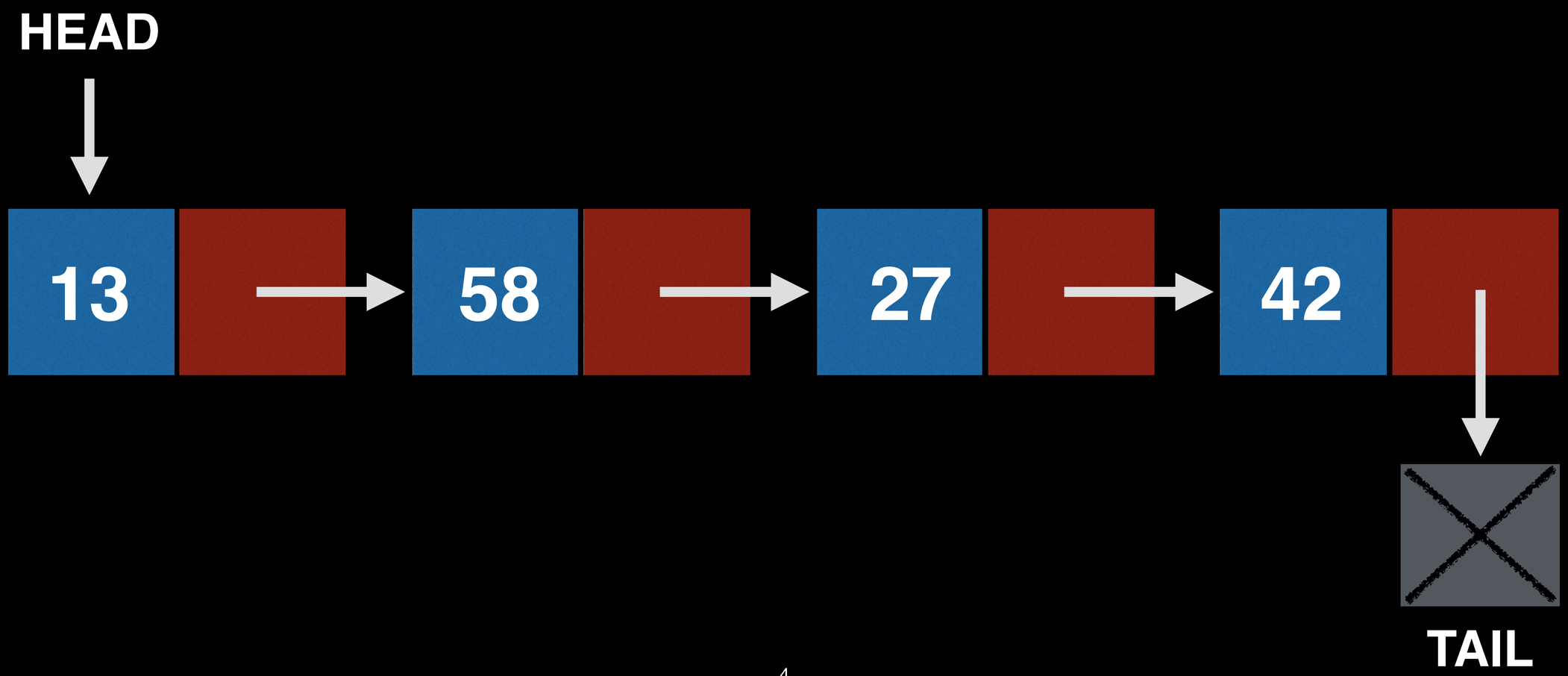
Each node stores some piece of **data**

Each node has a pointer to the **next** node

A favorite topic in Interviews… (and Queues!)

To Find a Node…

Use the next pointer to walk the List



Data Access

- No direct access to data, must look through nodes
- Operations at the front are quick, at the back are slow
- Accessing data further down the list requires a longer walk
- This “linear” time cost is more expensive than “constant” time of an Array
- Basically, Linked Lists slower than Arrays, $O(n) > O(1)$

Traversing the List

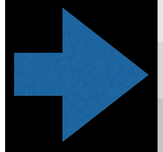
- Traversing a linked list is an important skill to have.
- It’s easy once you know how to do it.
- There are two ways commonly used:
  - Recursion (maybe easier to read, but more overhead)
  - Iteration (less overhead, uses a while loop)

Iteratively Traversing

```
class LinkedList {

  var head : Node?

  func addValue(value : Int) {
    if self.head == nil {
      self.head = Node(value: value)
    } else {
      var currentNode = self.head
      while currentNode!.next != nil {
        currentNode = currentNode!.next
      }
      currentNode?.next = Node(value: value)
    }
  }
}
```

Recursively Traversing

```
class LinkedList {

  var head : Node?

  func addValue(value : Int) {
    if self.head == nil {
      self.head = Node(value: value)
    } else {
      self.head?.addValue(value)
    }
  }
}
```



```
class Node {
  var value : Int
  var next : Node?
  init(value: Int) {
    self.value = value
  }

  func addValue(value : Int) {
    if self.next == nil {
      self.next = Node(value: value)
    } else {
      self.next?.addValue(value)
    }
  }
}
```

Demo

Week Wrap

Week 2 Patterns

- Crucial patterns for any language: Single Responsibility Pattern, Model View Controller, Lazy Loading, Concurrency, Callbacks, Encapsulation, **Functional Programming, DRY, Rule of 3**
- Crucial patterns for iOS: Delegation, Singletons, **Custom Protocols**
- **.gitignore**

Week 2 Frameworks

- Objective-C System frameworks: Foundation, UIKit
- Cocoa Frameworks: **CoreImage**, **Photos**, Social
- 3rd Party Frameworks: **Parse**

Week 2 UI Techniques

- Storyboards
- Autolayout
- Labels, ImageViews, Buttons
- **Collection Views**
- **Alert Controllers**
- **Animating Constraints**
- **General Segues**

Week 2 Important Classes

- **UITabBarController**
- UINavigationController

Week 2 Swift features

- Optional Binding
- Type methods
- [weak self]
- Closure Expressions
- **Property Observers**

Extra Credit Features

Selecting a thumbnail applies the filter to the primary image

Post more than just the image to Parse

- Add a message or hashtags to each post.
- When you show the posts in the other tab, show this text with each image as well.

Create an album with the photos framework to keep a local copy of all the photos you upload