

LET SWIFT ☀ JUNE 14TH, 2018

PIOTR SOCHALEWSKI

---

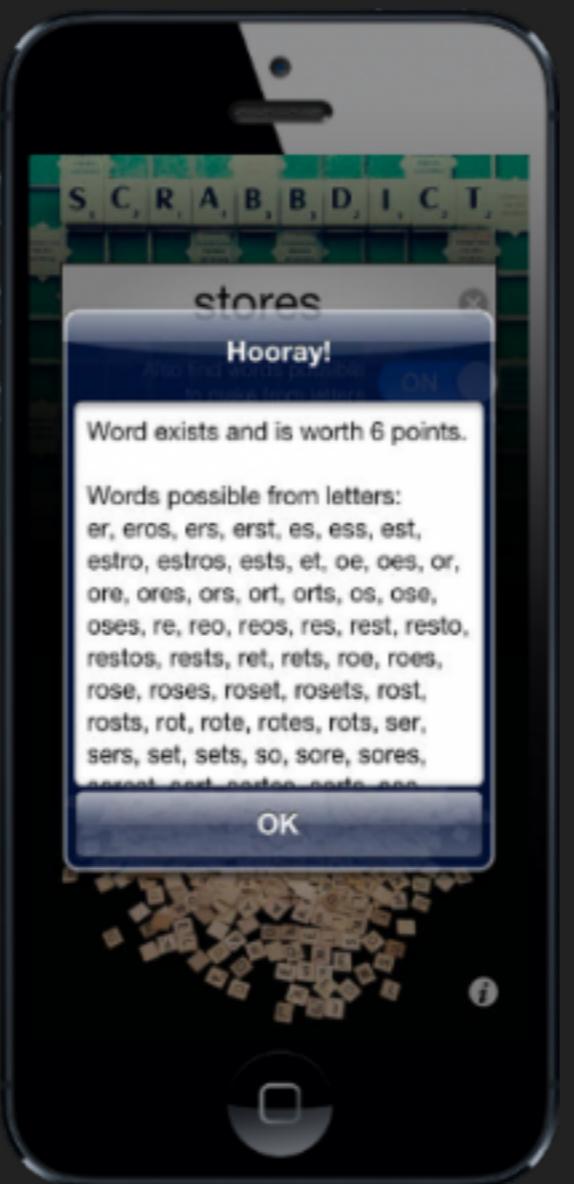
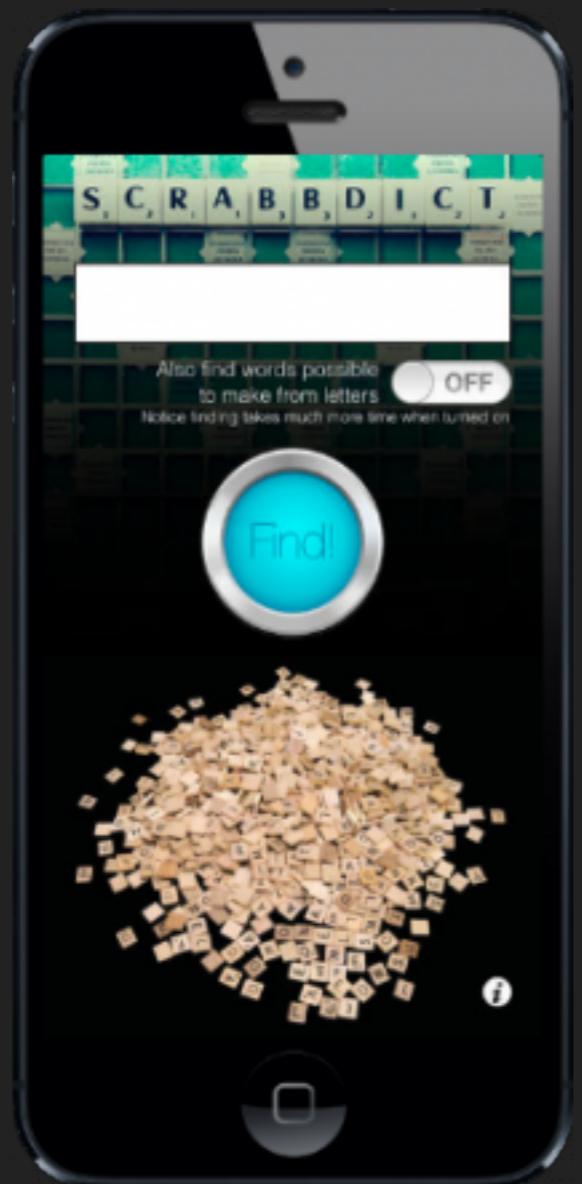
HANDLING ENORMOUS  
COLLECTION TYPES IN SWIFT

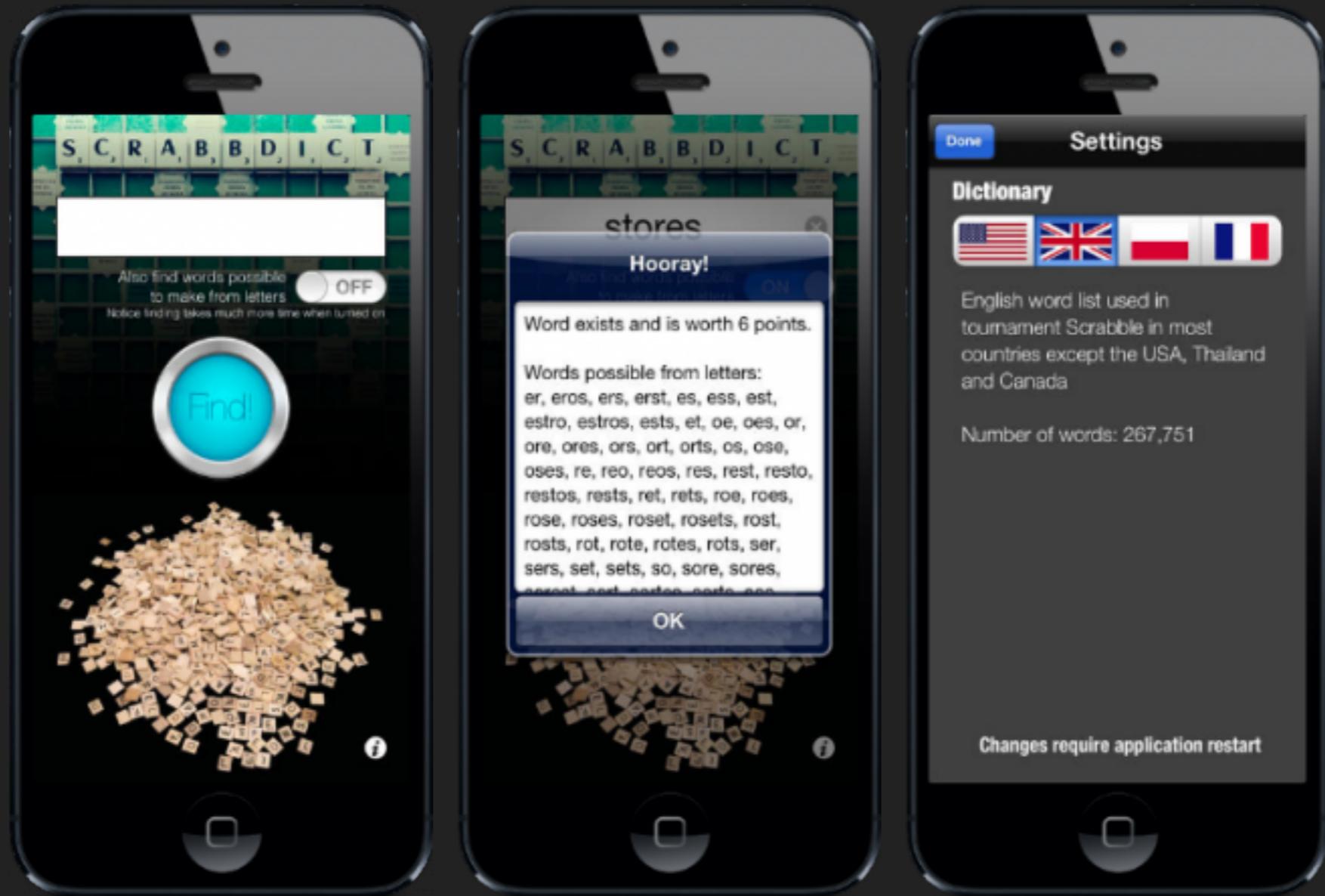
LET SWIFT ☀ JUNE 14TH, 2018

PIOTR SOCHALEWSKI

---

# SCRABBLE DICTIONARY CASE STUDY





## Features:

- ▶ validating words
- ▶ finding words containing letters
- ▶ multiple dictionaries



276,643



187,632



386,264



**2,965,223**

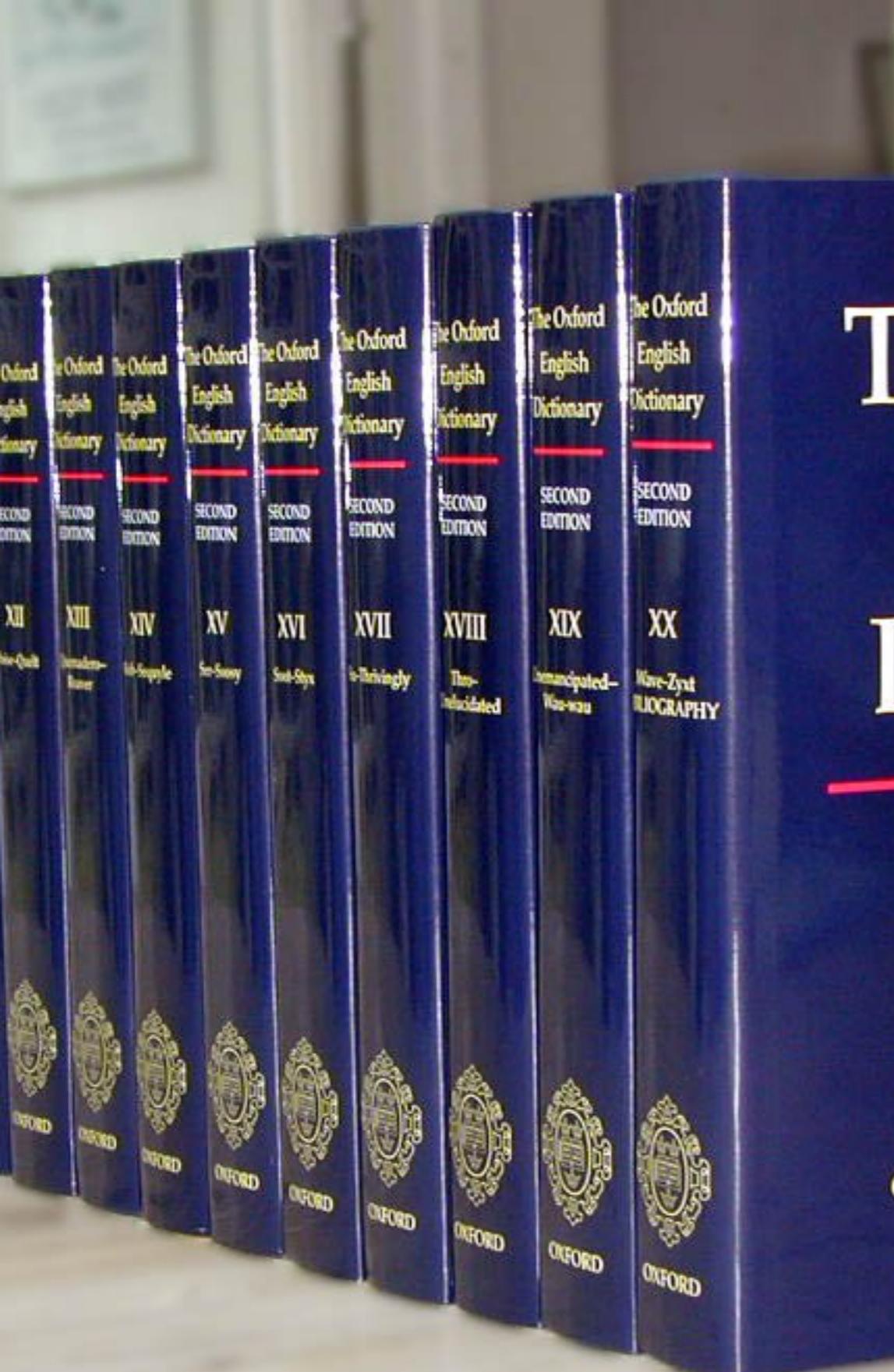




**DICTIONARY  
IS TOO BIG**

- ▶ It contains **almost 3.000.000 words**
- ▶ How to store it?
- ▶ How to make it fast?

Use raw .txt file  
and load it to NSArray or NSSet



# NOPE!

More than 300 MB  
RAM used just after  
start.

~~Use raw .txt file  
and load it to NSArray or NSSet~~

Split .txt file to multiple files  
and use them when needed

The image shows a screenshot of the Xcode IDE. On the left, the Project Navigator displays the project structure:

- Scrabbdict (target)
- Scrabbdict (group)
  - Generic Views
  - View Controllers
  - Helpers
- Files
  - en\_GB\_sowpods.txt
  - en\_US\_twl.txt
  - fr\_ODS6.txt
  - pl\_PL.txt
  - pl\_PL\_9.txt
  - pl\_PL\_10.txt
  - pl\_PL\_11.txt
  - pl\_PL\_12.txt
  - pl\_PL\_13.txt
  - pl\_PL\_14.txt
  - pl\_PL\_15.txt
- AppDelegate.swift
- Main.storyboard
- Assets.xcassets
- LaunchScreen.storyboard
- Info.plist

Middle section: A Swift file with code highlighting.

```
1 // Lang
2 // Scrabbdict
3 // Create
4 // Copy
5 // ...
6 // ...
7 // ...
8
9 import Foundation
10 import UIKit
11
12 enum Languages {
13     case en
14     case fr
15     case pl
16     case ...
17 }
18
19 var language: Languages = .en
20 }
21
22 var ...
23 }
24
25 var ...
26 }
27
28 }
29
30 var ...
31
32
33 }
```

Bottom section: A tab bar with three items.

# IT WORKS

But it is quite slow  
and limits  
possibilities.



**SCRABBDICT 3.0  
WANTS TO BE  
MORE USEFUL**

Apple Inc.

Mac iPad iPhone Watch TV Music Wsparcie

App Store Preview

This app is only available on the App Store for iOS devices.

**Scrabbdict** 4+

Offline word game dictionary

Piotr Sochalewski

2,99 zł

**Screenshots** [iPhone](#) [iPad](#)

scrabble

clabbers 14  
scrabble 14  
clabber 13  
scabble 13  
barbels 11  
cablers 11  
rabbles 11  
slabber 11  
babels 10  
barbel 10  
barbes 10  
braces 10  
cabers 10  
cabler 10  
cables 10  
rabble 10  
abbes 9  
acerb 9

stores

THE WORD IS VALID

6 POINTS

Cancel Settings Save

Dictionary

English SOWPODS

English TWL ✓

French

Polish

Official Tournament and Club Word List  
English official word authority for tournament  
Scrabble™ in the USA, Canada and Thailand

Number of words: 178,691

This app is only available on the App Store for iOS devices.

**Scrabbdict** 4+  
Offline word game dictionary  
Piotr Sochalewski

# Requested features:

- 1 points for found words
- simple regex patterns (blanks)
- better performance

The screenshot shows the Scrabbdict app's user interface. On the left, there's a table of words with their point values. In the center, a modal window displays the message "THE WORD IS VALID" with "6 POINTS" underneath. On the right, there's a dictionary settings screen with language options like English SOWPODS, French, and Polish, along with tournament information and a word count of 178,691.

# IMPROVING PERFORMANCE

**REALM?**

```
import RealmSwift

final class Dictionary: Object {
    @objc dynamic var language = ""
    let words = List<StringObject>()
}

final class StringObject: Object {
    @objc dynamic var value = ""

    override static func indexedProperties() -> [String] {
        return ["value"]
    }

    override static func primaryKey() -> String? {
        return "value"
    }
}
```



IT'S  
EXTREMELY  
SLOW

when getting real  
RLMObject values  
in a wrong way.

## REALM

---

```
let words: List<StringObject>

func validate(word: String) -> Bool {
    return words
        .first(where: { $0.value == word.lowercased() }) != nil
}
```

List<T>



[ T ]



filter { }

## REALM

---

```
let words: List<StringObject>

func validate(word: String) -> Bool {
    return words
        .first(where: { $0.value == word.lowercased() }) != nil
}

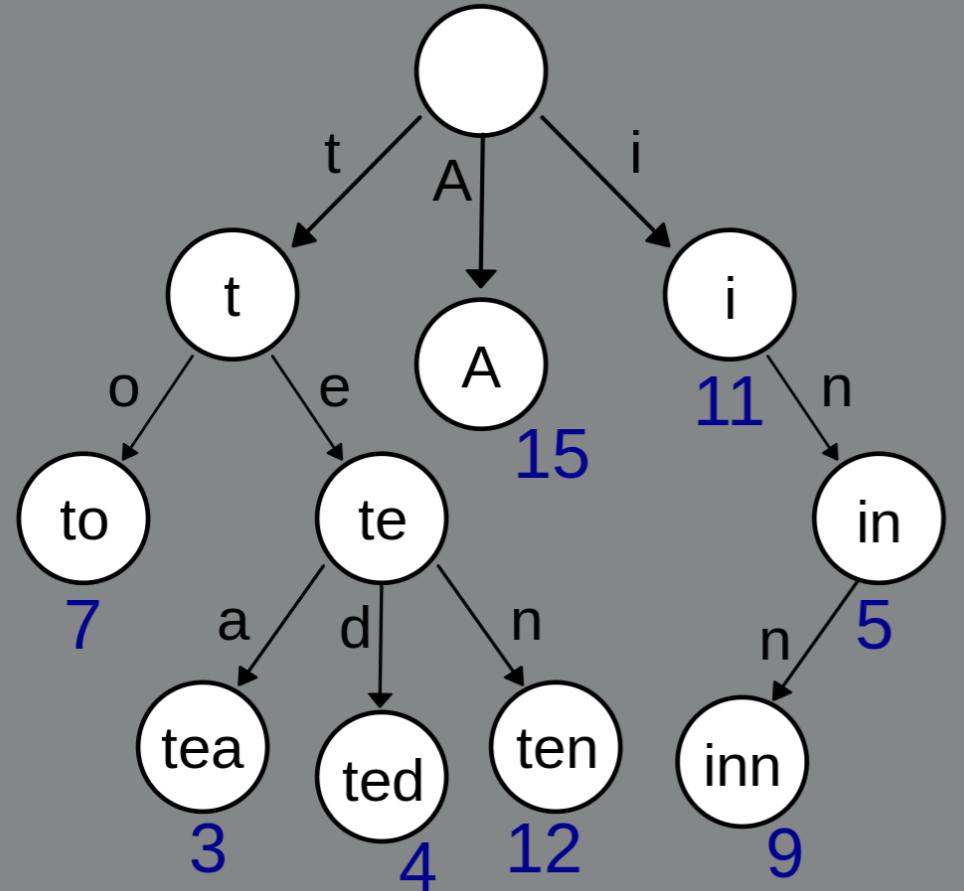
func validate(word: String) -> Bool {
    let predicate = NSPredicate(format: "%K == %@", "value", word.lowercased())
    return !words.filter(predicate).isEmpty
}
```

**TRY?**



# GOLDEN MEAN?

Slower than Flash\*,  
still requires  
splitting files, but is  
great



GOOD START  
FOR YOUR  
OWN TRIE  
AND OTHER COLLECTIONS

[github.com/mauriciosantos/Buckets-Swift](https://github.com/mauriciosantos/Buckets-Swift)

# COLLECTION DATA TYPES

---

- **Arrays**

- Standard Array,
- Array with reserved capacity that equals a count of strings,
- ContiguousArray,
- ContiguousArray with reserved capacity that equals a count of strings,
- CleverArray that is standard two-dimensional Array where arrays inside contains words with letters count that equals an index of array.

- **Sets**

- Standard Set,
- CleverSet that is standard Array of standard Sets where sets inside contains words with letters count that equals an index of set.

- **Tries**

- Modified Trie from Buckets-Swift by Mauricio Santos

- **Realm database**

- Standard Realm database with List of StringObjects,
- CleverRealm database with multiple Lists of StringObjects where each list contains words with different letters count.

# STANDARD VS „CLEVER” OBJECT TYPES

## STANDARD COLLECTION

aa  
aaa  
aaronowa  
aaronową  
aaronowe  
aaronowego  
aaronowej  
aaronowemu  
aaronowi  
aaronowy  
aaronowych  
aaronowym  
aaronowymi  
abace  
abachicie  
abachit  
abachitach  
abachitami  
abachitem  
abachitom  
abachitowi  
abachitów

## CLEVER COLLECTION

### 2 LETTERS

aa  
ad

### 3 LETTERS

aaa  
abo  
aby

### 4 LETTERS

abak  
abat  
abba

### 5 LETTERS

abace  
abaci  
abaka

## VALIDATORS

---

```
protocol Validator: class {
    init()

    func validate(word: String) -> Bool
    func words(from letters: String) -> [String]?
}

protocol CleverValidator: Validator {
    func regex(phrase: String) -> [String]?
}
```

## SAMPLE VALIDATOR

---

```
final class ArrayWordValidator: Validator {

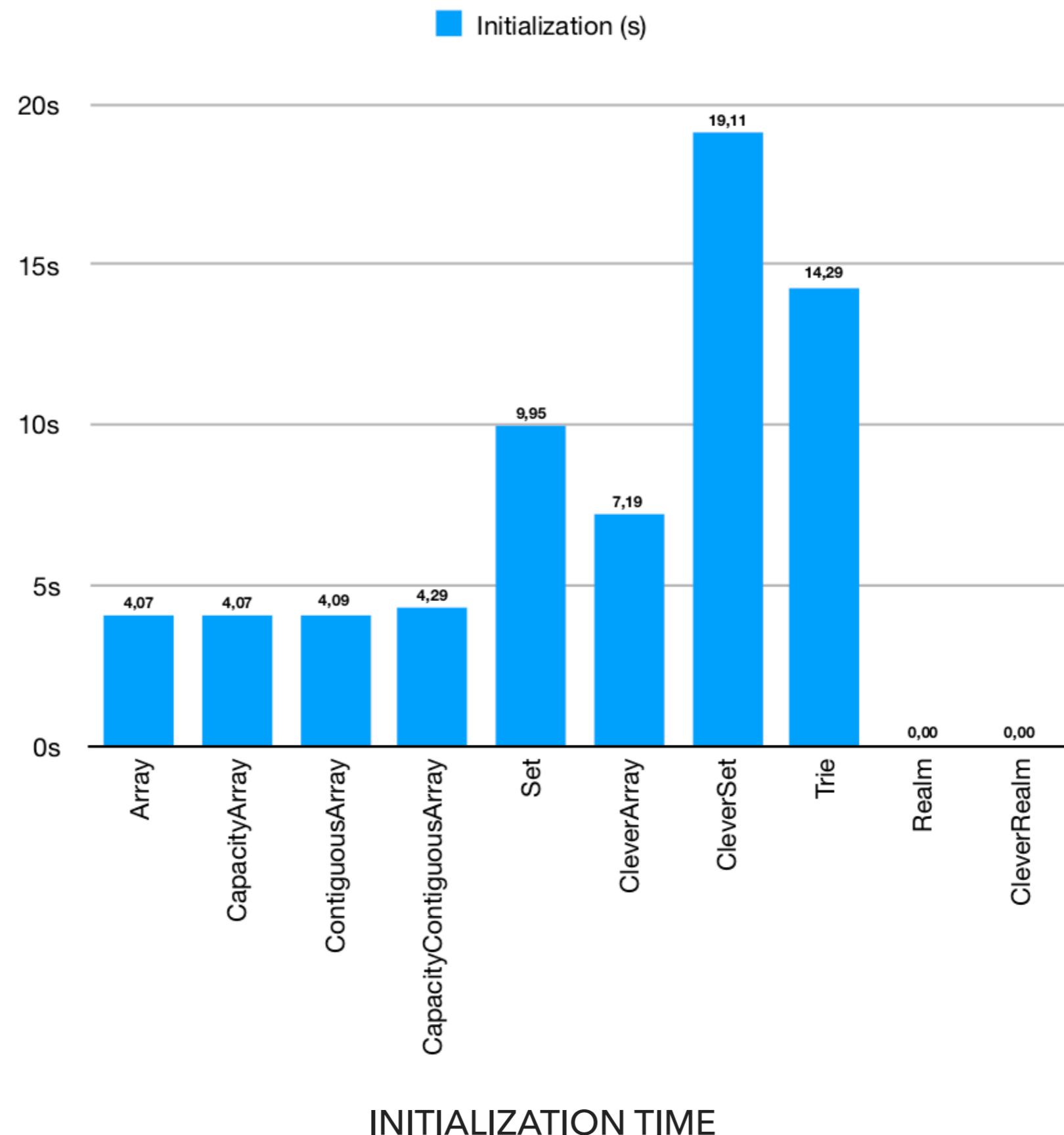
    /// An array of strings.
    private let words: [String]

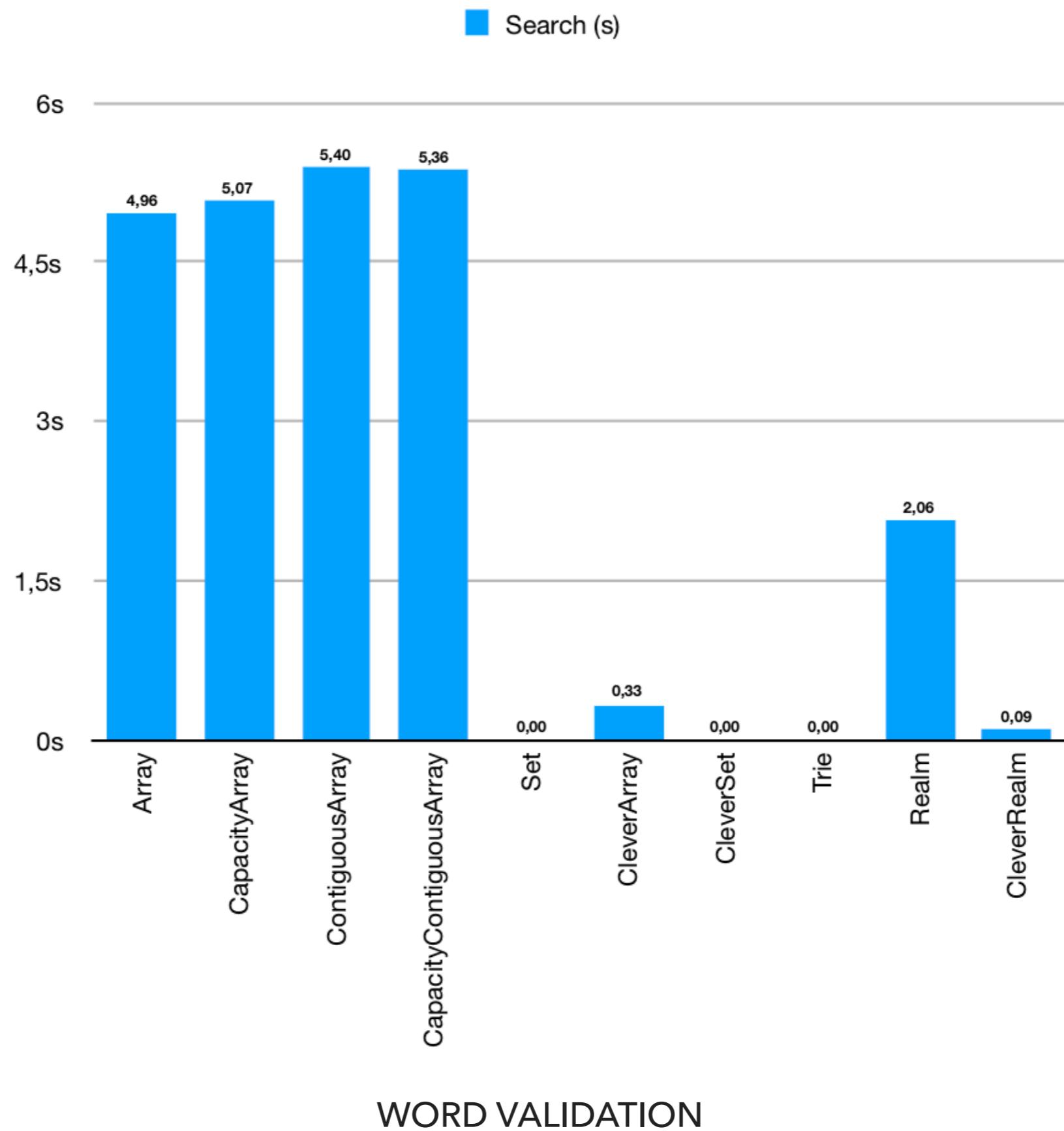
    init() {
        let string = try! String(contentsOf: fileURL, encoding: .utf8)
        words = string.components(separatedBy: .newlines)
    }

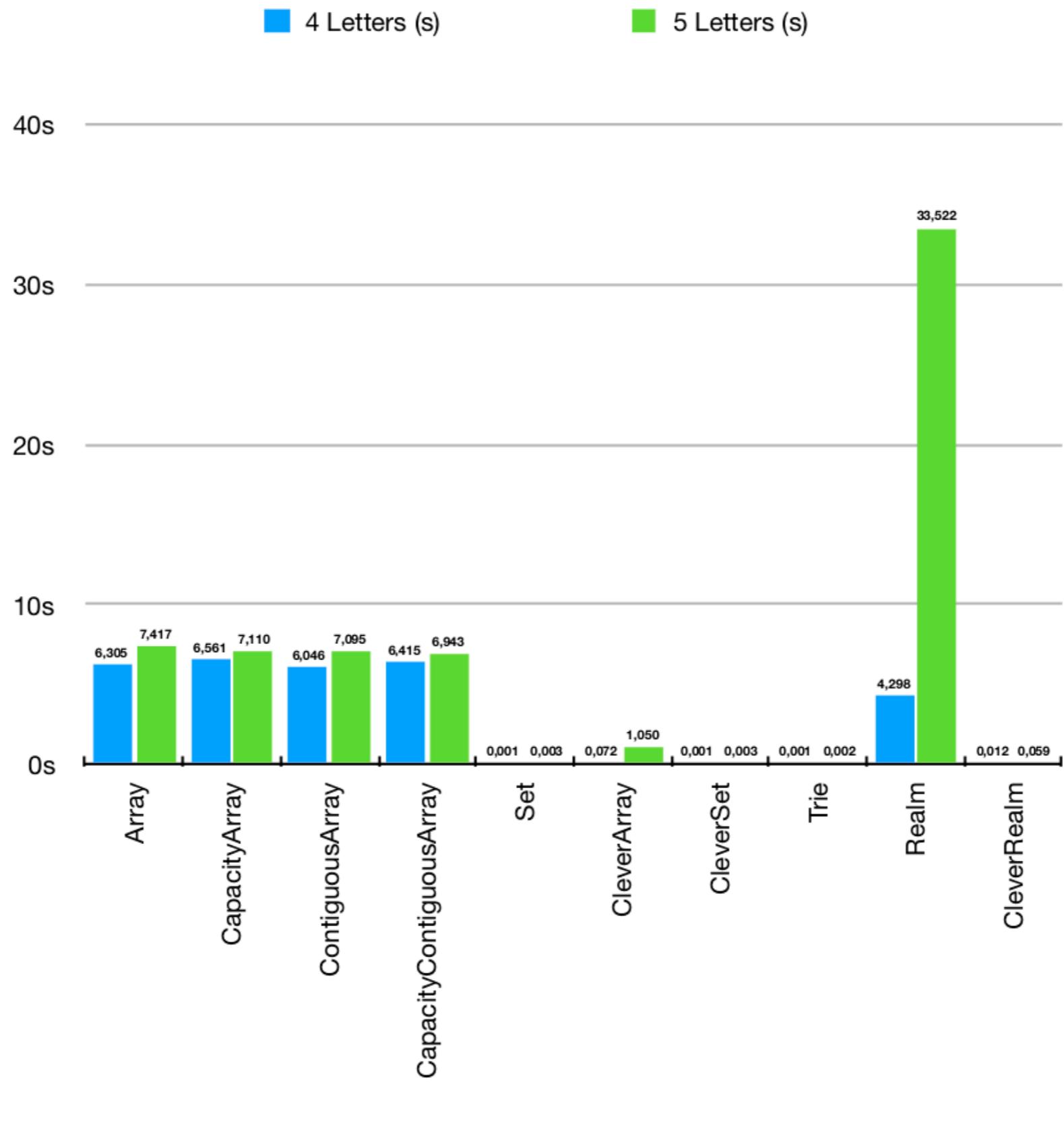
    func validate(word: String) -> Bool {
        return words.contains(word.lowercased())
    }

    func words(from letters: String) -> [String]? {
        let permutes = letters.lowercased()
            .map { String($0) }
            .permute()
        let predicate = NSPredicate(format: "SELF IN %@", permutes)

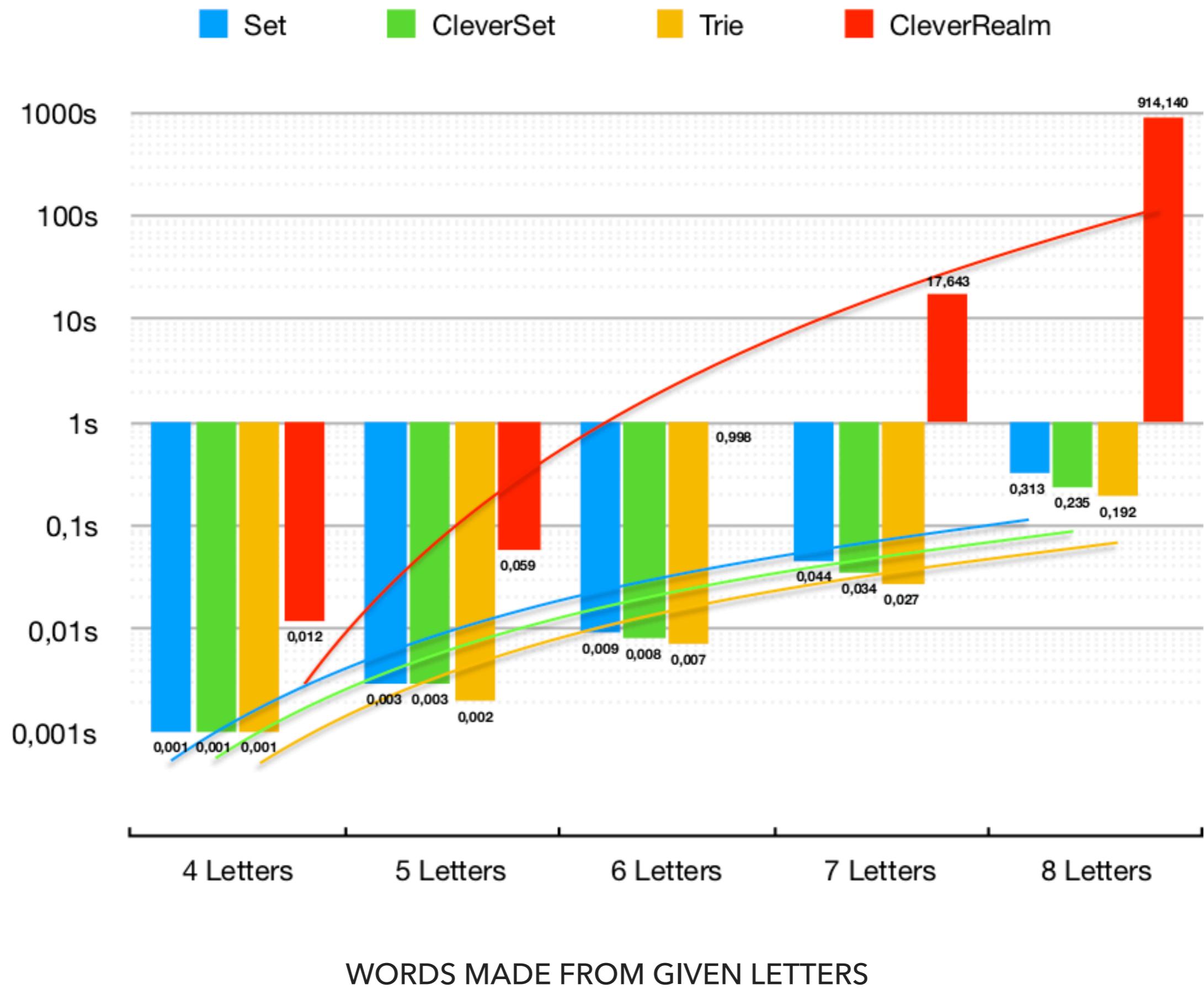
        // NSPredicate is multiple times faster than permutes.filter
        // { words.contains($0) }
        return words.filter { predicate.evaluate(with: $0) }
    }
}
```

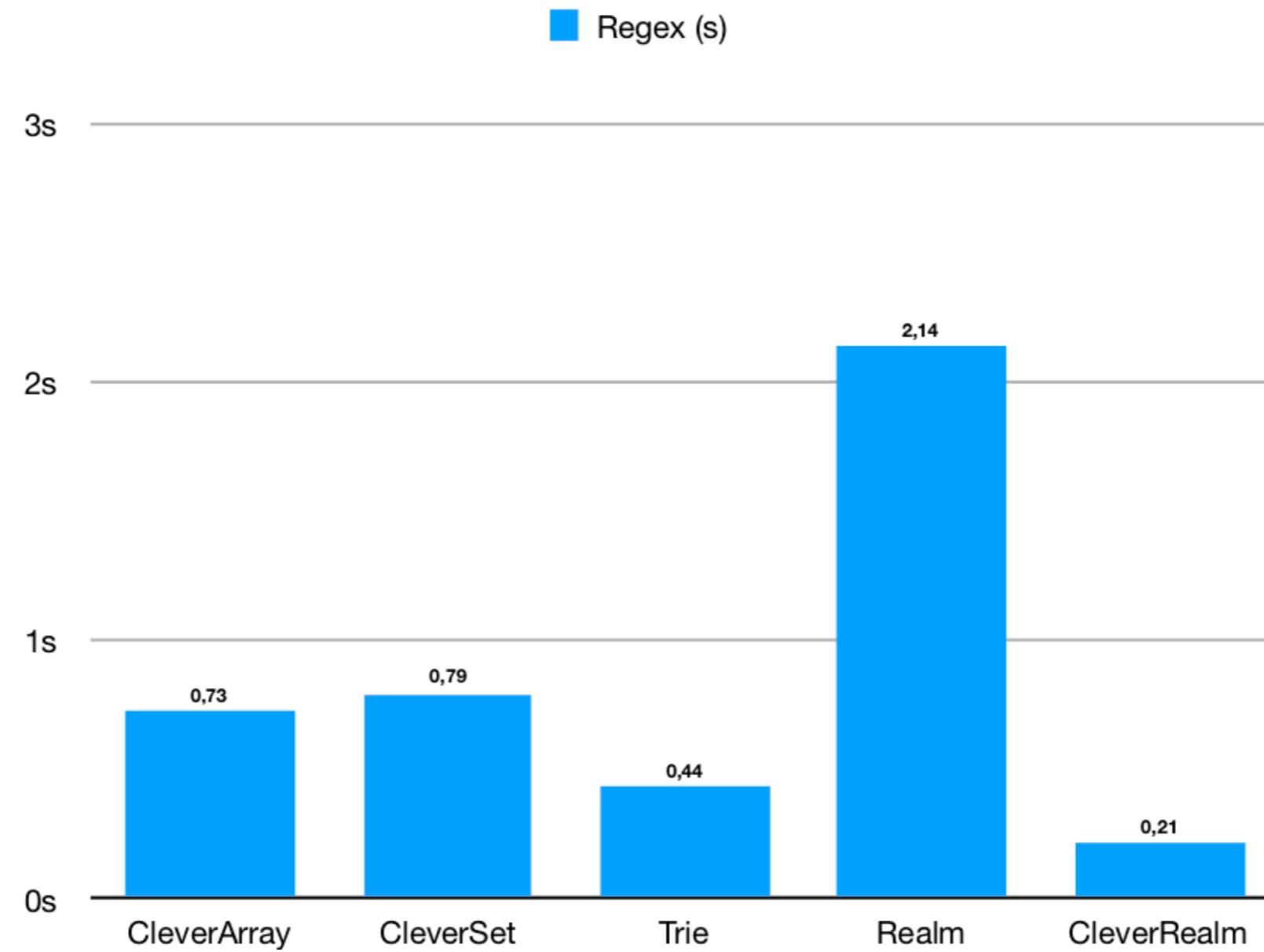




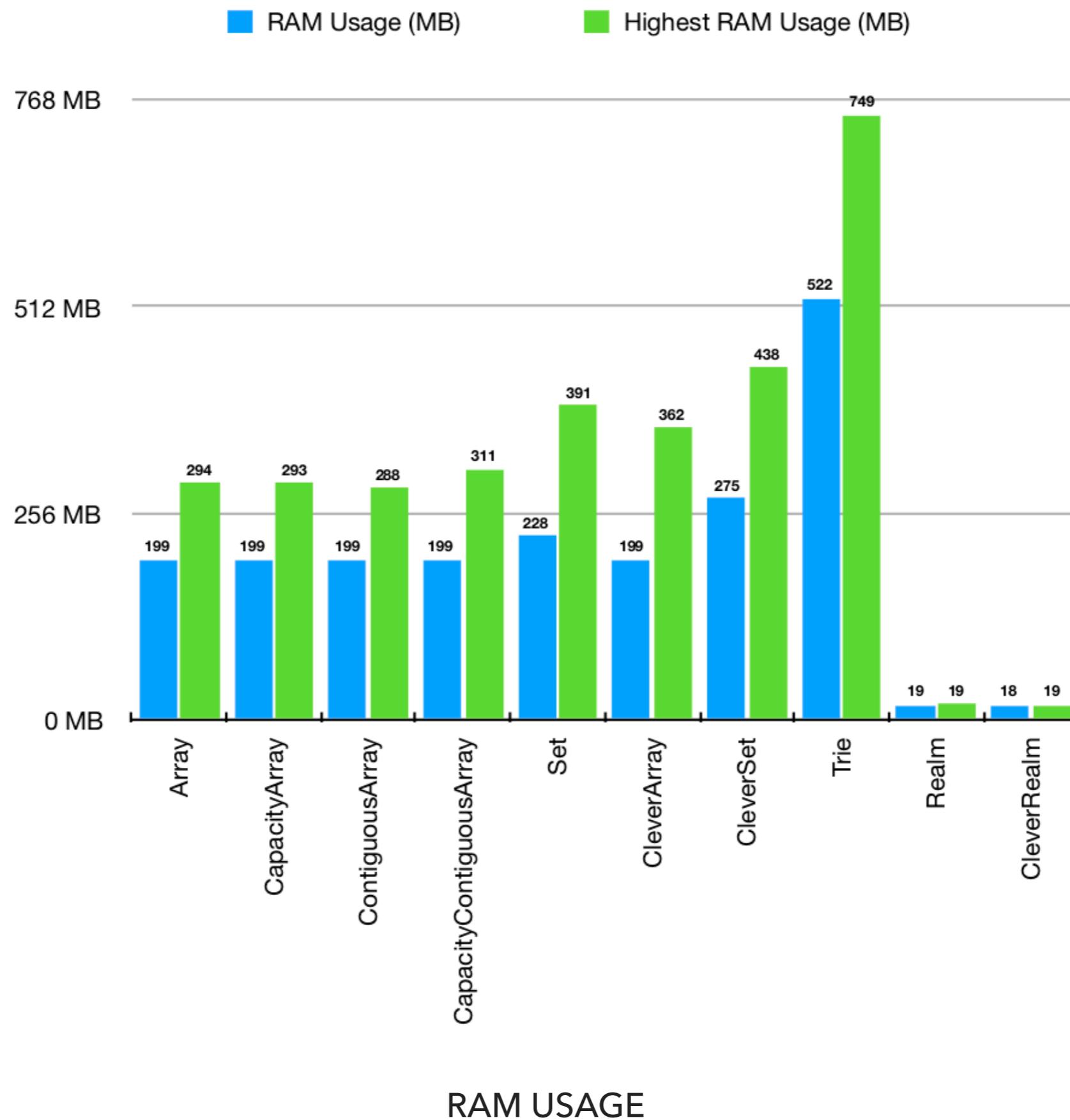


WORDS MADE FROM GIVEN LETTERS





SIMPLE REGEX (BLANKS)





REALM + TRIE = ❤

The screenshot shows a Mac OS X browser window displaying the netguru.co website. The page features a navigation bar with links for Services, Clients, About us, Resources, Blog, Codestories, Careers, and an 'Estimate project' button. A sidebar on the right contains a video thumbnail for a career video.

**Handling Enormous Collection Types in Swift**

Piotr Sochalewski | Feb 16, 2018 | 11 min read | [iOS](#)

Usually there is no difference when you choose Array or Set. You take Set if you want a collection type with unique unordered elements. But you do not care about performance or RAM usage, because they are often similar. Have you ever asked yourself what happens if your collection has to store almost 3 millions of elements like strings?

[Read more](#)

# Piotr Sochalewski

iOS Developer w **Netguru**  
piotr.sochalewski@netguru.co