LET SWIFT ☀ JUNE 14TH, 2018

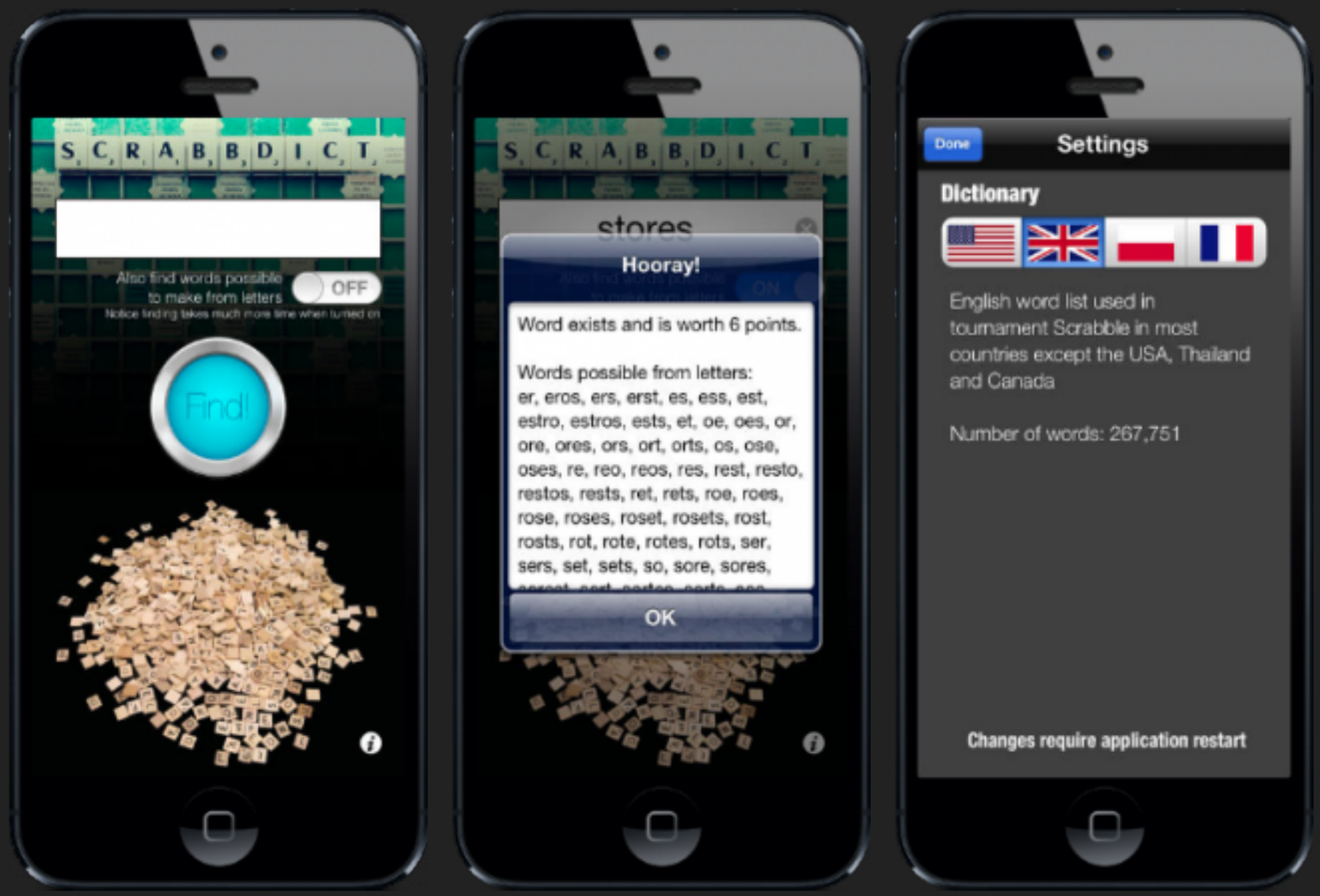
PIOTR SOCHALEWSKI

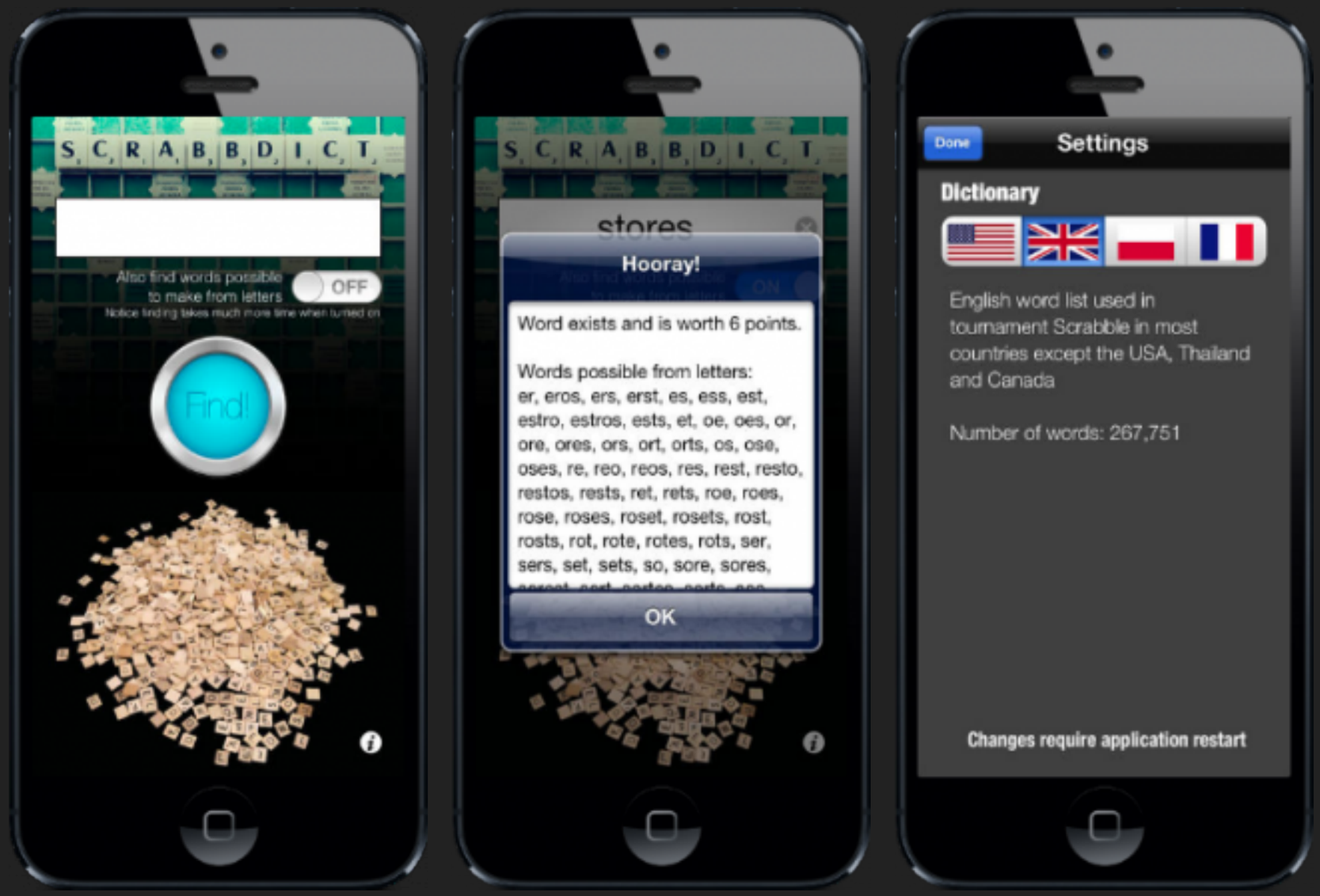
HANDLING ENORMOUS COLLECTION TYPES IN SWIFT

LET SWIFT ☀ JUNE 14TH, 2018

PIOTR SOCHALEWSKI

SCRABBLE DICTIONARY CASE STUDY


S C R A B B D I C T
Also find words possible to make from letters
OFF
Notice finding takes much more time when turned on
Find!
stores
Hooray!
Word exists and is worth 6 points.
Words possible from letters:
er, eros, ers, erst, es, ess, est, estro, estros, ests, et, oe, oes, or, ore, ores, ors, ort, orts, os, ose, oses, re, reo, reos, res, rest, resto, restos, rests, ret, rets, roe, roes, rose, roses, roset, rosets, rost, rosts, rot, rote, rotes, rots, ser, sers, set, sets, so, sore, sores,
OK
Done
Settings
Dictionary
English word list used in tournament Scrabble in most countries except the USA, Thailand and Canada
Number of words: 267,751
Changes require application restart



Features:

- validating words
- finding words containing letters
- multiple dictionaries

🇺🇸	🇬🇧	🇫🇷	🇵🇱
276,643	187,632	386,264	**2,965,223**

😮

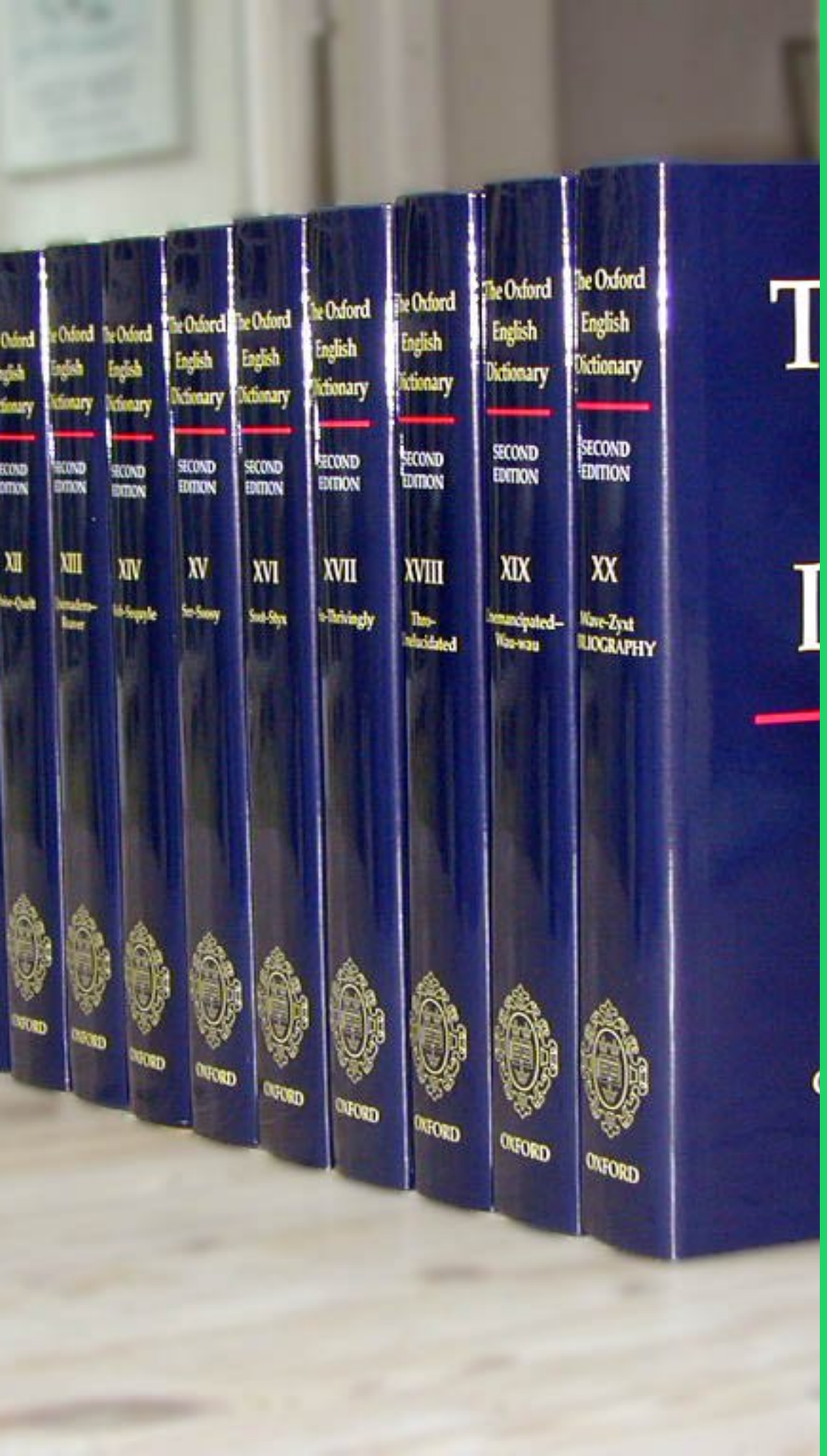
DICTIONARY IS TOO BIG

🇵🇱 DICTIONARY IS TOO BIG

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

SCRABBDICT 1.0-2.0

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

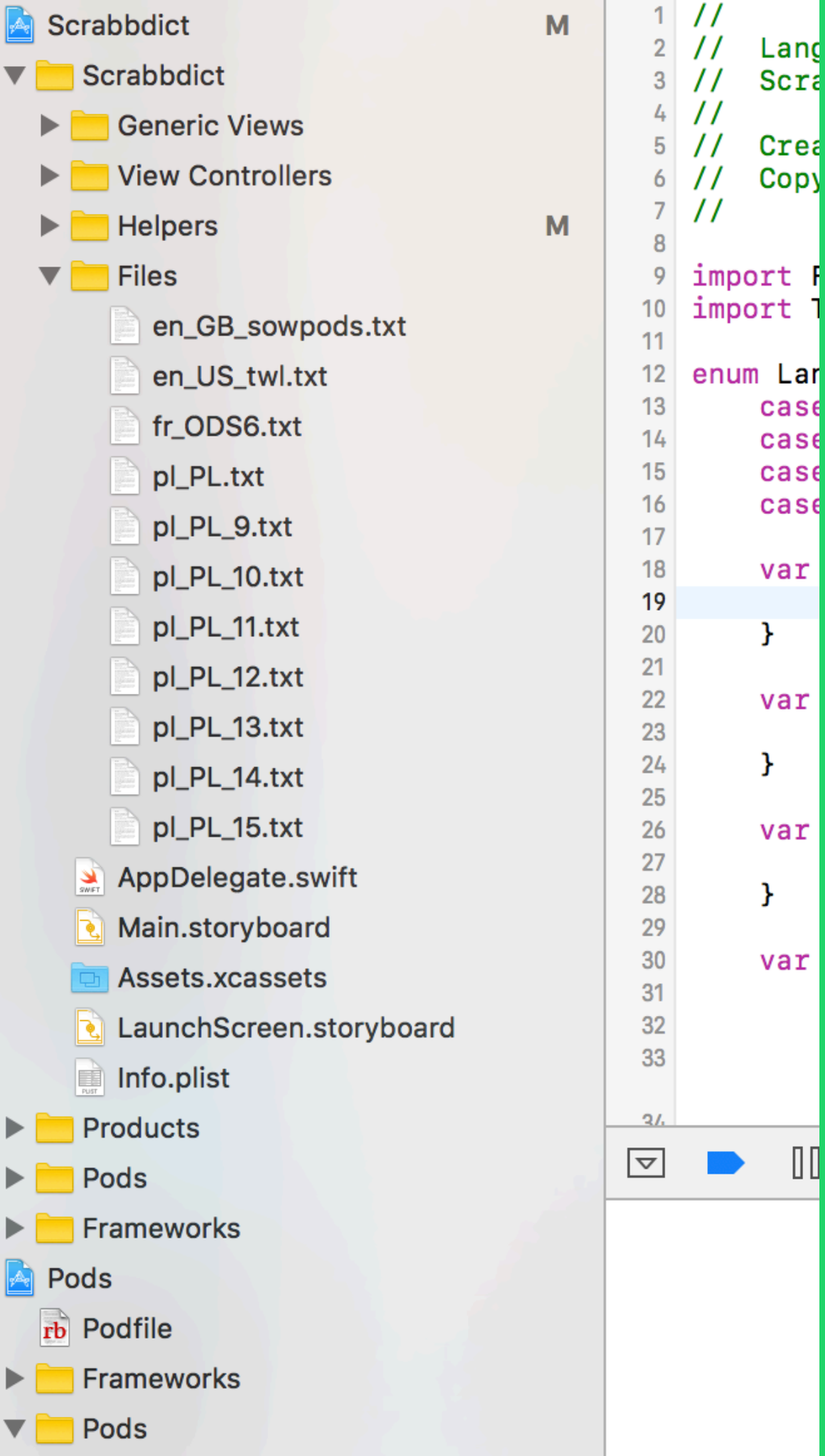


NOPE!

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



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.

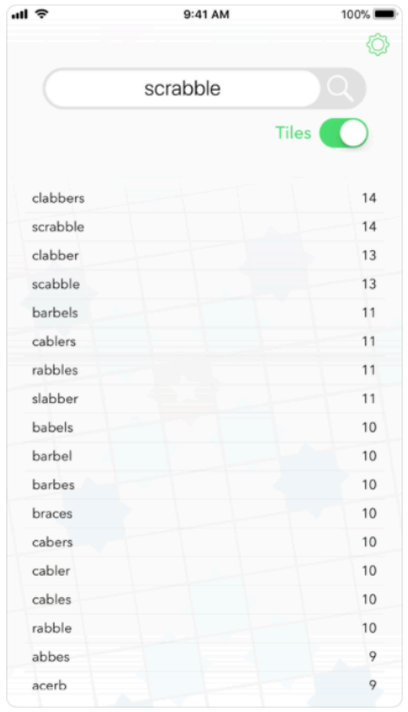
Scrabbdict 4+

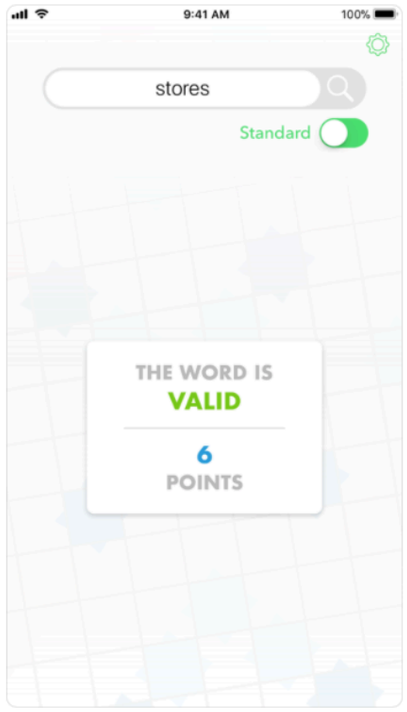
Offline word game dictionary

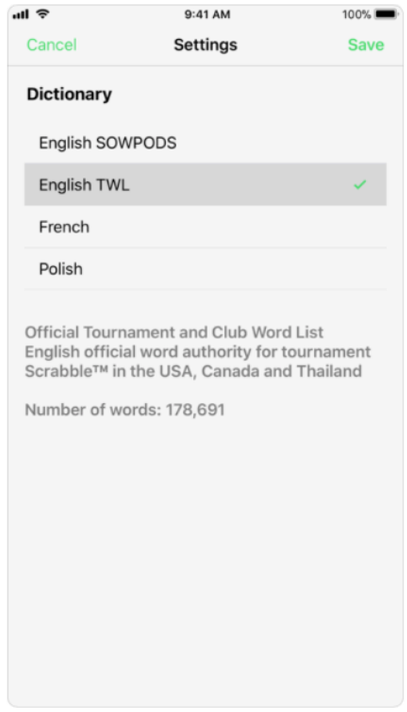
Piotr Sochalewski

2,99 zł

Screenshots iPhone iPad







Requested features:

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

IMPROVING PERFORMANCE

REALM?

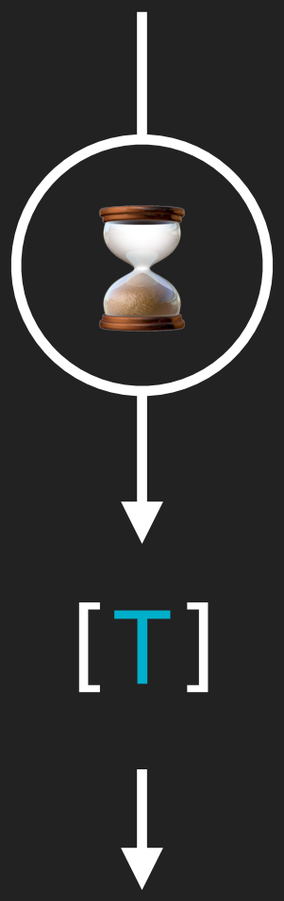
```
import RealmSwift

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.

```
let words: List<StringObject>

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

List<T>



[T]

filter { }

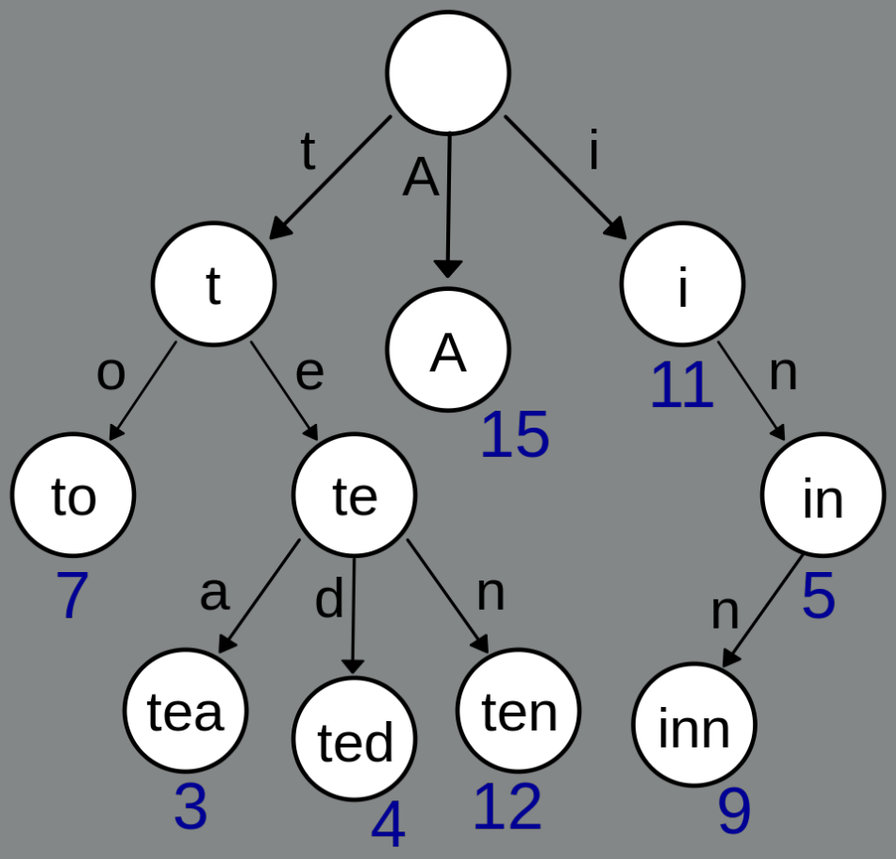
```
let words: List<StringObject>

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
}
```

TRIE?

GOLDEN MEAN?

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

* - a superhero from DC Comics, not Adobe Flash



GOOD START FOR YOUR OWN TRIE

AND OTHER COLLECTIONS

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

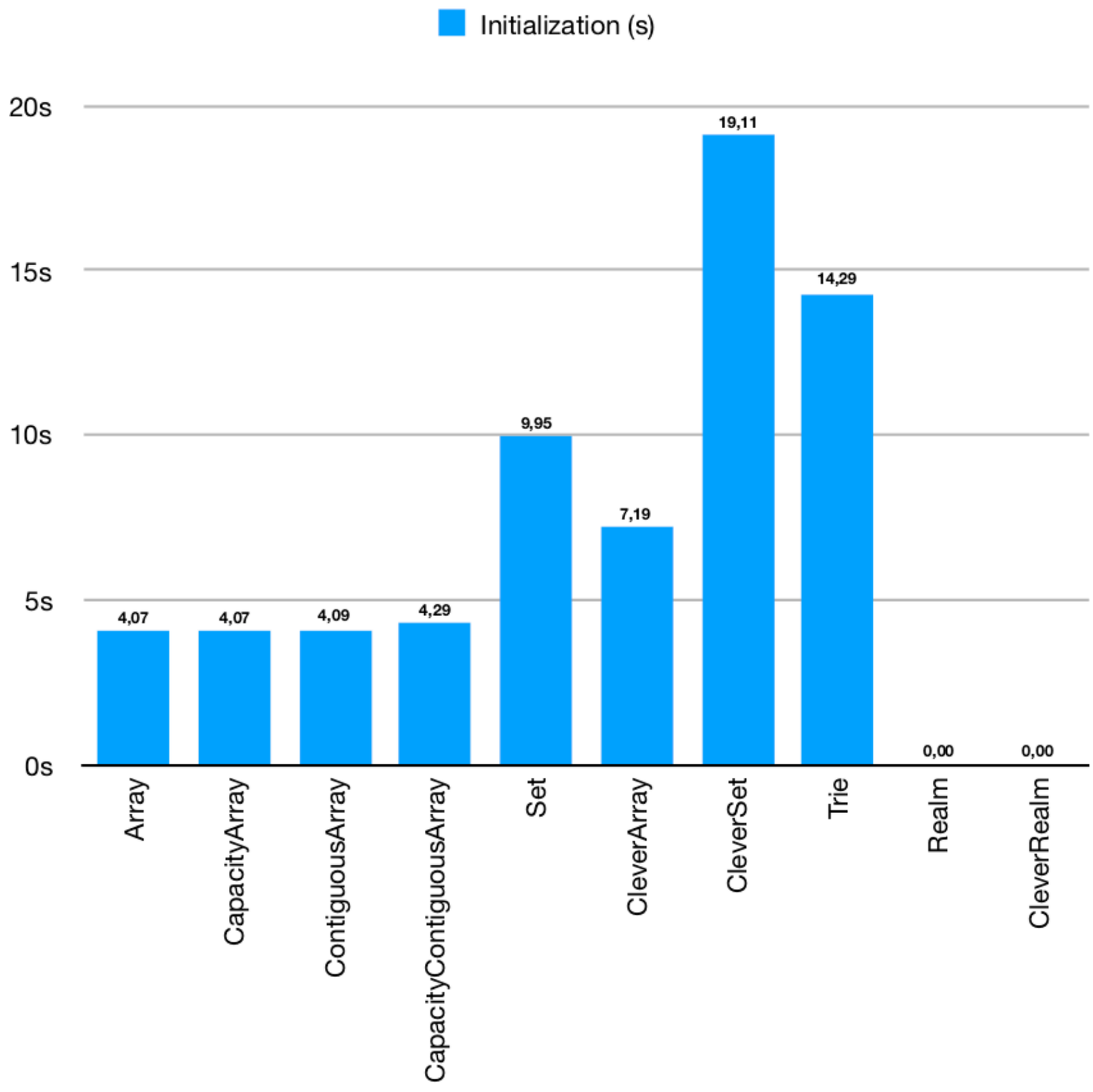
5 LETTERS

abace
abaci
abaka

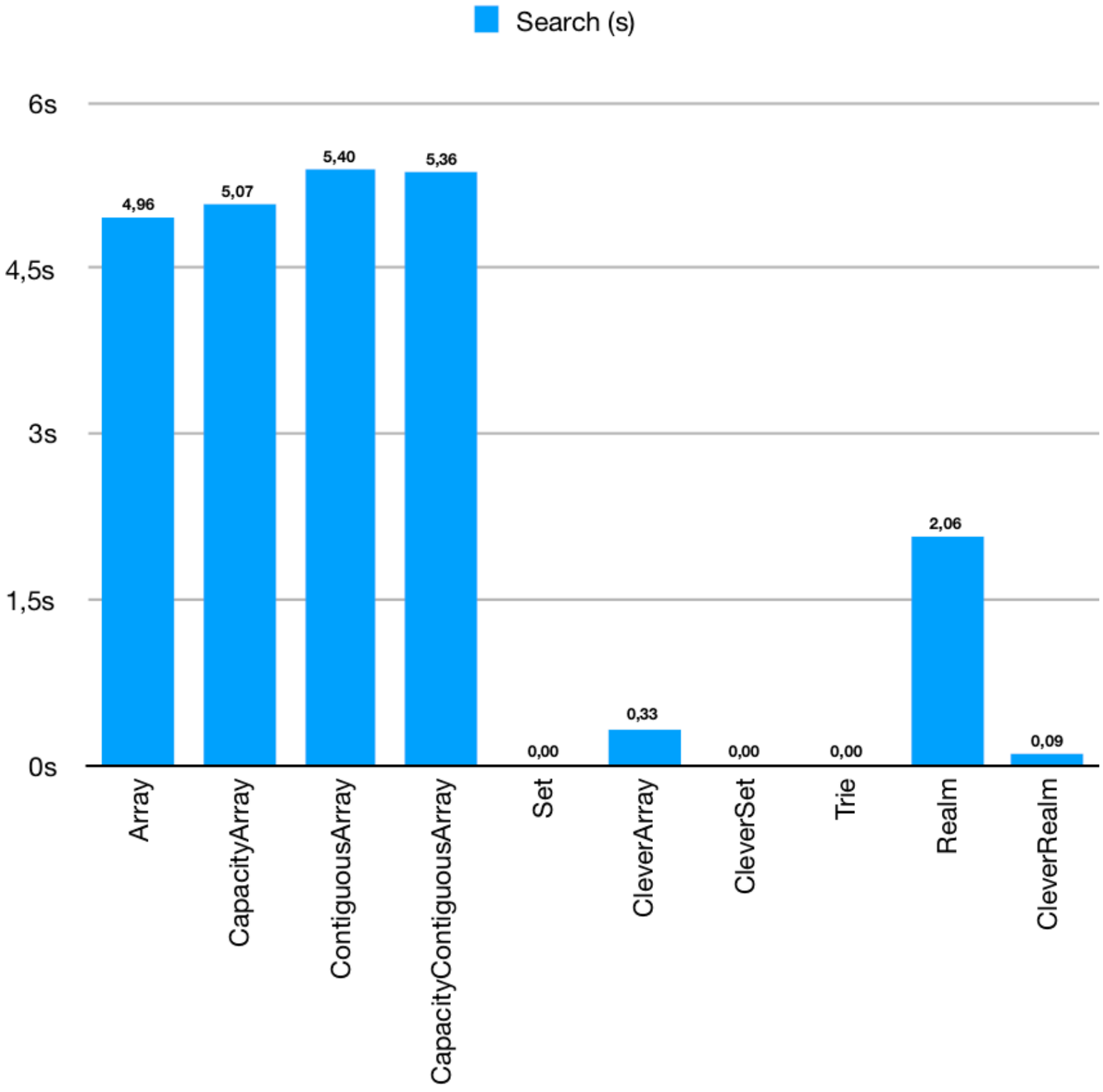
```
protocol Validator: class {
   init()

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

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

SAMPLE VALIDATOR

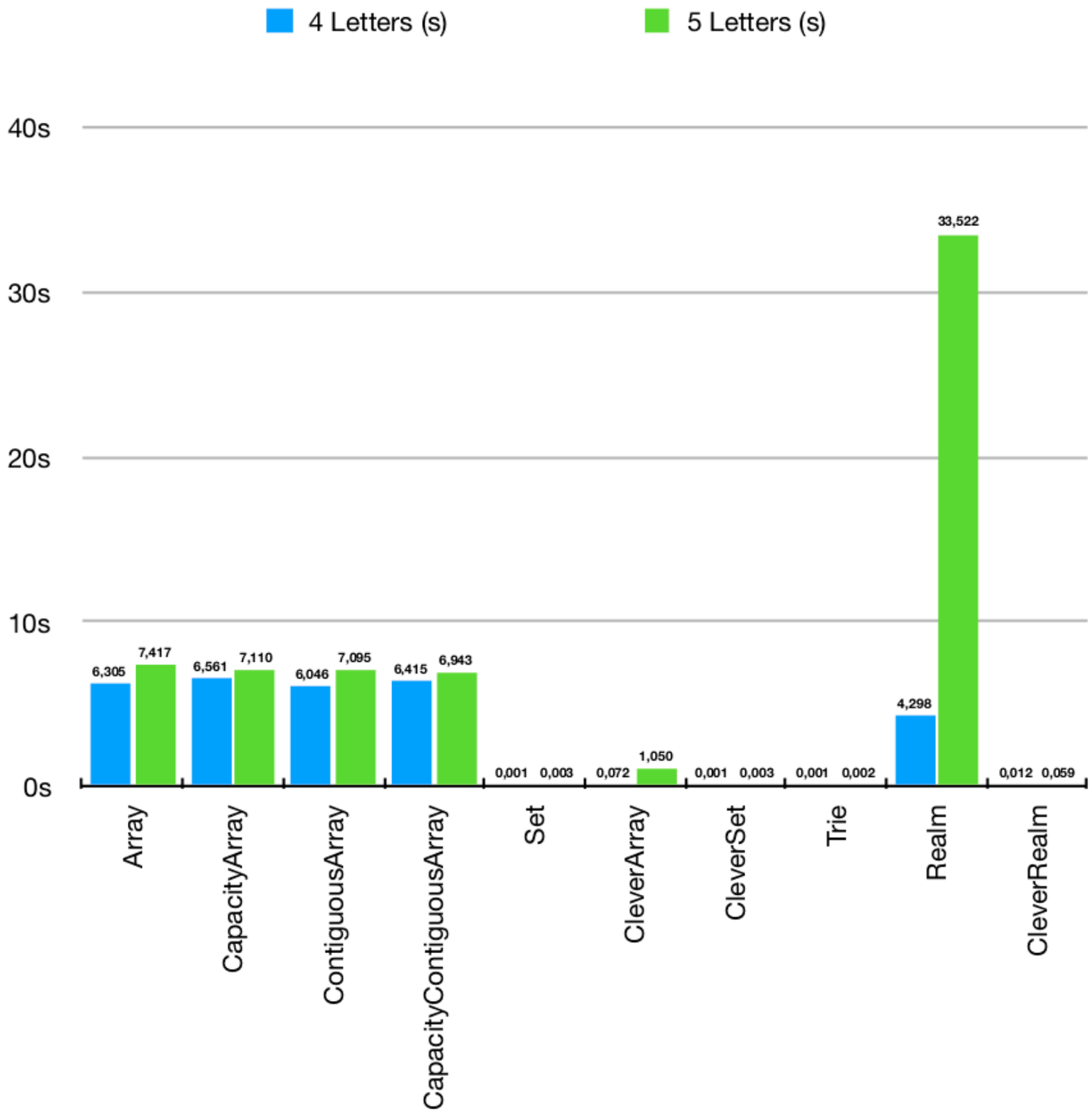
```
final class ArrayWordValidator: Validator {

  /// 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) }
  }
}
```



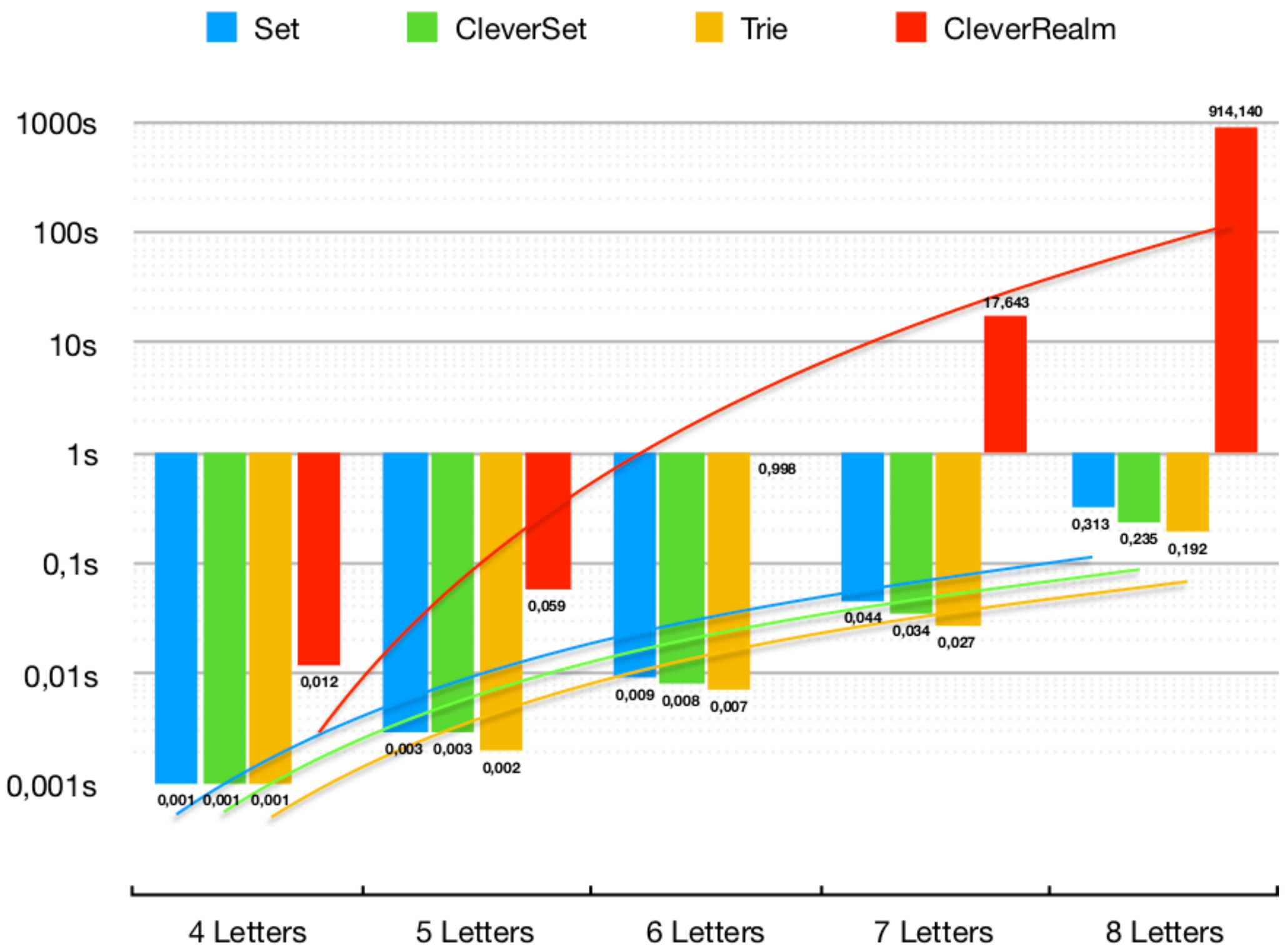
INITIALIZATION TIME



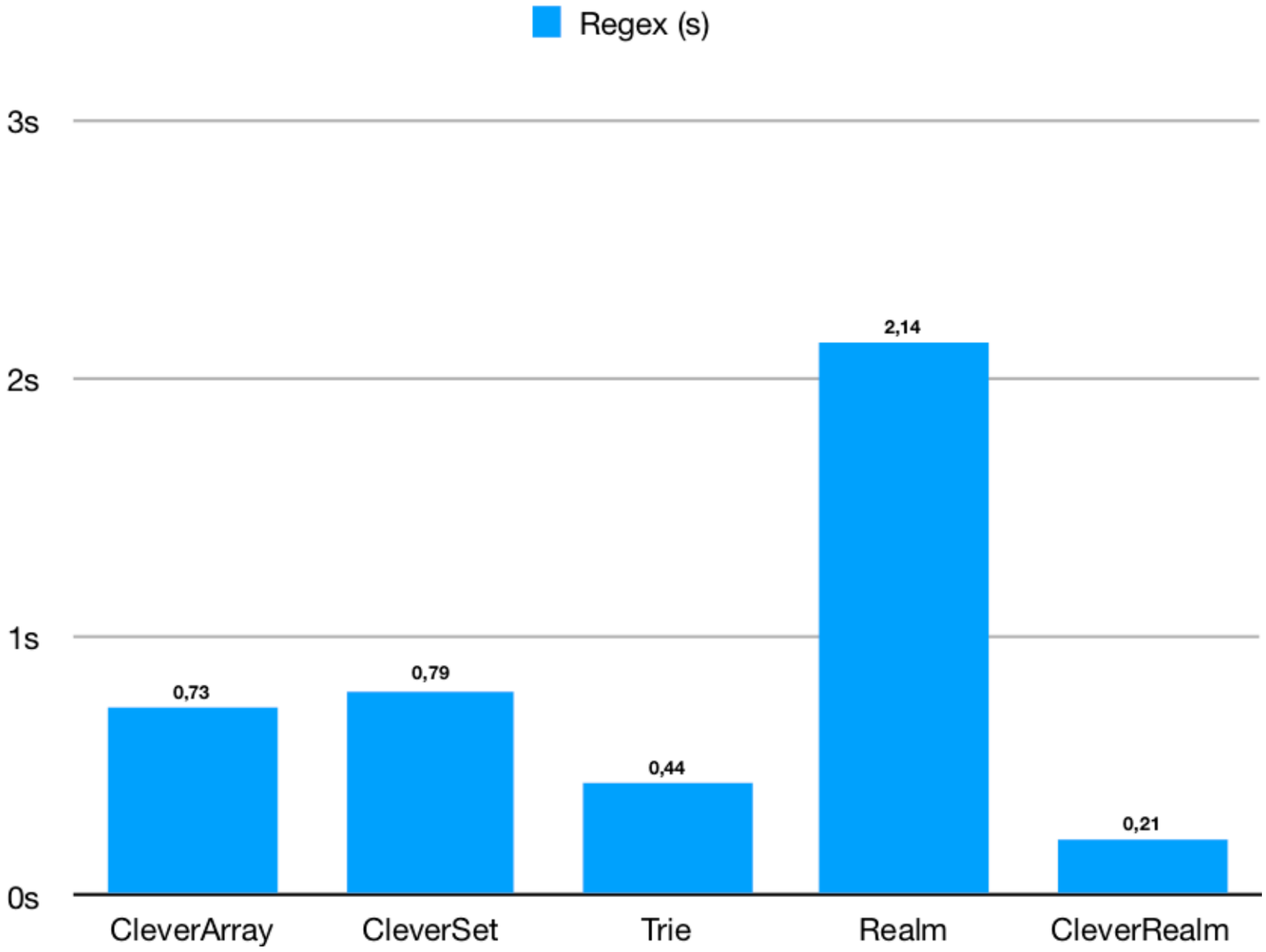
WORD VALIDATION



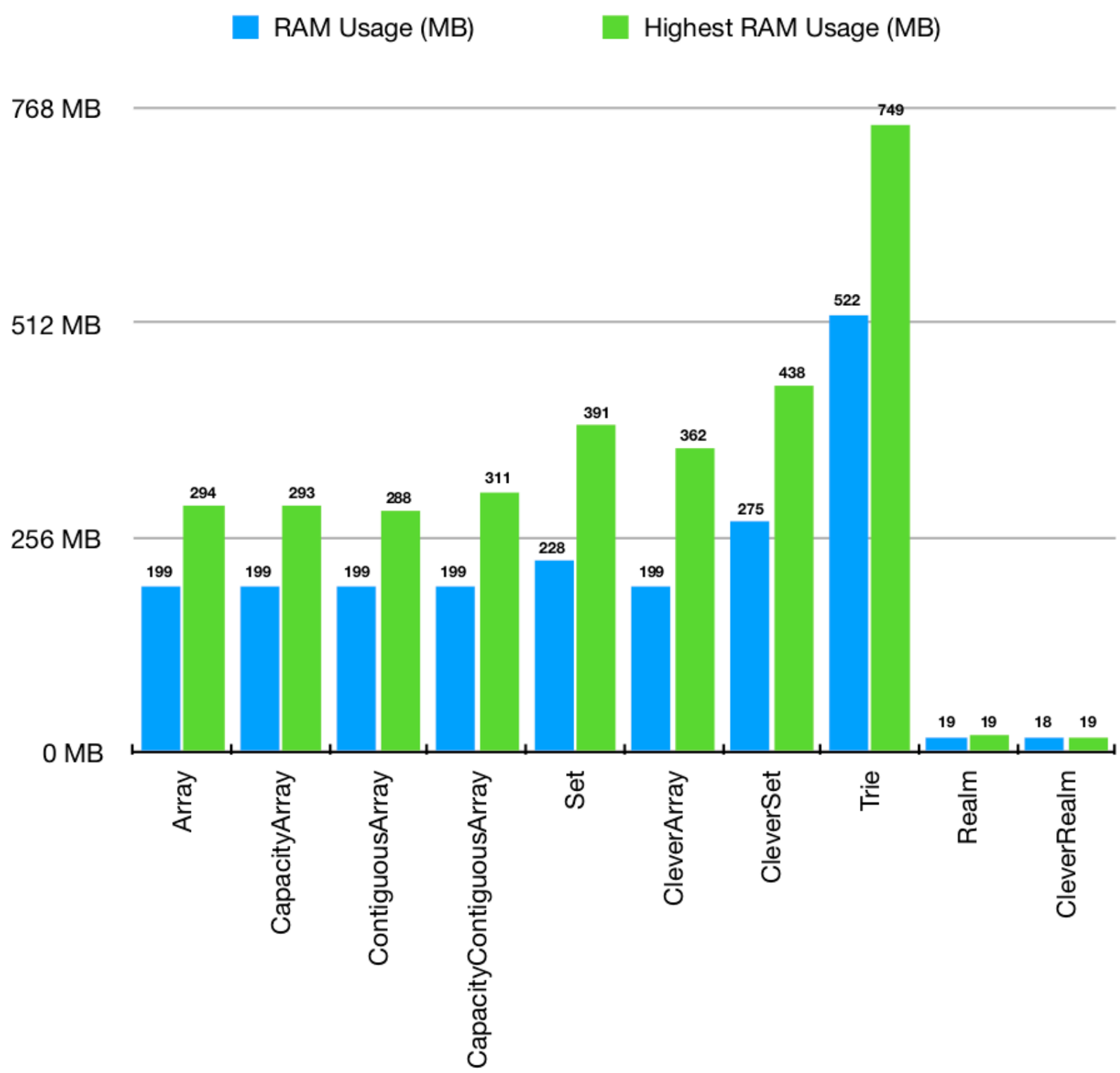
WORDS MADE FROM GIVEN LETTERS



WORDS MADE FROM GIVEN LETTERS

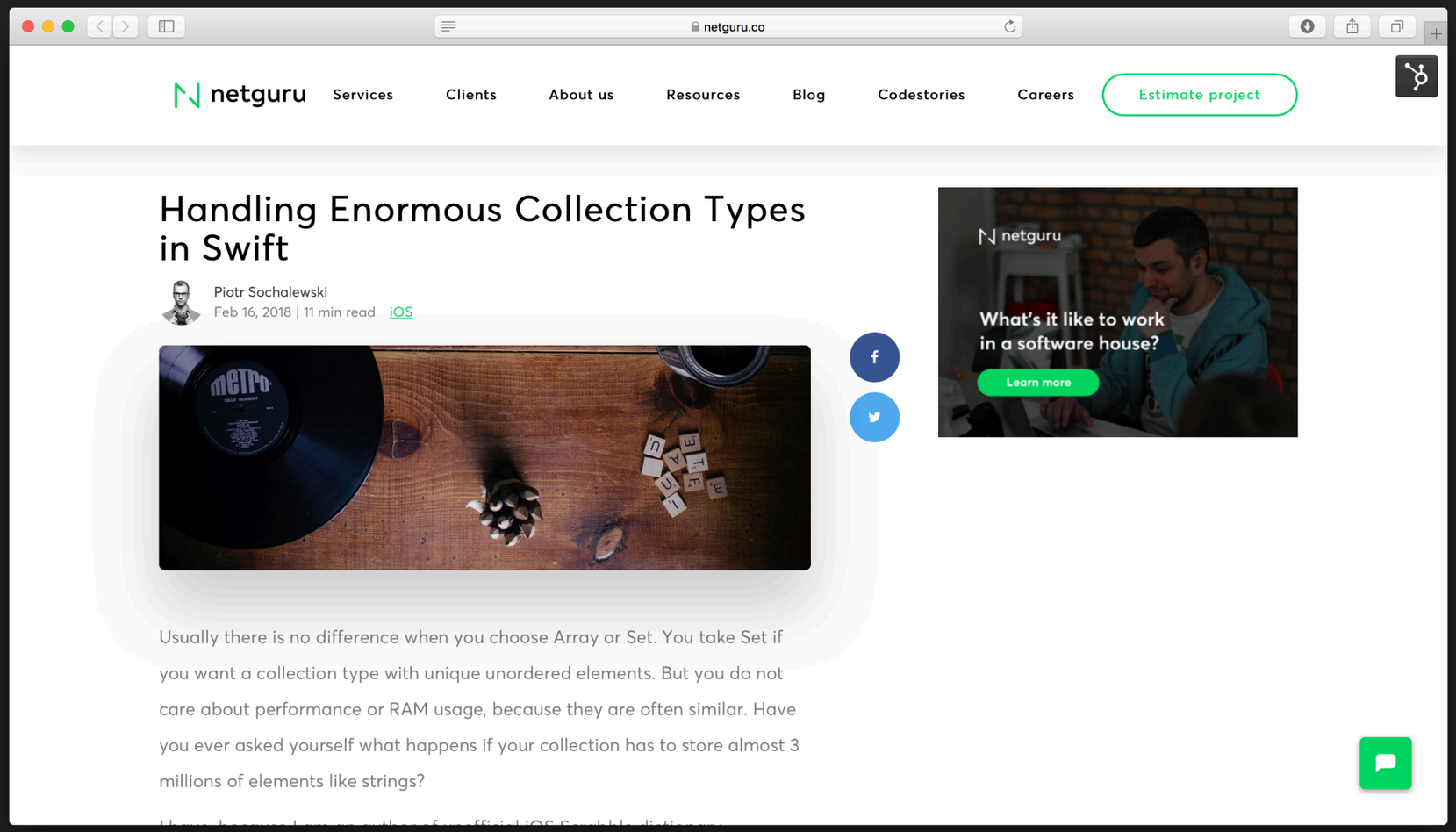


SIMPLE REGEX (BLANKS)


RAM Usage (MB)
Highest RAM Usage (MB)
768 MB
512 MB
256 MB
0 MB
199
294
199
293
199
288
199
311
228
391
199
362
275
438
522
749
19
19
18
19
Array
CapacityArray
ContiguousArray
CapacityContiguousArray
Set
CleverArray
CleverSet
Trie
Realm
CleverRealm

RAM USAGE

REALM + TRIE = ❤️



Piotr Sochalewski

iOS Developer w **Netguru**

piotr.sochalewski@netguru.co