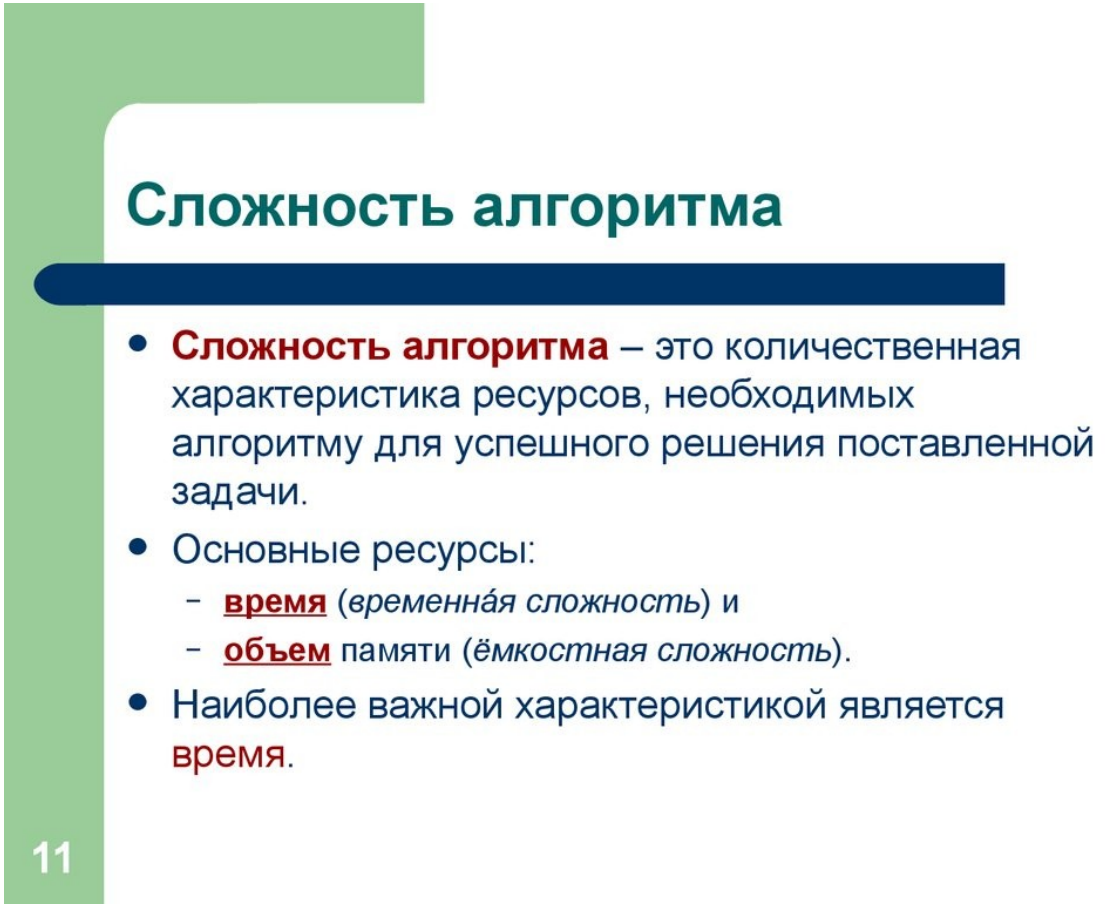
Опеределение сложности алгоритмов
(пример со столовой(O(1)) и рестораном(O(N)))
O(1) – постоянное время
O(N) – линейное время, N – кол-во людей



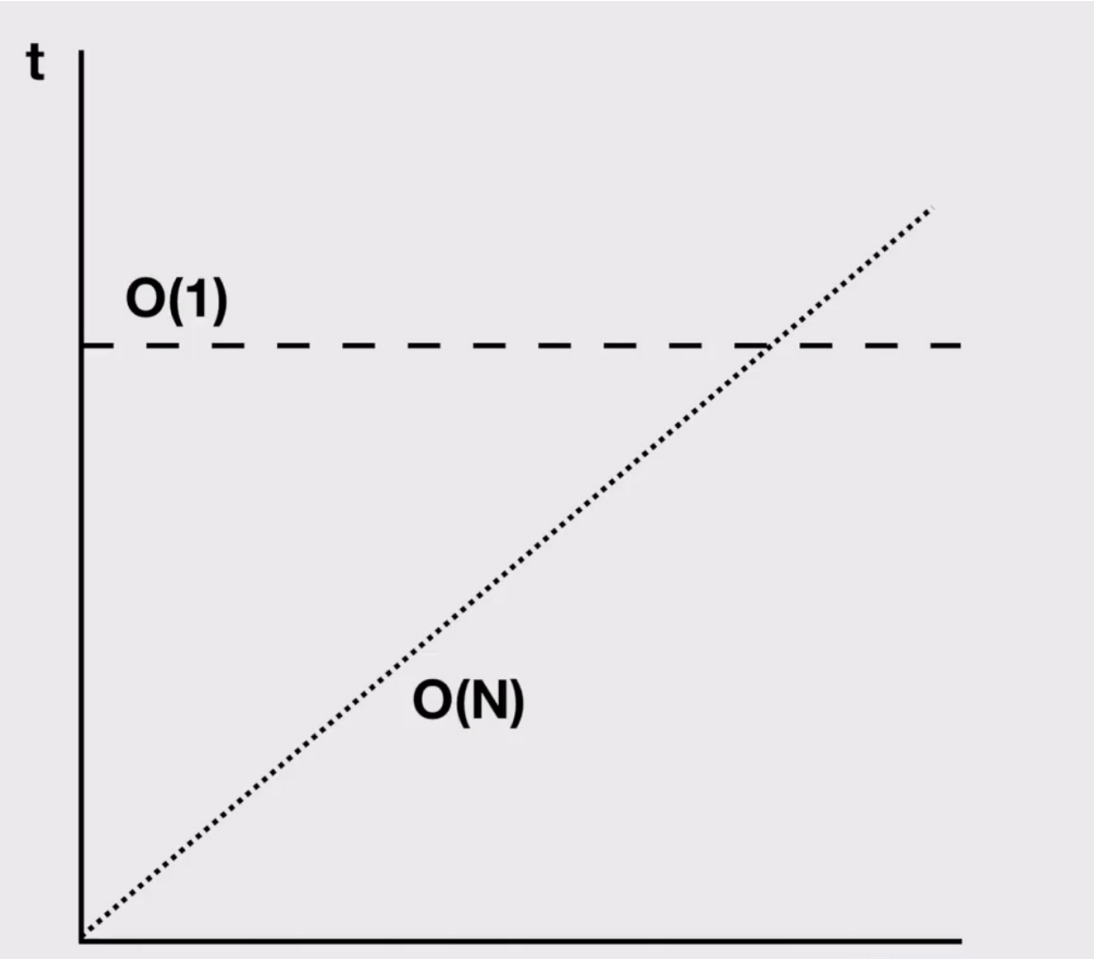
Определите сложность

```swift
func getFirstElement(array: [Int]) -> Int {
    return array[0]
}

getFirstElement(array: [4,3,6,8,2,5,6])
```

```swift
func getSum(array: [Int]) -> Int {
    var sum = 0
    for (_, item) in array.enumerated() {
        sum = sum + item
    }
    return sum
}
getSum(array: [1,4,6,8,4,6])
```

График

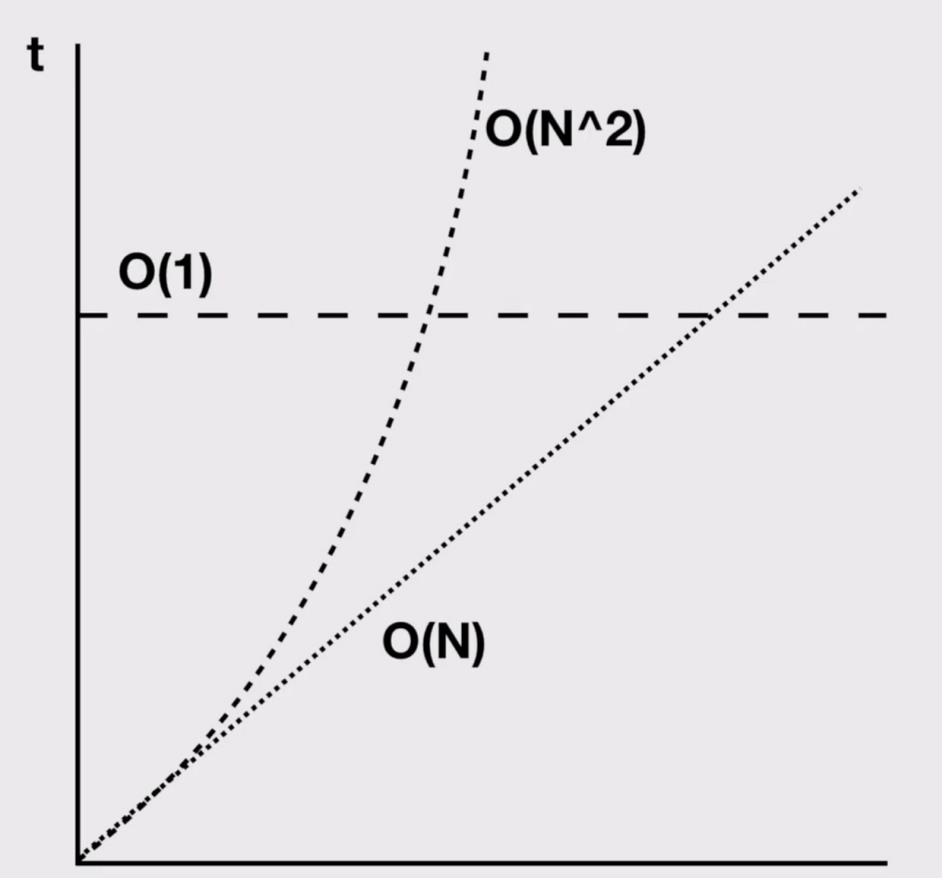


Необходимо найти первую пару в массиве

```swift
func findFirstPair(array: [Int]) -> Int? {
    for i in 0 ..< array.count - 1  {
        for j in i + 1 ..< array.count  {
            if array[i] == array[j] {
                return array[i]
            }
        }
    }
    return nil
}

findFirstPair(array: [1,3,6,8,1,4,7,9,3,5])
```

График



Еще примеры

```swift
func foo(array: [Int], min: inout Int, max: inout Int) {
    for (_,item) in array.enumerated() {
        if item < min {
            min = item
        }
        if item < max {
            max = item
        }
    }
}
```

O(N)

```swift
func foo1(array: [Int], min: inout Int, max: inout Int) {
    for (_,item) in array.enumerated() {
        if item < max {
            max = item
        }
    }
    for (_,item) in array.enumerated() {
        if item < min {
            min = item
        }
    }
}
```

O(2N) ⟶ **O(N)**

Еще примеры

```swift
func foo(arrayA: [Int], arrayB: [Int]) {
    for (_,item) in arrayA.enumerated() {
        print(item)
    }
    for (_,item) in arrayB.enumerated() {
        print(item)
    }
}
```

O(A + B)

```swift
func foo1(arrayA: [Int], arrayB: [Int]) {
    for (_,_) in arrayA.enumerated() {
        for (_,item) in arrayB.enumerated() {
            print(item)
        }
    }
}
```

O(A * B)

Еще примеры

```swift
func foo1(arrayA: [Int], arrayB: [Int]) {
    for (_,_) in arrayA.enumerated() {
        for (_,item) in arrayB.enumerated() {
            print(item)
        }
    }

    for (_,item) in arrayA.enumerated() {
        print(item)
    }
}
```

~~O(A * B + A)~~

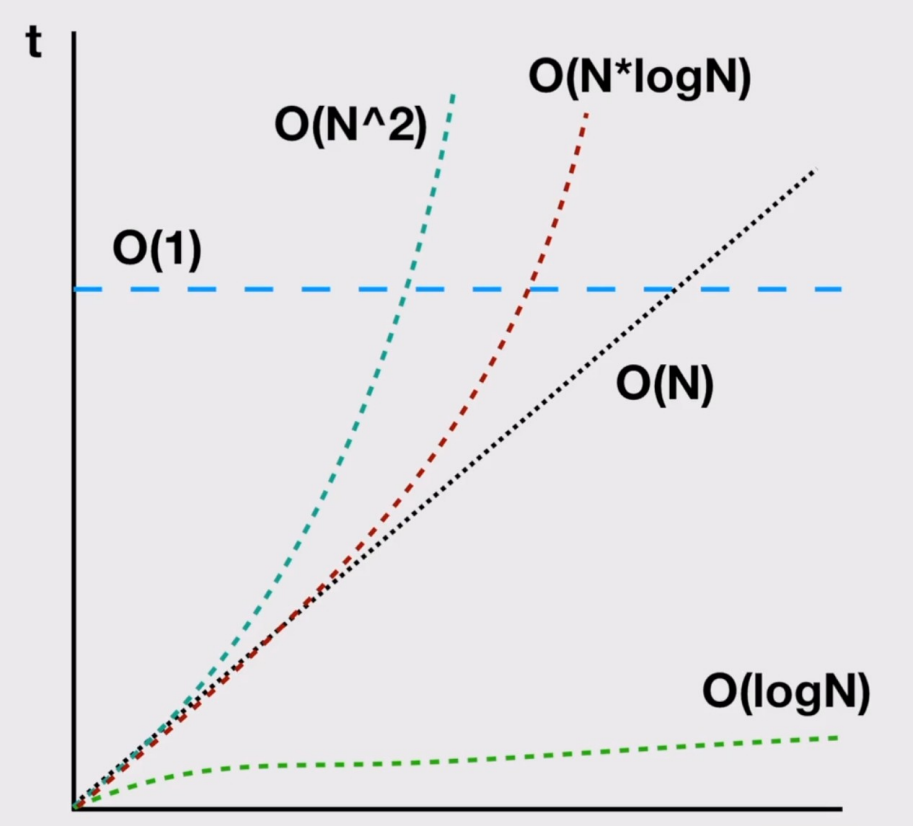
O(A * B)

O(N^2 + N) ⟶ O(N^2)

O(N + logN) ⟶ O(N)

O(3N + 2^N) ⟶ O(2^N)

График



Решите задачу #1

- На вход подается массив чисел [Int], и число value: Int
- На выходе получить индексы элементов, сумма которых равна value

Уточняющие вопросы

- Сколько может быть решений?
- Элементы повторяются?
- Nil если ничего не найдено?

Решение

- (arr: [3, 7, 6, 8, 16, 5], value: 15) → O(N^2)
- Используем цикл в цикле и сравним все элементы

Используем Hash таблицу

- (arr: [3, 7, 6, 8, 16, 5], value: 15) → O(N)
- Записывая каждый элемент проверяем его, ищем, если (15 - новый элемент) есть уже в записанных ключах, то мы возвращаем их значения

Ключ	Значение
3	0
7	1
6	2
8	3

Решите задачу #2

- Удалить одинаковые значение из отсортированного массива, добиться O(N)
- [2, ~~2~~, 5] → 2
- [0, ~~0~~, 1, ~~1~~, 3, 4, ~~4, 4~~, 5, ~~5~~, 8] → 6

Решение

- Сравниваем элемент с предыдущим, если совпадает, удаляем
- [0, ~~0~~, 1, ~~1~~, 3, 4, ~~4, 4~~, 5, ~~5~~, 8] → 6

Решите задачу #3

- На вход подается [Int], в этом массиве нужно найти такой подмассив, который, если вы отсортируете по возрастанию→ отсортируется весь массив. O(N).
- [1,4,3,2,6] → искомый [4,3,2] → 3
- [6,4,10,10,4,15] → искомый [6,4,10,10,4] → 5
- [1,1]→ нет → 0

Решение

- [1, 4, 2, 3, 2, 6]
- a = 4, b = 2
- L = b – a + 1 // (4 – 1 + 1 = 4)
- var max = 1;
- Сравниваем с максимальным (4 > max(1)?) → max = 4
- (2 > max(4)?) → записываем индекс 2
- (3 > max(4)?) → записываем индекс 3
- (2 > max(4)?) → записываем индекс 4
- (6 > max(4)?) → Max = 6
- Дальше ищем минимальное значение в обратном порядке
- var min = 6
- 2 < min(6) → min = 2
- 3 < min(2) → записываем индекс 3
- … нашли стартовый индекс 1