Мои знания алгоритмов для структур данных

Содержание

Массив

Связный список

`Связный список` – цепочка узлов, в которой узел имеет только Данные и Указатель на другой(-ие) узел

Односвязный

`Односвязный список` – связный список, который имеет один указатель

Структура

Для целочисленных (`Int`)

```
public class ListNode {
    public var val: Int
    public var next: ListNode?

    public init(_ val: Int, _ next: ListNode?) {
        self.val = val
        self.next = next
    }
}
```

Алгоритм прохода

```
var list = ListNode(...)

while list != nil {
    // фиксируем по current.val
    current = current.next
}
```

Fast slow

Пока один движется как `current` из прошлого примера (`slow`), другой прыгает через один узел (`fast`)

```
var slow: ListNode = list
var fast: ListNode = list

while fast?.next != nil {
    slow = slow?.next
    fast = fast?.next?.next
}
```

Двусвязный

`Двусвязный список` – связный список, который имеет один указатель

Структура

Для целочисленных (`Int`)

```
public class ListNode {
    public var val: Int
    public var next: ListNode?
    public var previous: ListNode?

    public init(_ val: Int, next: ListNode?, previous: ListNode?) {
        self.val = val
        self.next = next
        self.previous = previous
    }
}
```

Граф

Дерево

`Дерево` – граф без циклов

Бинарное дерево

`Бинарное дерево` – дерево с 2-мя ветками: Правое и Левое

Структура

Для целочисленных (`Int`)

```
public class TreeNode {
    public var val: Int
    public var left: TreeNode?
    public var right: TreeNode?
    public init(val: Int, left: TreeNode?, right: TreeNode?) {
        self.val = val
        self.left = left
        self.right = right
    }
}
```

Алгоритм прохода по бинарному дереву

DFS

```
var stack: [TreeNode] = [root] // дерево

while !stack.isEmpty {
    let node = stack.popLast()

    // тут можно фиксировать текущую позицию через node

    if let left = node?.left {
        stack.append(left)
    }
    if let right = node?.right {
        stack.append(right)
    }
}
```

BFS

```
var queue: [TreeNode] = [root] // дерево

while !queue.isEmpty {
    let node = queue.popFirst()

    // тут можно фиксировать текущую позицию через node

    if let left = node.left {
        queue.append(left)
    }
    if let right = node.right {
        queue.append(right)
    }
}
```