Algorithms, where no developer wants to go.

DANIJELA VRZAN

319. Bulb Switcher

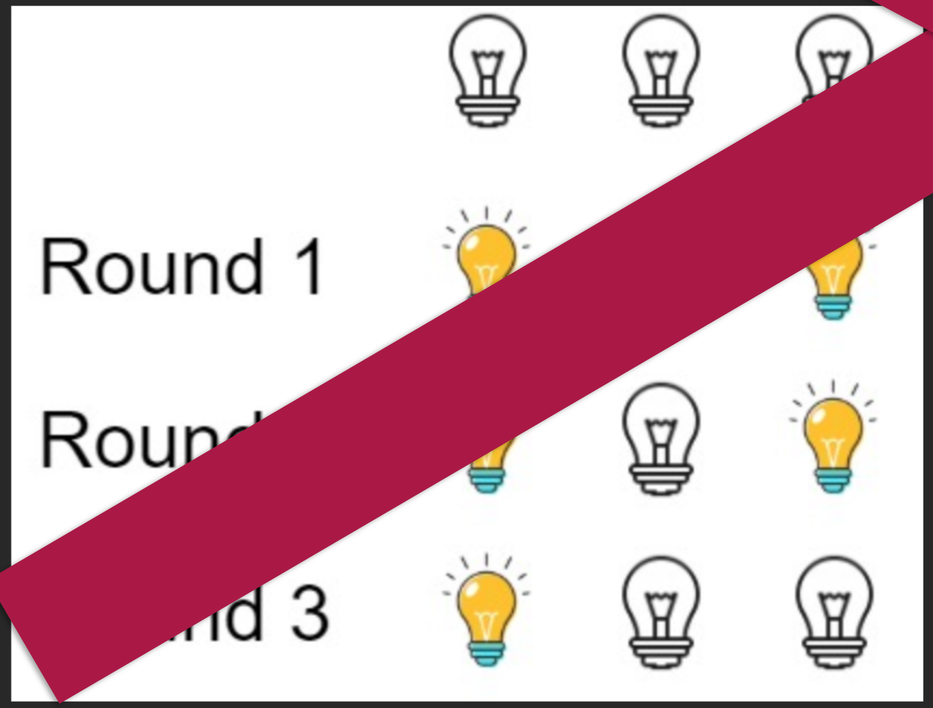
dium 1.3K 1.9K

There initially off. You first turn on all the bulbs, then you turn off every second bulb.

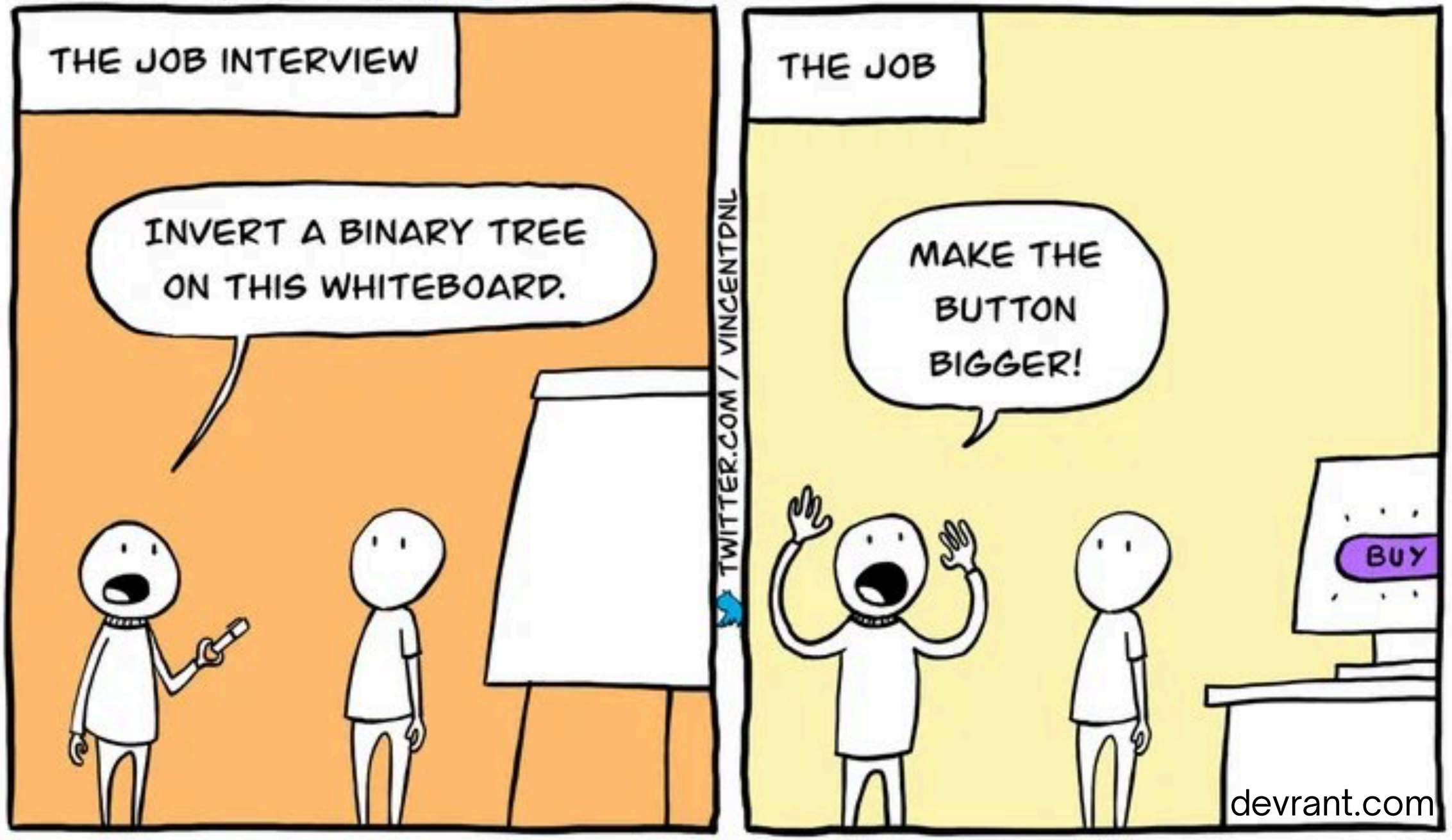
On the third rou hird bulb (turning on if it's off or turning off if it's on). For the i^{th} round. bulb. For the n^{th} round, you only toggle the last bu

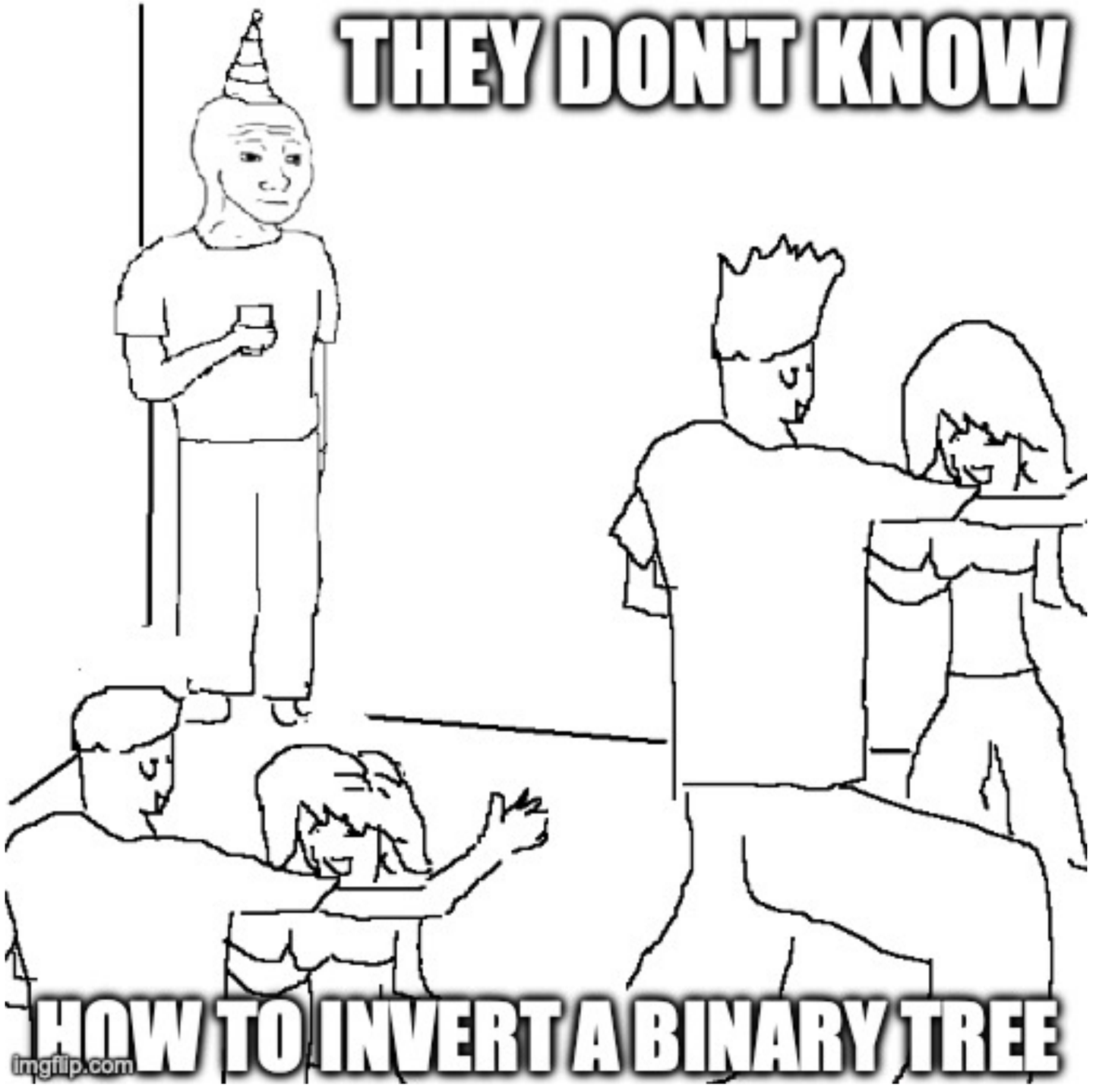
Return *the number of bulbs that a*

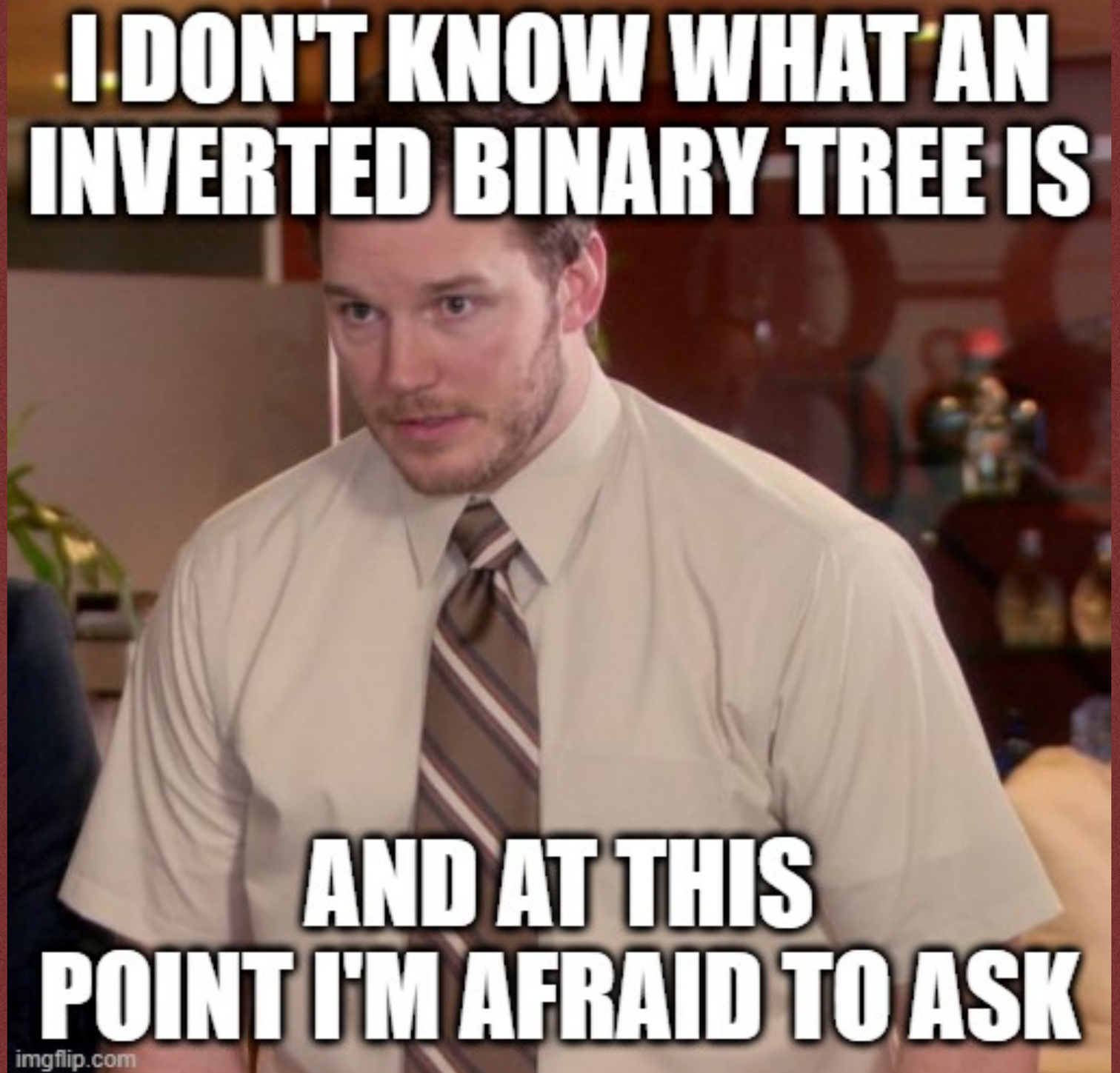
Example 1:



JOBS REQUIREMENTS




THEY DON'T KNOW
HOW TO INVERT A BINARY TREE
imgflip.com


I DON'T KNOW WHAT AN INVERTED BINARY TREE IS
AND AT THIS POINT I'M AFRAID TO ASK
imgflip.com

Swift Algorithms

Open-source package of sequence and collection algorithms

https://github.com/apple/swift-algorithms

main | 11 branches | 9 tags | Go to file | Add file | Code

dickoff Merge pull request #191 from dickoff/dickoff/fix-max-code-docume... 288d8b0 on Dec 7, 2022 151 commits

.github	Prepare for Swift Algorithms 1.0 (#167)	2 years ago
Guides	Add `partitioned(by:)` (#152)	2 years ago
Sources/Algorithms	Corrects variable name in max functions to better reflect usage	5 months ago
Tests	Clarify `endIndex` behavior of `partitioningIndex(where:)` in doc...	2 years ago
.gitignore	Ignore Package.resolved	3 years ago
CHANGELOG.md	Prepare for Swift Algorithms 1.0 (#167)	2 years ago
CODE_OF_CONDUCT.md	Initial import	3 years ago
CONTRIBUTING.md	Initial import	3 years ago
LICENSE.txt	Initial import	3 years ago
Package.swift	Prepare for Swift Algorithms 1.0 (#167)	2 years ago
README.md	Add `partitioned(by:)` (#152)	2 years ago

README.md

Swift Algorithms

Swift Algorithms is an open-source package of sequence and collection algorithms, along with their related types.

Read more about the package, and the intent behind it, in the announcement on swift.org.

Contents

About

Commonly used sequence and collection algorithms for Swift

algorithm | iterator | itertools

Readme
Apache-2.0 license
Code of conduct
5k stars
235 watching
369 forks
Report repository

Releases 9

Swift Algorithms 1.0.0 Latest
on Sep 8, 2021

+ 8 releases

Packages

No packages published

Used by 1

@akirataguchi115 / akirataguchi115

Contributors 36

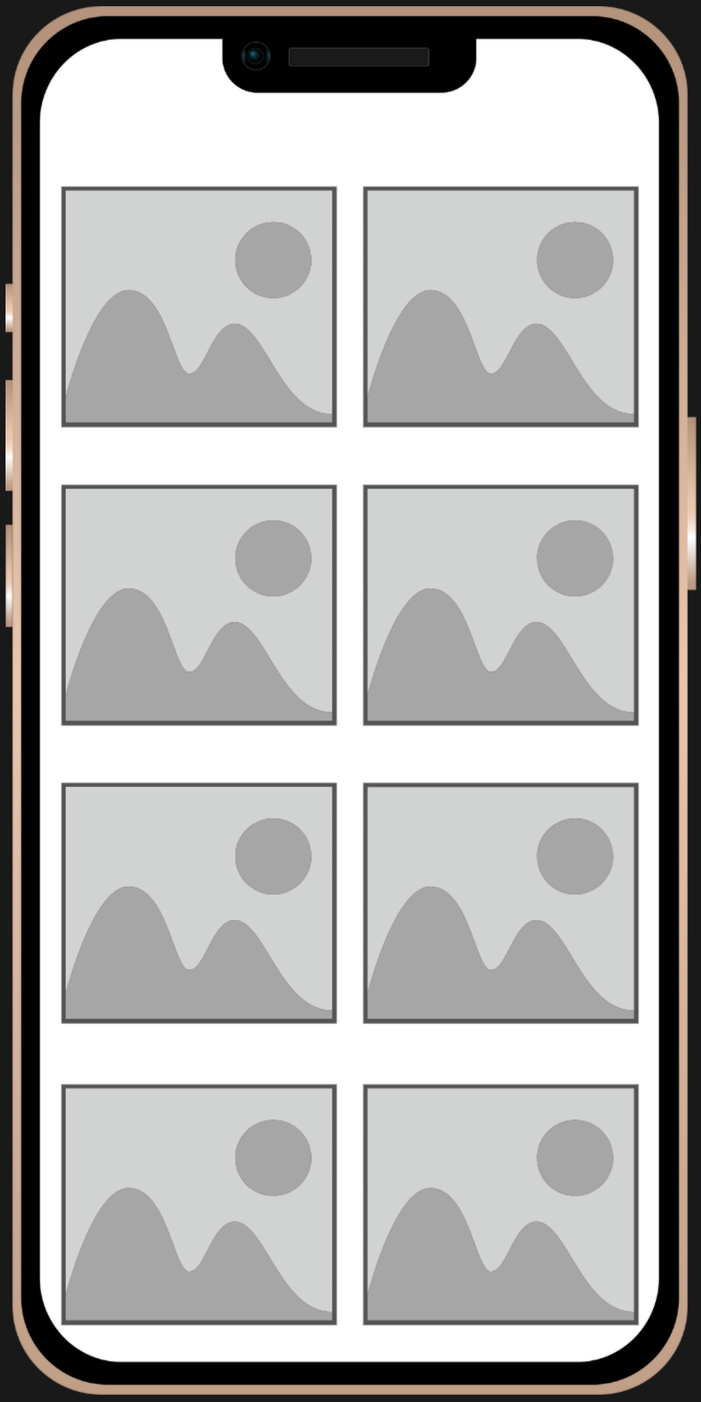
Engage the community earlier in the development process

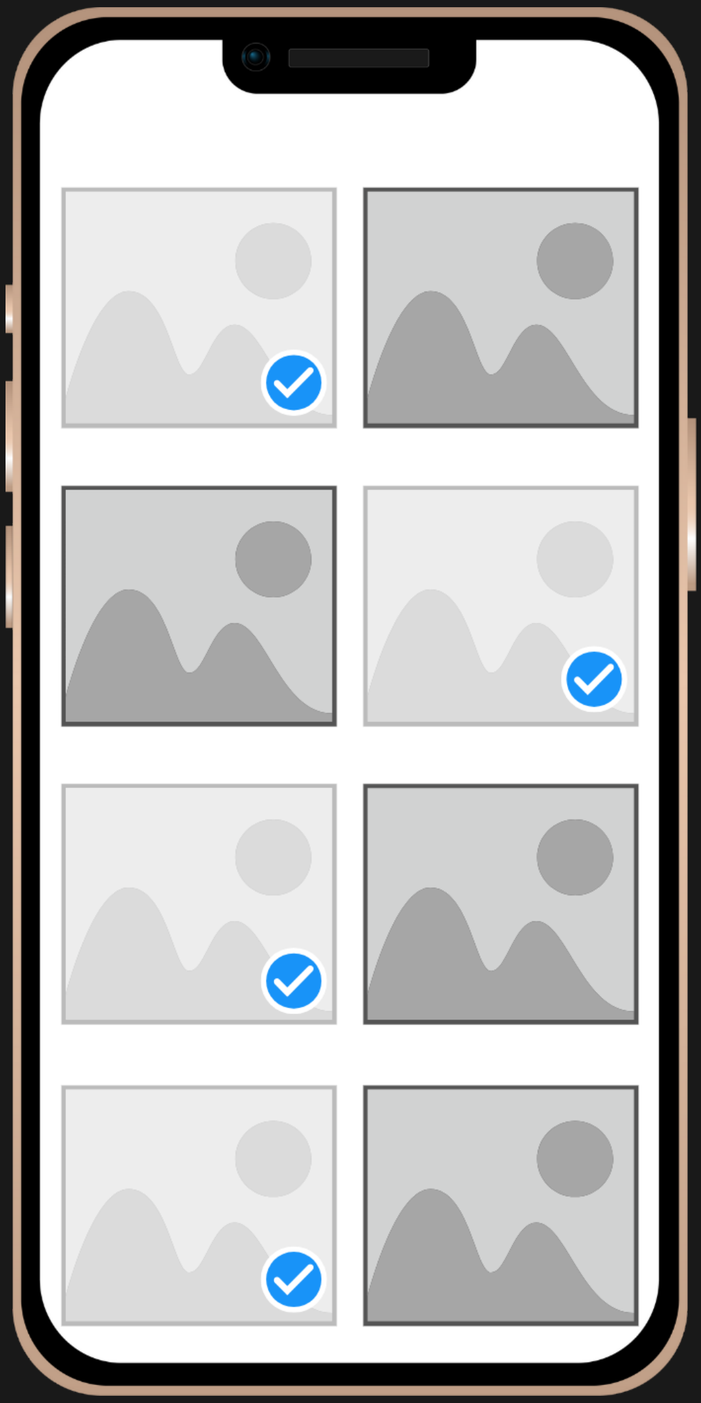
Seek feedback informed by real-world usage

Swift standard library has a rich collection of built-in algorithms

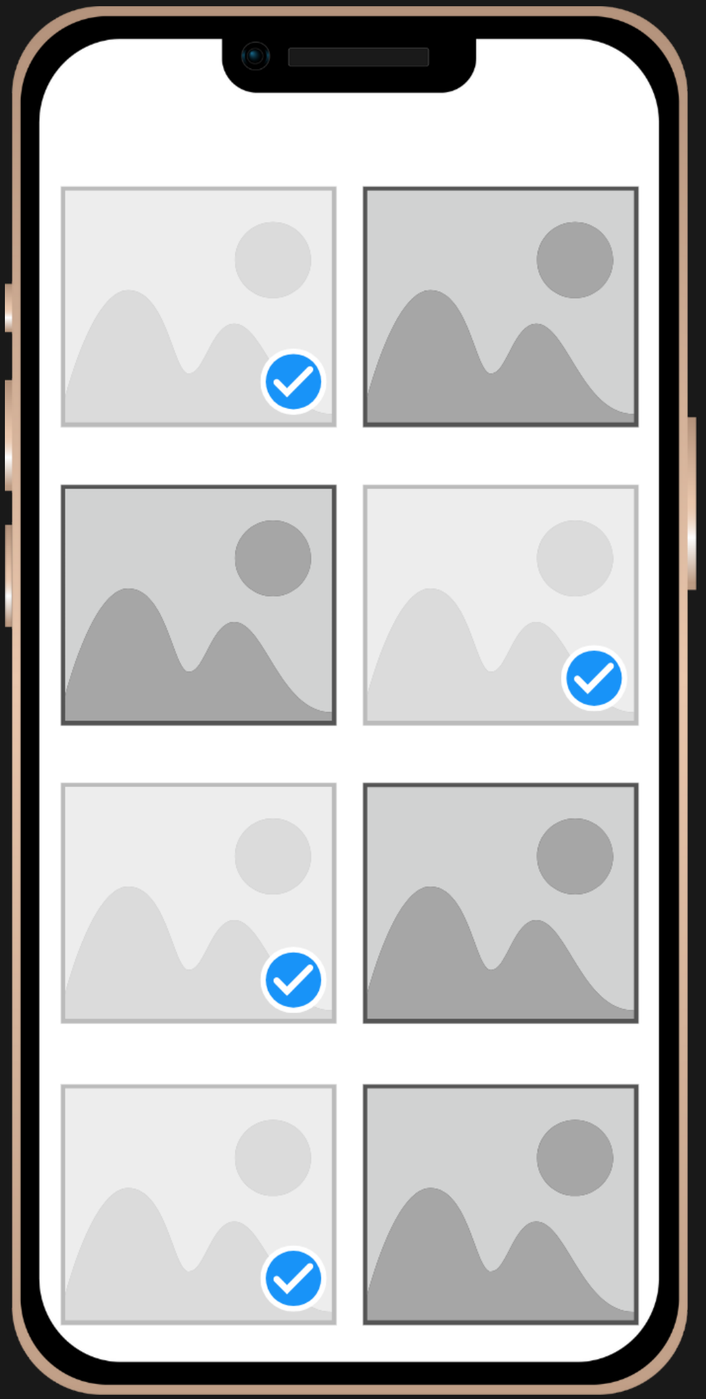
Fast

Concise



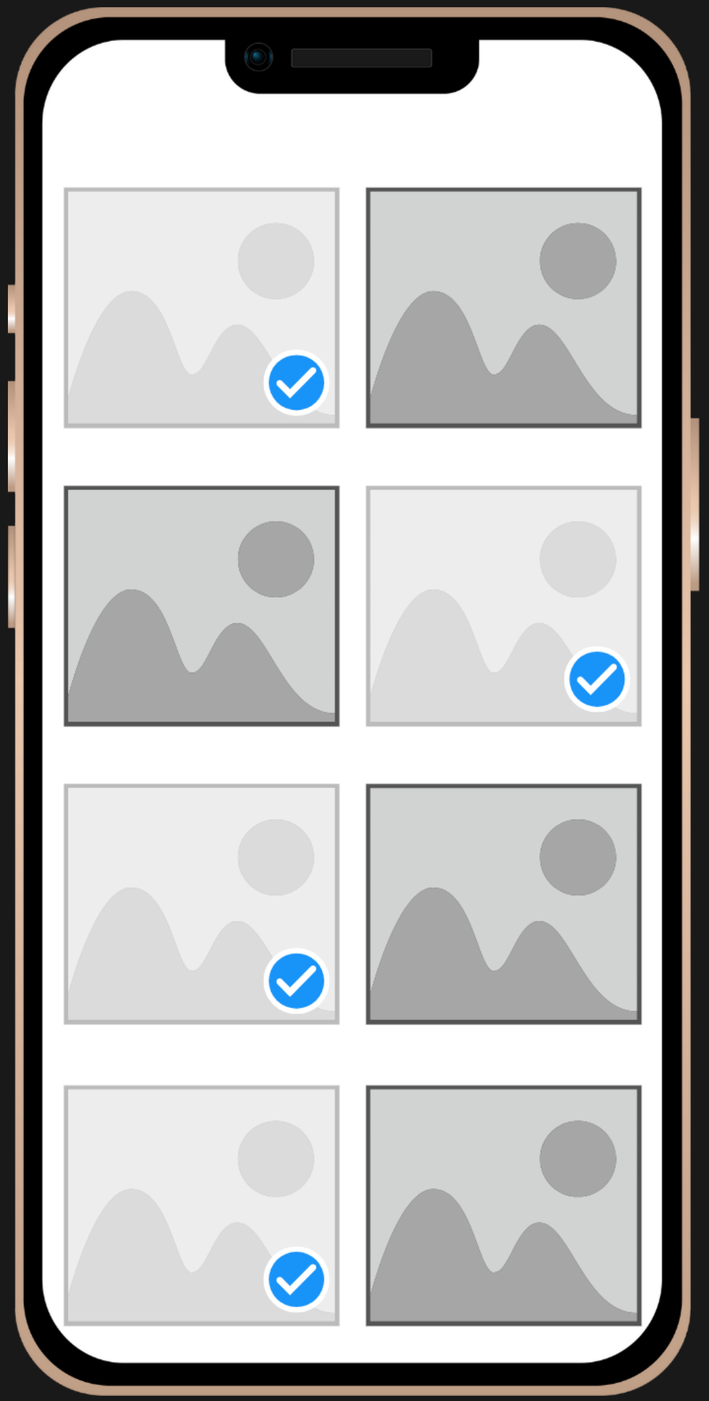


```
var selectedImages = [Image]()

for indexPath in indexPathsForSelectedItems {
    selectedImages.append(images[indexPath.item])
}
```



```
var selectedImages = [Image]()

for indexPath in indexPathsForSelectedItems {
    selectedImages.append(images[indexPath.item])
}
```

```
indexPathsForSelectedItems.map { images[$0.row] }
```



```
var numbers = [
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
    39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
    57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
    75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
    93, 94, 95, 96, 97, 98, 99, 100
]
var squares = [Int]()
```

```
var numbers = [
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
    39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
    57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
    75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
    93, 94, 95, 96, 97, 98, 99, 100
]
var squares = [Int]()
```

```
func withForLoop() {
    for number in numbers {
        let square = number * number
        squares.append(square)
    }
}

func calculateTime() -> CFAbsoluteTime {
    let start = CFAbsoluteTimeGetCurrent()

    withForLoop()

    return CFAbsoluteTimeGetCurrent() - start
}
print("For loop took \(calculateTime()) seconds")
```

```
var numbers = [
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
    39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
    57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
    75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
    93, 94, 95, 96, 97, 98, 99, 100
]
var squares = [Int]()
```

For loop took 5.1975250244140625e-05 seconds

For loop took 5.0067901611328125e-05 seconds

For loop took 5.5909156799316406e-05 seconds

For loop took 4.9948692321777344e-05 seconds

~~~ 5 sec 🙈

~ 0.00005 sec
~~~

```
var numbers = [
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
    39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
    57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
    75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
    93, 94, 95, 96, 97, 98, 99, 100
]
var squares = [Int]()
```

```
func withMap() {
    var squares = numbers.map { $0 * $0 }
}

func calculateTime() -> CFAbsoluteTime {
    let start = CFAbsoluteTimeGetCurrent()

    withMap()

    return CFAbsoluteTimeGetCurrent() - start
}

print("Map took \(calculateTime()) seconds")
```

```
var numbers = [
    1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
    39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
    57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
    75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
    93, 94, 95, 96, 97, 98, 99, 100
]
var squares = [Int]()
```

```
Map took 3.3020973205566406e-05 seconds

Map took 3.0994415283203125e-05 seconds

Map took 2.9921531677246094e-05 seconds

Map took 3.0994415283203125e-05 seconds
```

~~~ 3 sec 🙈

~ 0.00003 sec
~~~

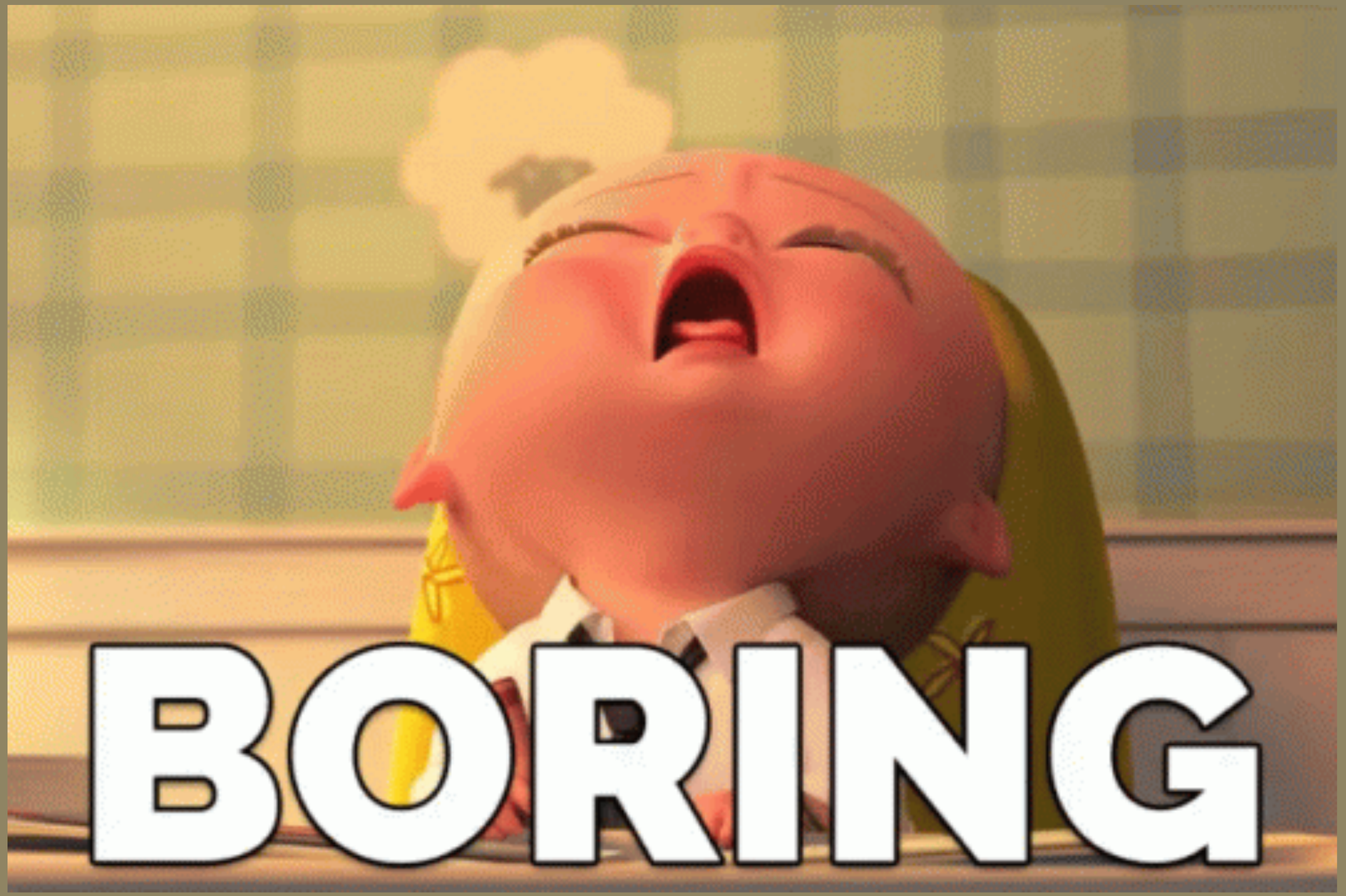
~ 0.00003 sec

VS

~ 0.00005 sec

Use .map() to make your code faster

Use algorithms to make your code faster


BORING

Fast

Concise

Swift standard library has a rich collection of built-in algorithms

.filter()

.joined()

.map()

.compactMap()

.flatMap()

.compactMap() = .map() + .filter()

.joined()

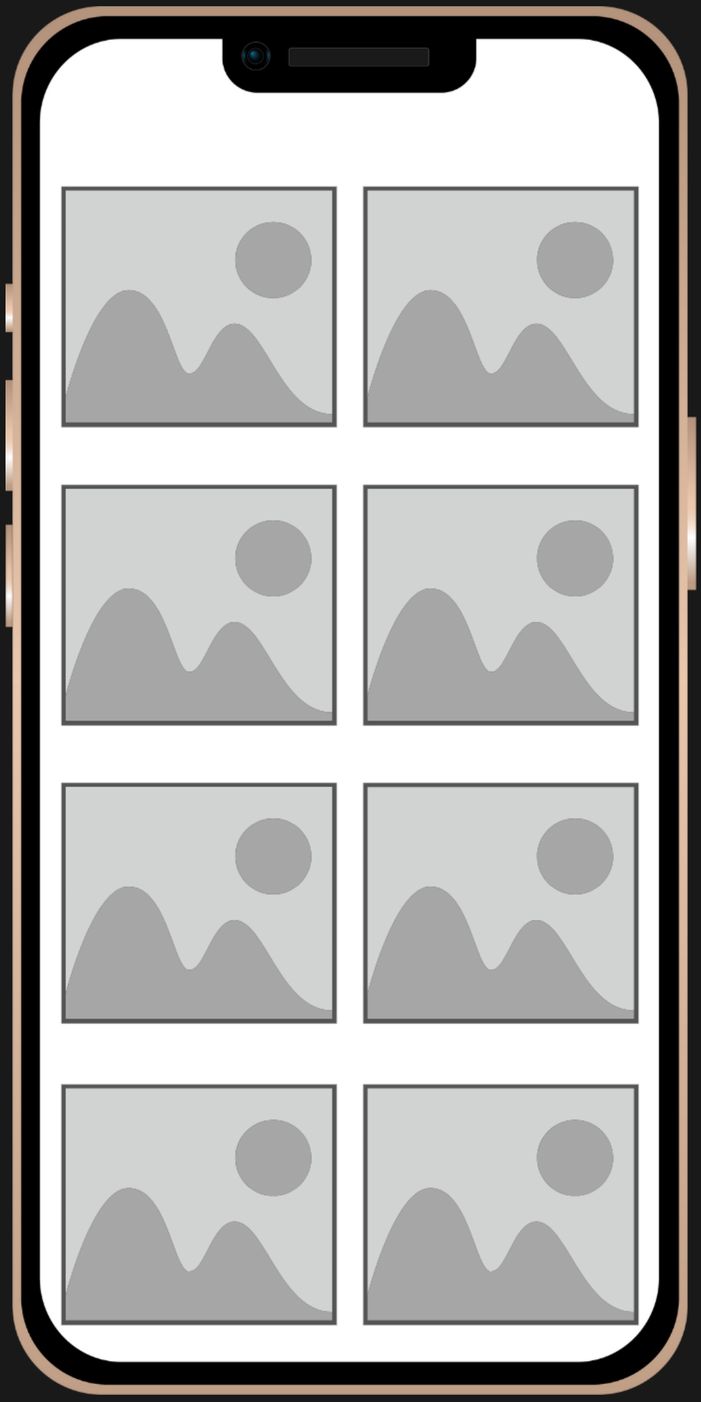
.flatMap()

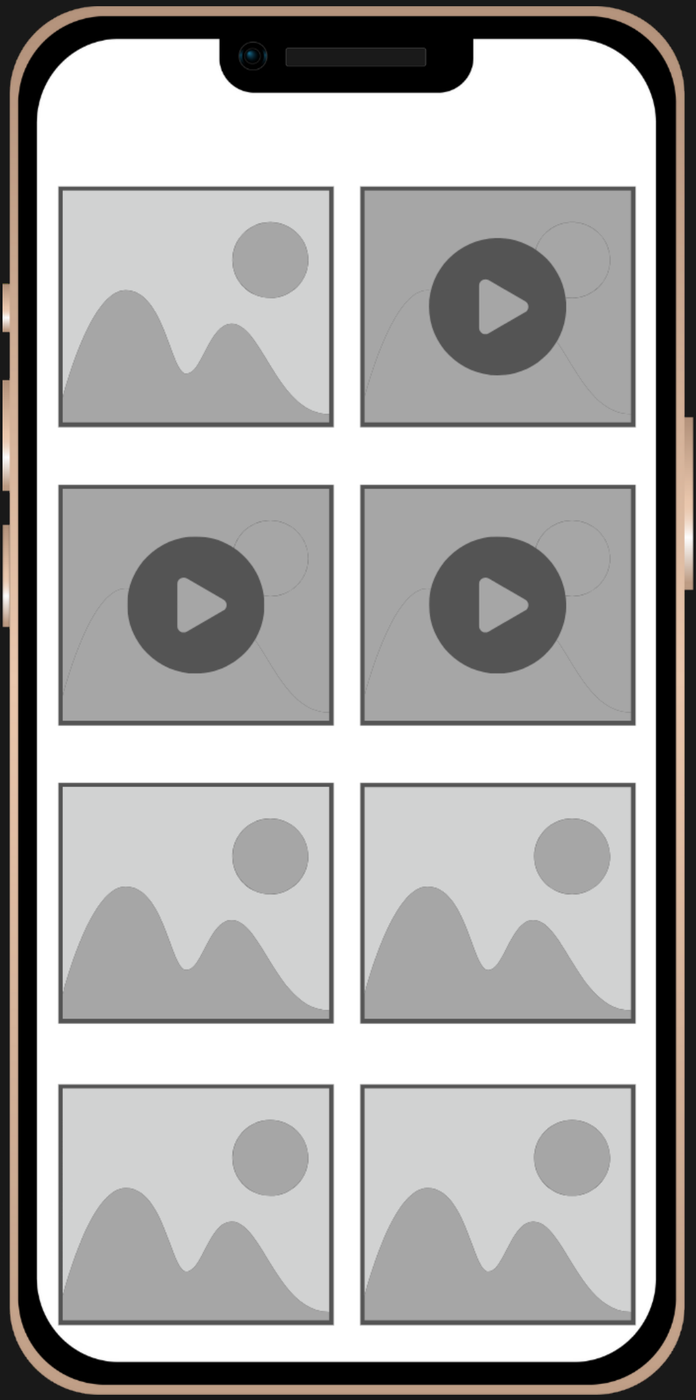
.compactMap() = .map() + .filter()

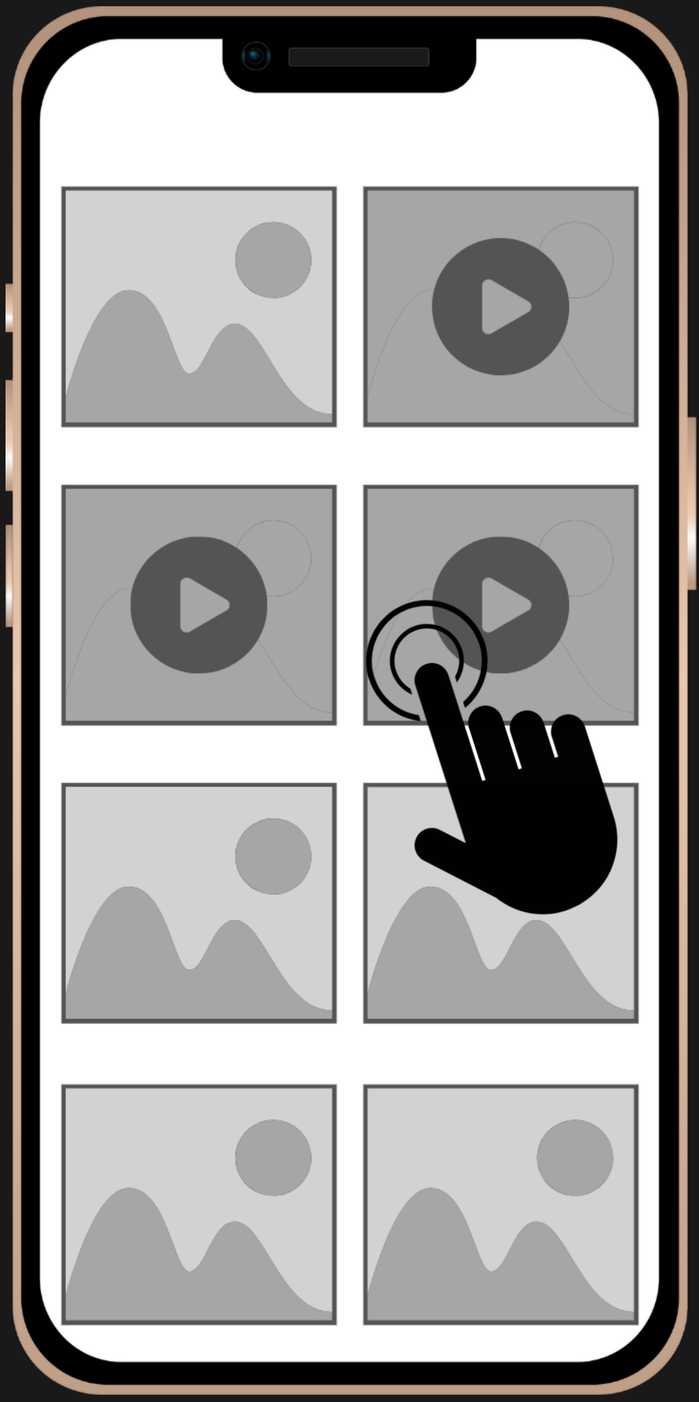
.flatMap() = .map() + .joined()

.compactMap() = .map() + .filter()

.flatMap() = .map() + .joined()



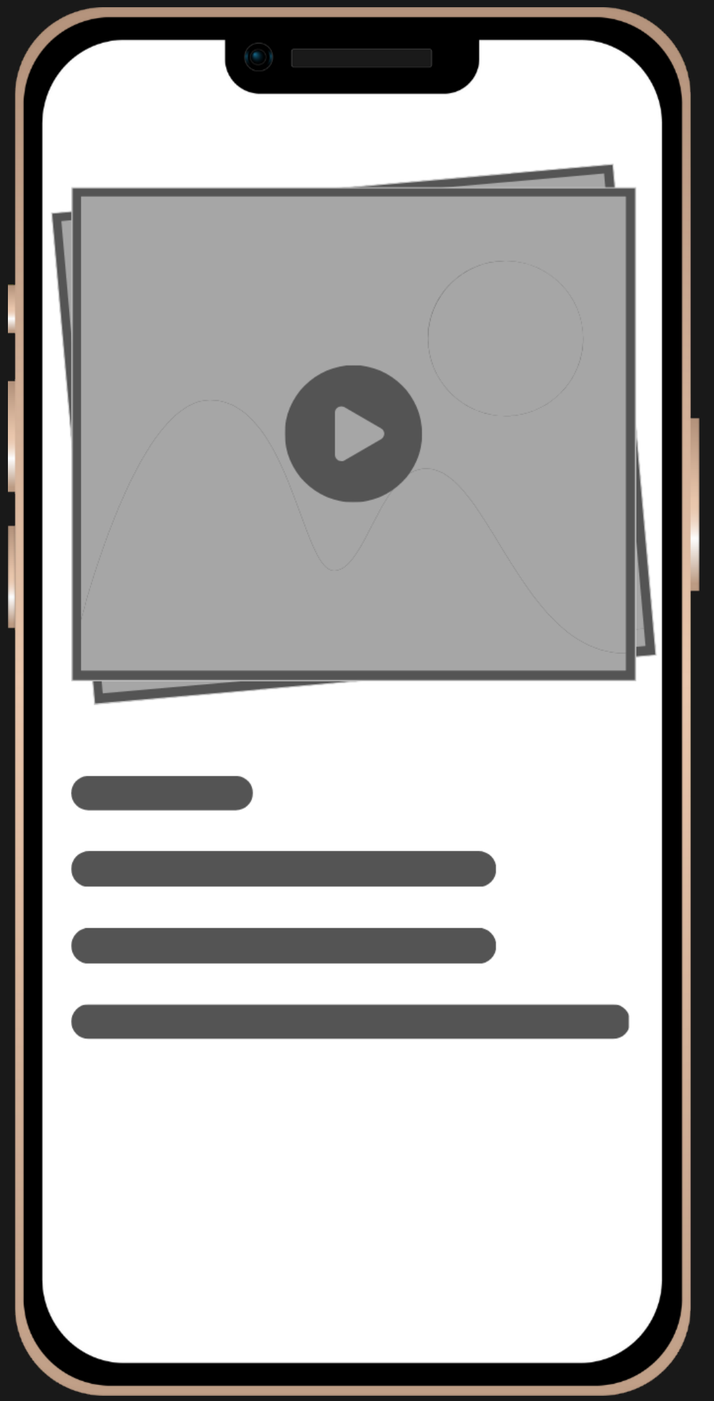




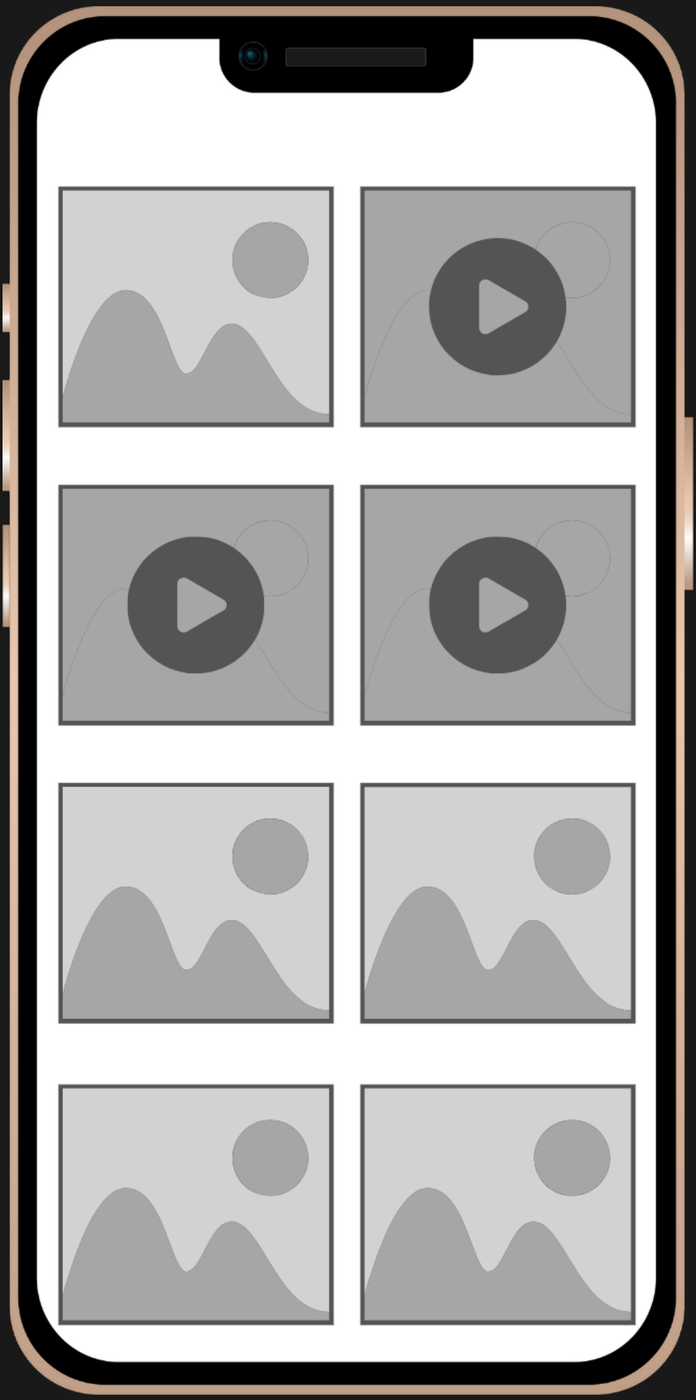
```swift
var videos: [Video] = []

for item in media.reversed() {
    if let video = item as? Video {
        videos.append(video)
        if videos.count == 5 {
            break
        }
    }
}
```

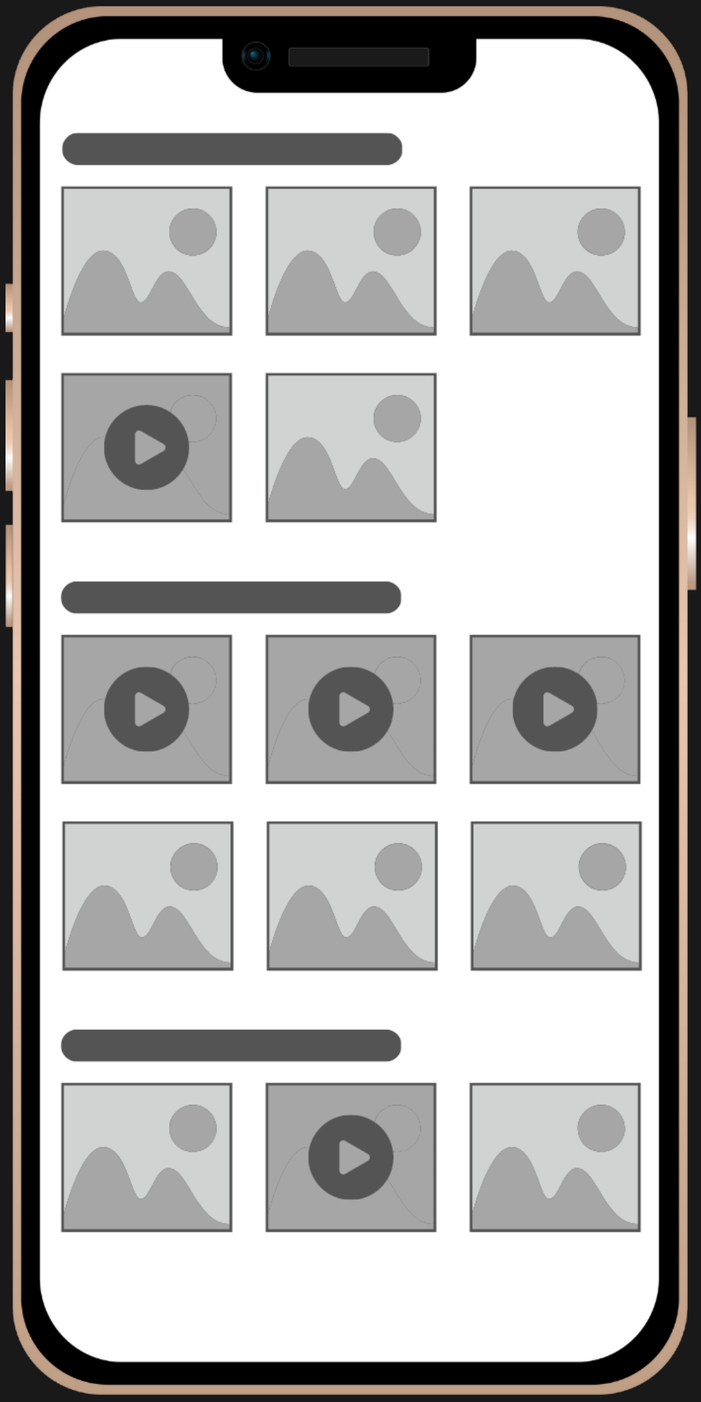
```swift
media
  .reversed()
  .compactMap { $0 as? Video }
  .prefix(5)
```

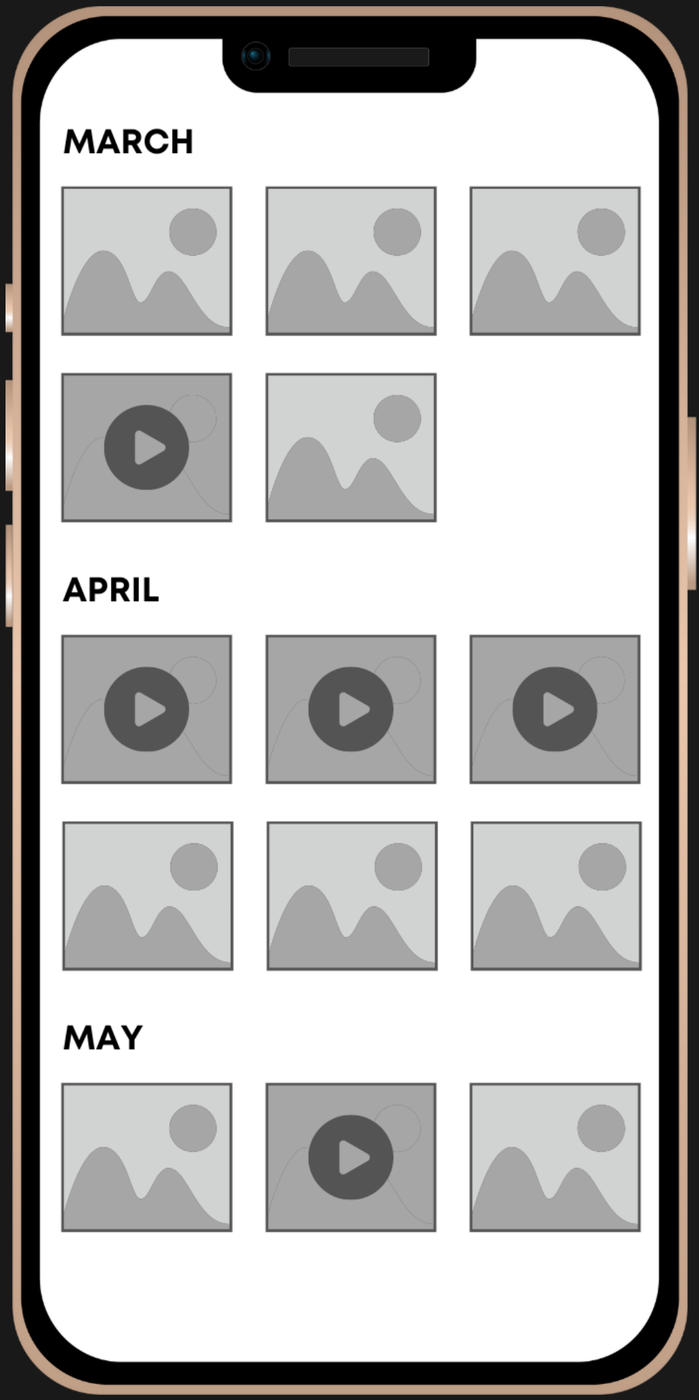
```
media
    .reversed()
    .compactMap { $0 as? Video }
    .prefix(5)
```



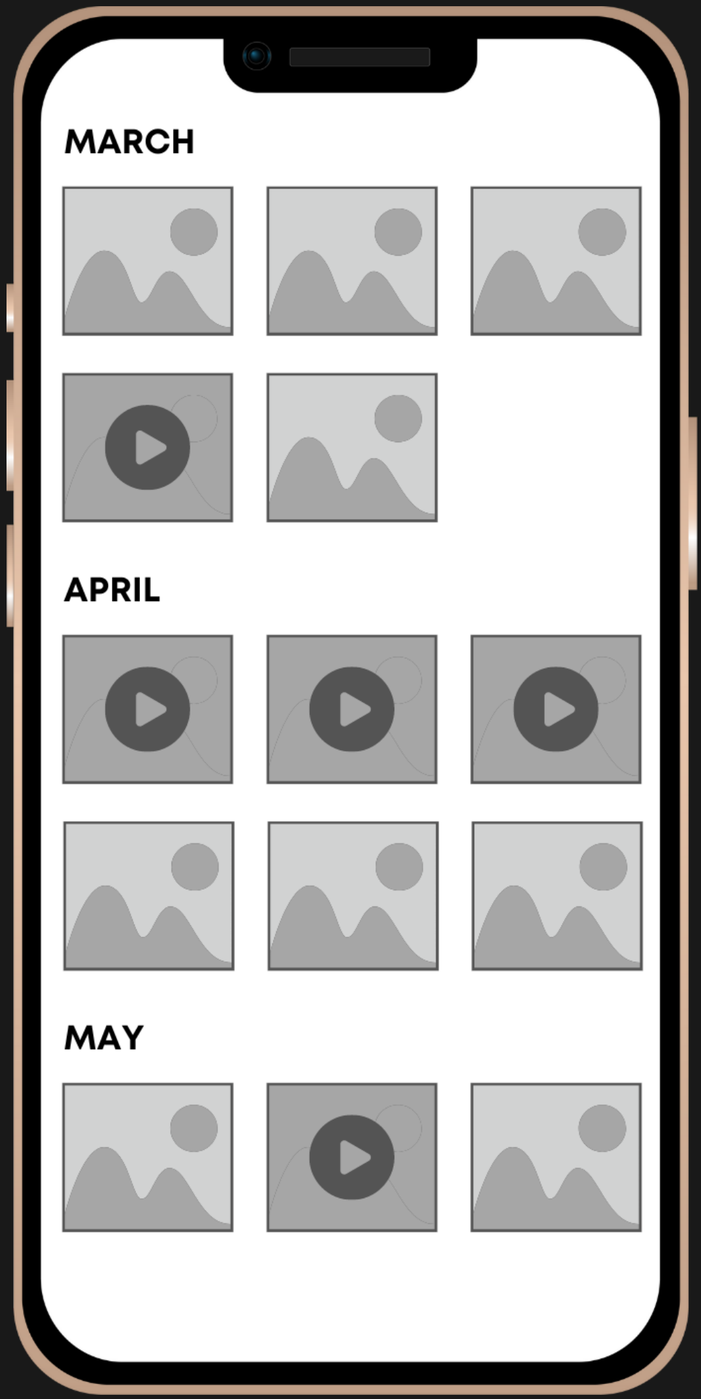
Use algorithms to make your code concise





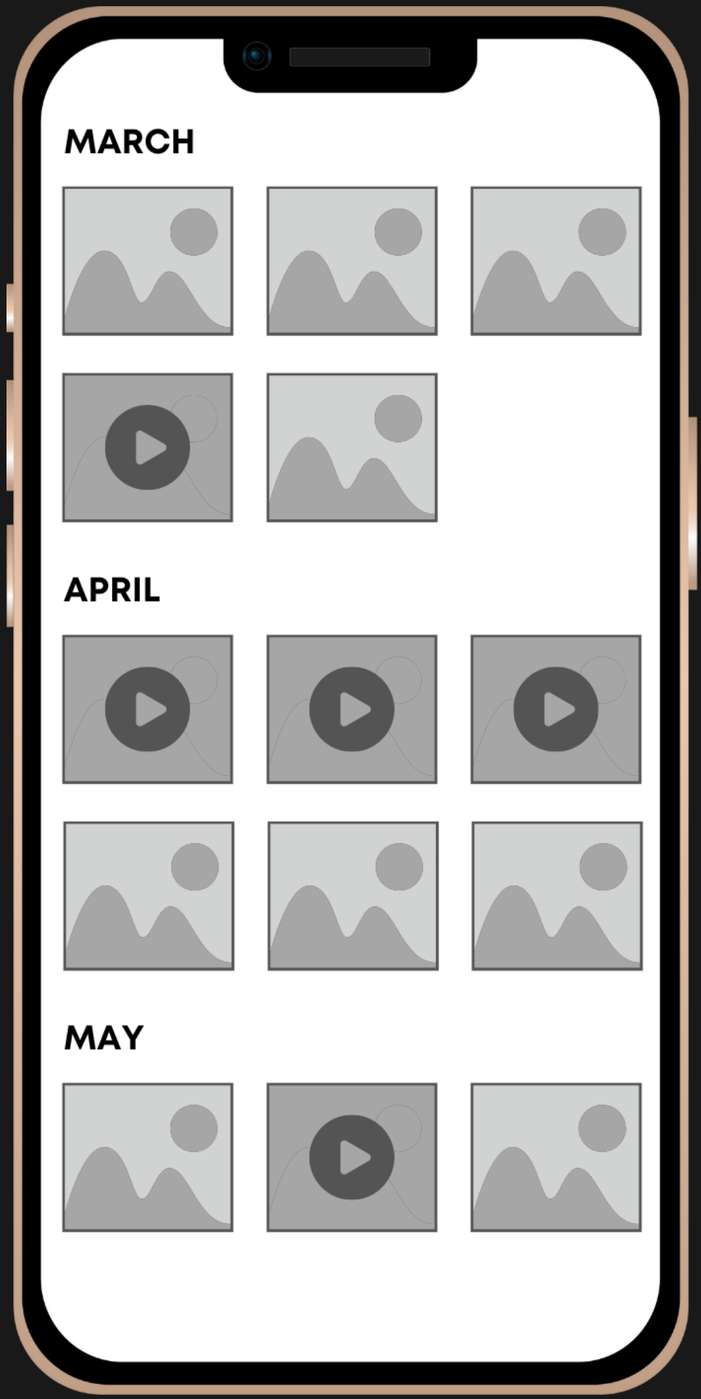

MARCH
APRIL
MAY

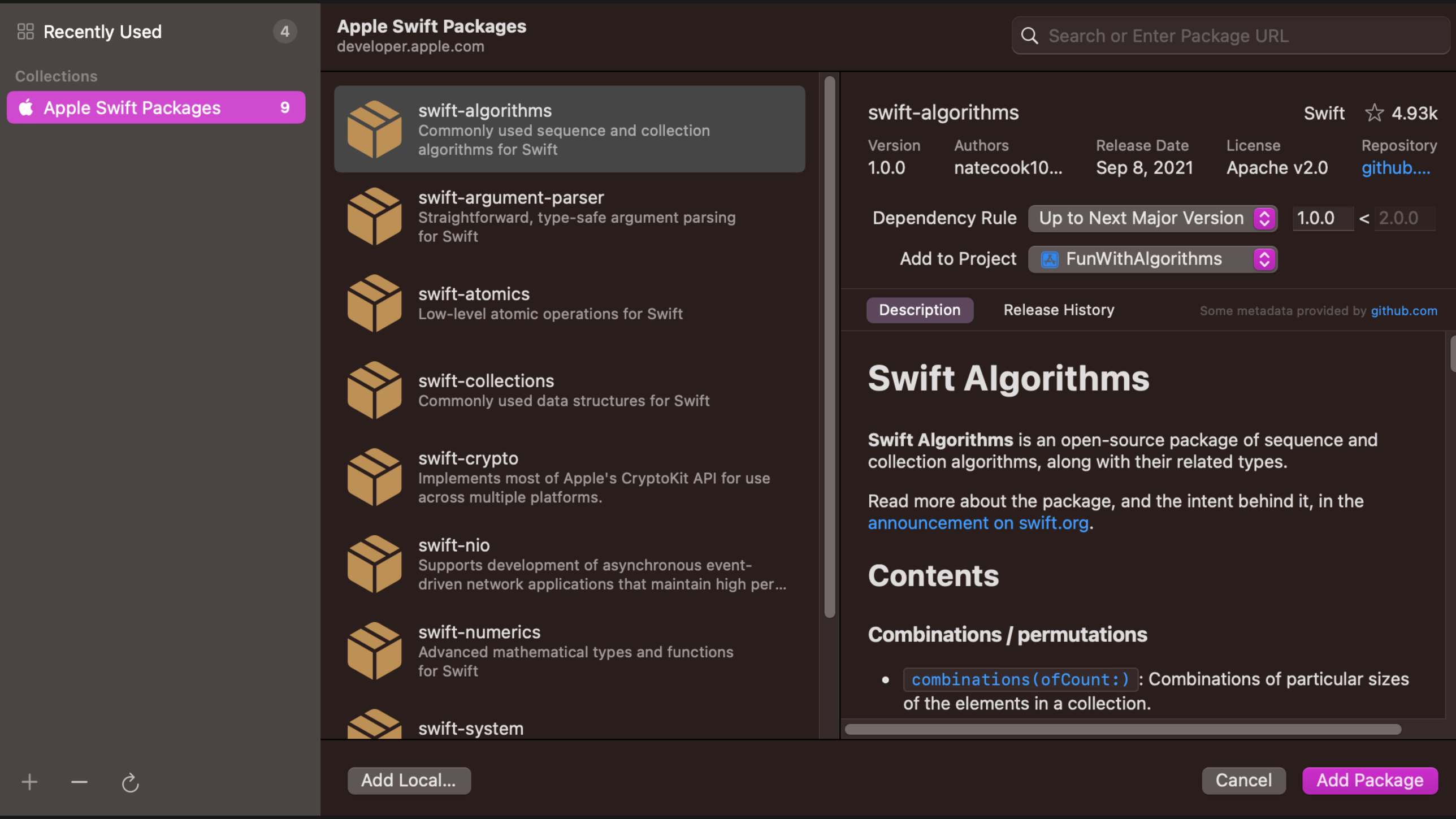
```swift
extension Media {
    func createCollectionElements() -> [MediaElement]
}


media
  .map { $0.createCollectionElements() }

    // [[MediaElement]]
```



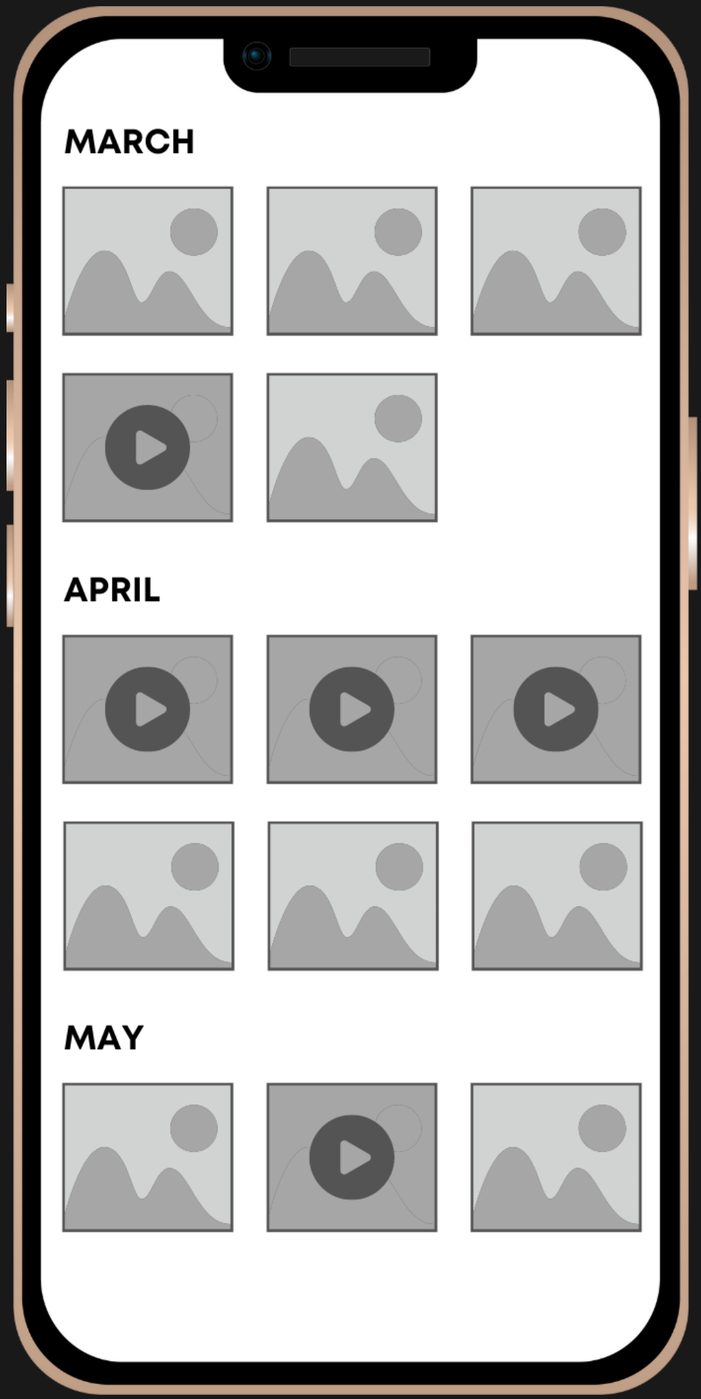
```swift
extension Media {
    func createCollectionElements() -> [MediaElement]
}
```

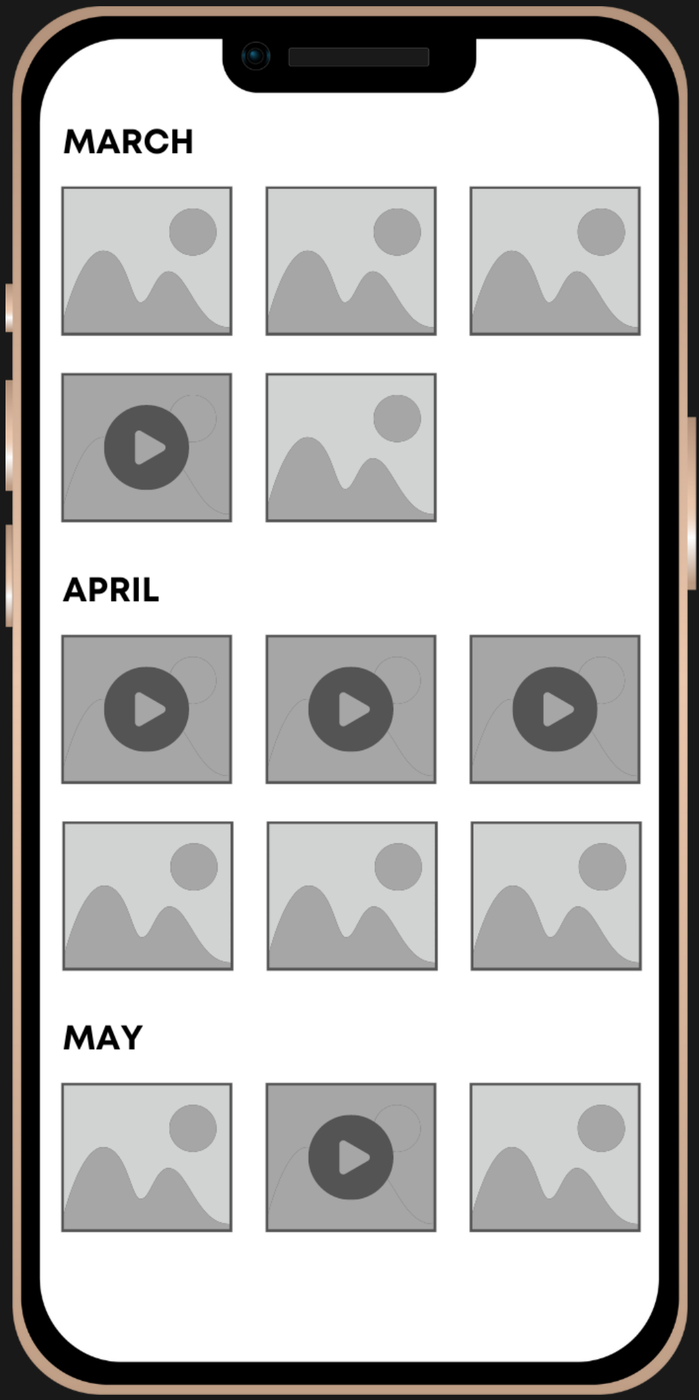
```swift
media
  .flatMap { $0.createCollectionElements() }

    // [MediaElement]
```

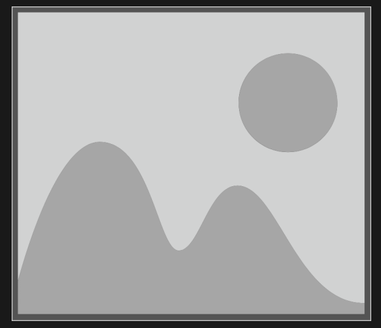
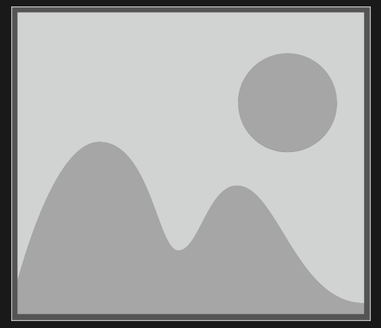
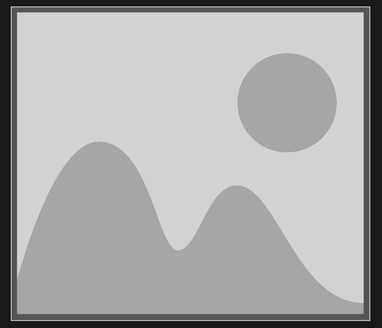


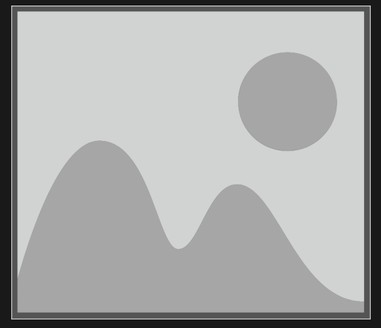

Recently Used
4
Collections
Apple Swift Packages
9
Apple Swift Packages
developer.apple.com
Search or Enter Package URL
swift-algorithms
Commonly used sequence and collection algorithms for Swift
swift-argument-parser
Straightforward, type-safe argument parsing for Swift
swift-atomics
Low-level atomic operations for Swift
swift-collections
Commonly used data structures for Swift
swift-crypto
Implements most of Apple's CryptoKit API for use across multiple platforms.
swift-nio
Supports development of asynchronous event-driven network applications that maintain high per...
swift-numerics
Advanced mathematical types and functions for Swift
swift-system
swift-algorithms
Swift
4.93k
Version
1.0.0
Authors
natecook10...
Release Date
Sep 8, 2021
License
Apache v2.0
Repository
github....
Dependency Rule
Up to Next Major Version
1.0.0
< 2.0.0
Add to Project
FunWithAlgorithms
Description
Release History
Some metadata provided by github.com
Swift Algorithms
Swift Algorithms is an open-source package of sequence and collection algorithms, along with their related types.
Read more about the package, and the intent behind it, in the announcement on swift.org.
Contents
Combinations / permutations
combinations(ofCount:): Combinations of particular sizes of the elements in a collection.
Add Local...
Cancel
Add Package

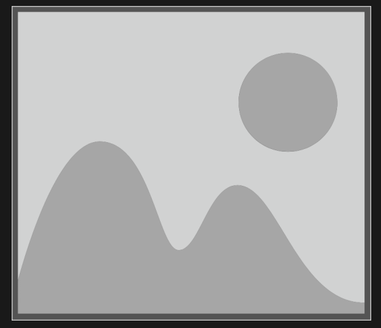
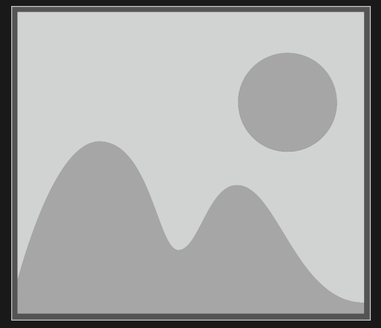
```swift
import Algorithms

extension Media {
    func createCollectionElements() -> [MediaElement]
}


media
  .flatMap { $0.createCollectionElements() }
```

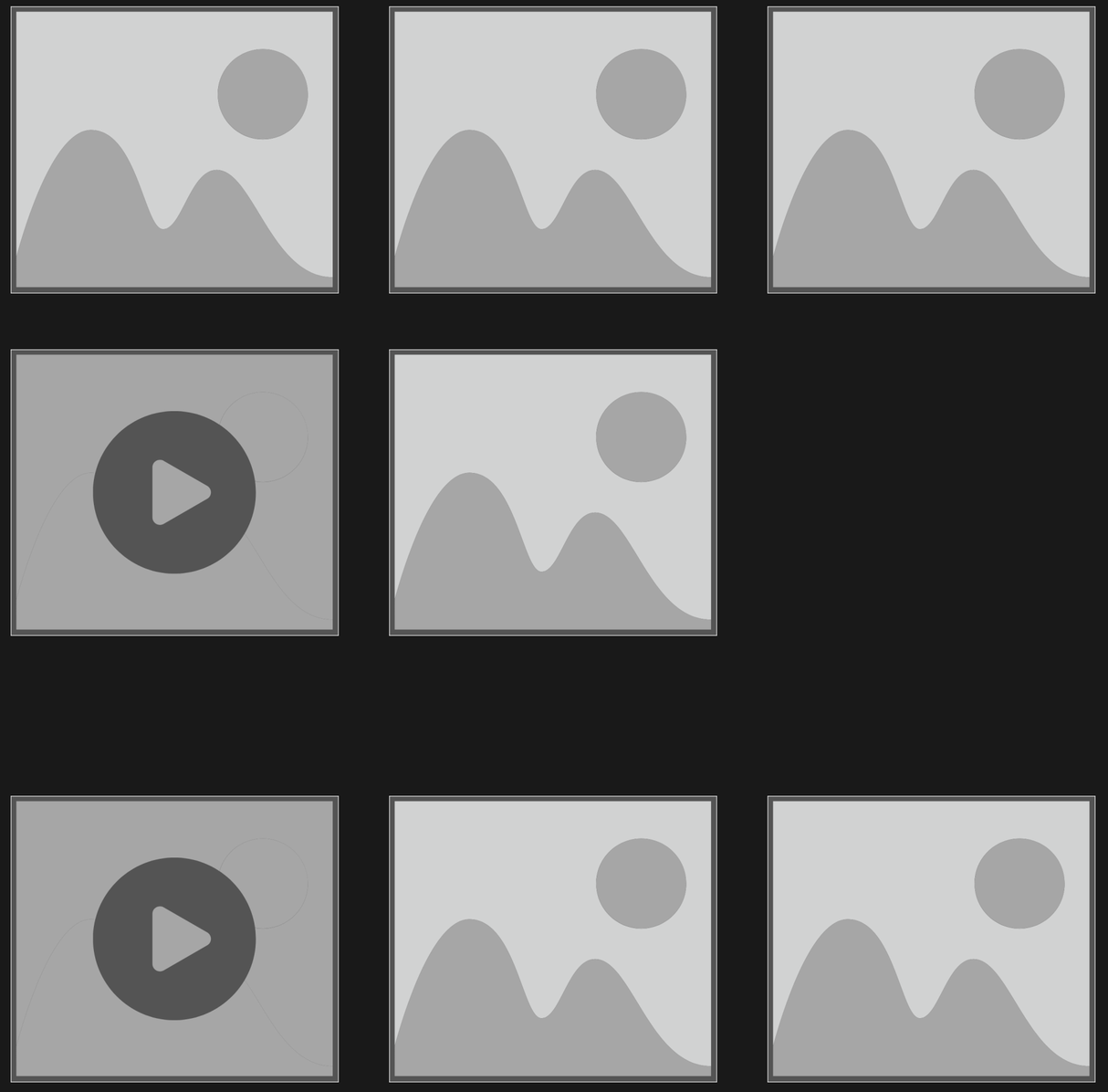



MARCH
APRIL
MAY

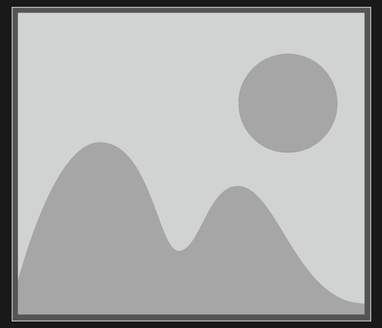
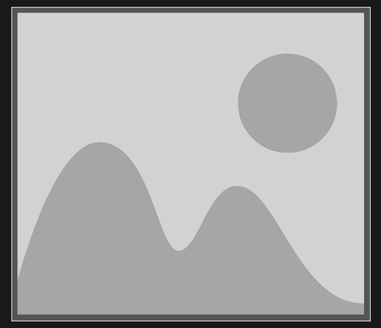
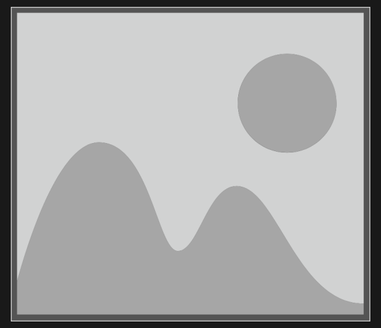




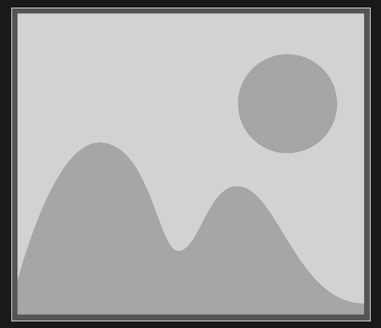


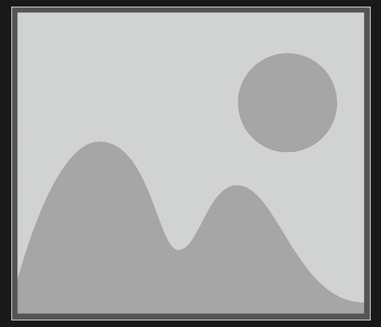
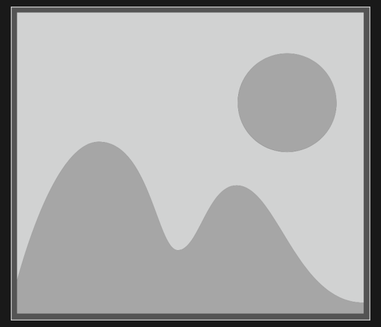


APRIL



MAY

main | 11 branches | 9 tags | Go to file | Add file | Code

dickoff Merge pull request #191 from dickoff/dickoff/fix-max-code-docume... 288d8b0 on Dec 7, 2022 151 commits

.github	Prepare for Swift Algorithms 1.0 (#167)	2 years ago
Guides	Add `partitioned(by:)` (#152)	2 years ago
Sources/Algorithms	Corrects variable name in max functions to better reflect usage	5 months ago
Tests	Clarify `endIndex` behavior of `partitioningIndex(where:)` in doc...	2 years ago
.gitignore	Ignore Package.resolved	3 years ago
CHANGELOG.md	Prepare for Swift Algorithms 1.0 (#167)	2 years ago
CODE_OF_CONDUCT.md	Initial import	3 years ago
CONTRIBUTING.md	Initial import	3 years ago
LICENSE.txt	Initial import	3 years ago
Package.swift	Prepare for Swift Algorithms 1.0 (#167)	2 years ago
README.md	Add `partitioned(by:)` (#152)	2 years ago

README.md

Swift Algorithms

Swift Algorithms is an open-source package of sequence and collection algorithms, along with their related types.

Read more about the package, and the intent behind it, in the announcement on swift.org.

Contents

About

Commonly used sequence and collection algorithms for Swift

algorithm | iterator | itertools

Readme
Apache-2.0 license
Code of conduct
5k stars
235 watching
369 forks
Report repository

Releases 9

Swift Algorithms 1.0.0 Latest
on Sep 8, 2021

+ 8 releases

Packages

No packages published

Used by 1

@akirataguchi115 / akirataguchi115

Contributors 36

Chunked

[Source | Tests]

Break a collection into subsequences where consecutive elements pass a binary predicate, or where all elements in each chunk project to the same value.

Also includes a `chunks(ofCount:)` that breaks a collection into subsequences of a given `count`.

There are two variations of the `chunked` method: `chunked(by:)` and `chunked(on:)`. `chunked(by:)` uses a binary predicate to test consecutive elements, separating chunks where the predicate returns `false`. For example, you can chunk a collection into ascending sequences using this method:

```
let numbers = [10, 20, 30, 10, 40, 40, 10, 20]
let chunks = numbers.chunked(by: { $0 <= $1 })
// [[10, 20, 30], [10, 40, 40], [10, 20]]
```

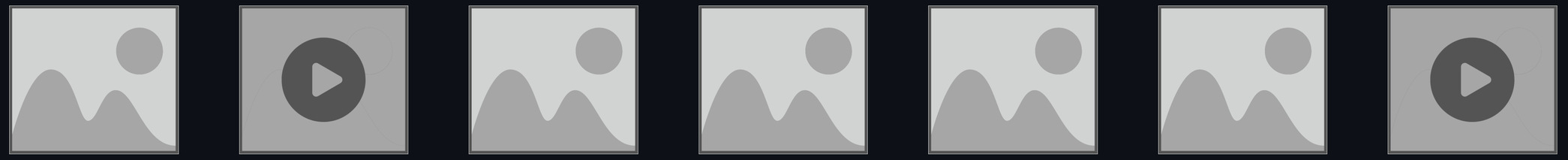
The `chunked(on:)` method, by contrast, takes a projection of each element and separates chunks where the projection of two consecutive elements is not equal. The result includes both the projected value and the subsequence that groups elements with that projected value:

```
let names = ["David", "Kyle", "Karoy", "Nate"]
let chunks = names.chunked(on: \.first!)
// [("D", ["David"]), ("K", ["Kyle", "Karoy"]), ("N", ["Nate"])]
```

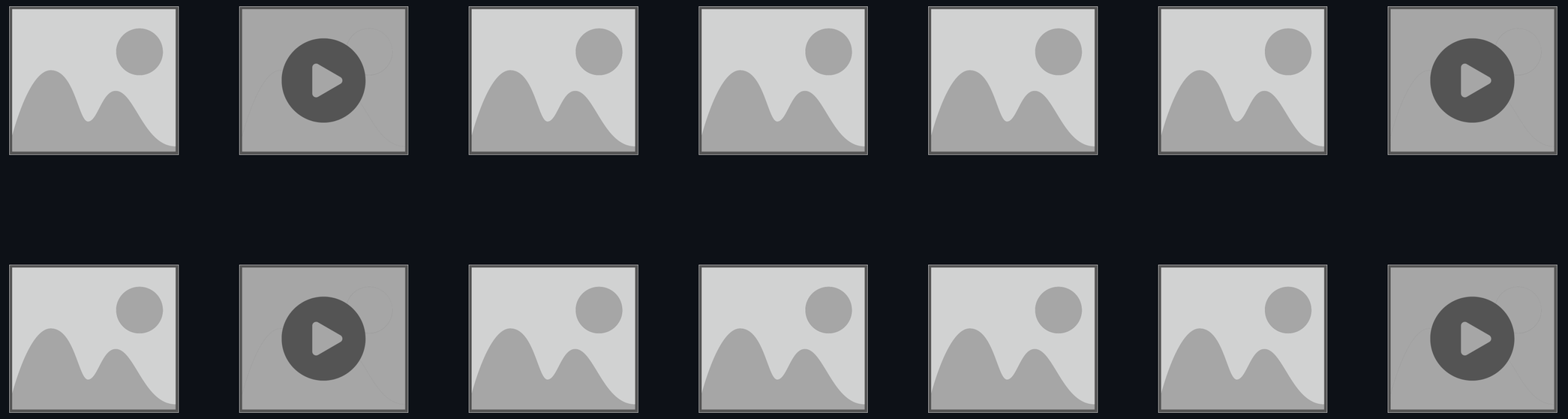
The `chunks(ofCount:)` method takes a `count` parameter (greater than zero) and separates the collection into chunks of this given count. If the `count` parameter is evenly divided by the count of the base `Collection`, all the chunks will have a count equal to the parameter. Otherwise, the last chunk will contain the remaining elements.

chunks(ofCount: ???)

chunks(ofCount: 3)



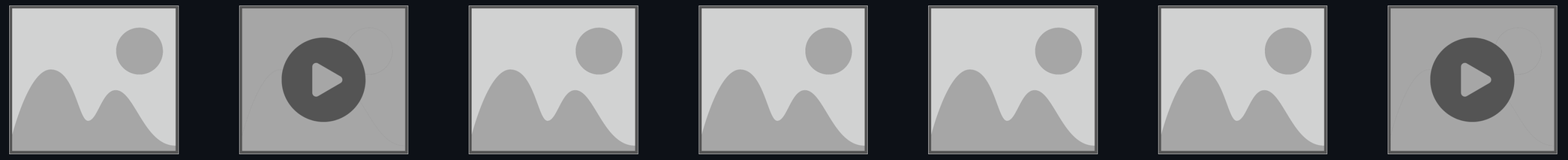
chunks(ofCount: 3)



chunks(ofCount: 3)

chunked(by: ???)

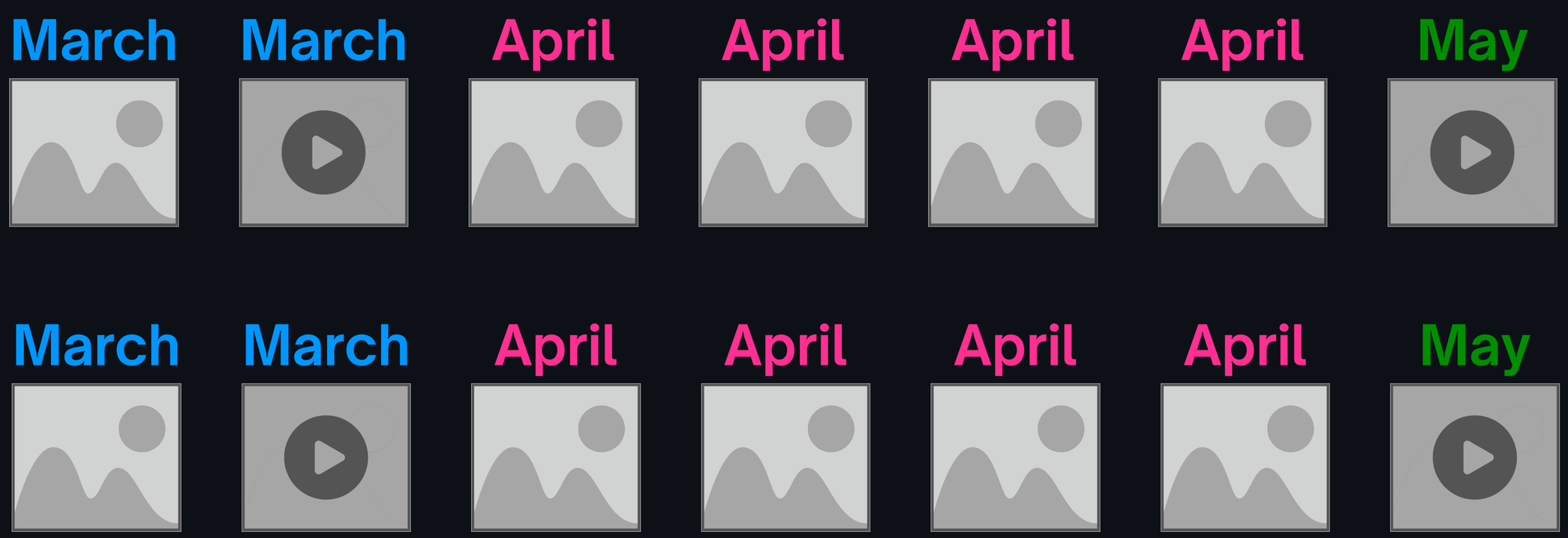
.chunked { $0.date.month == $1.date.month }



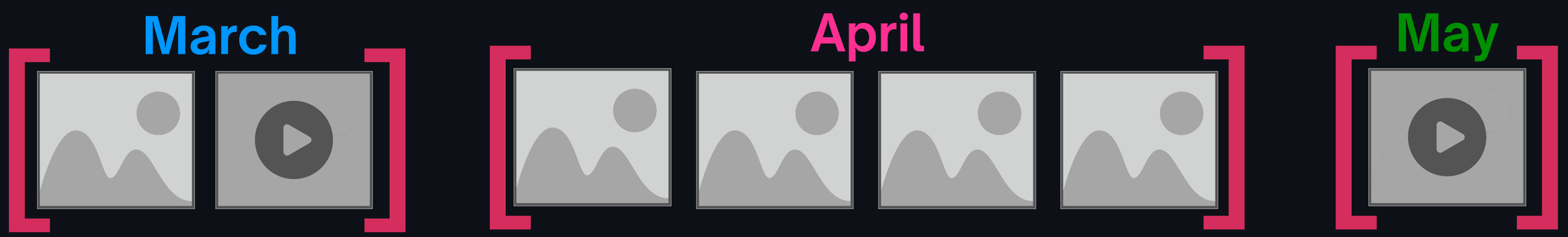
.chunked { $0.date.month == $1.date.month }

March March April April April April May

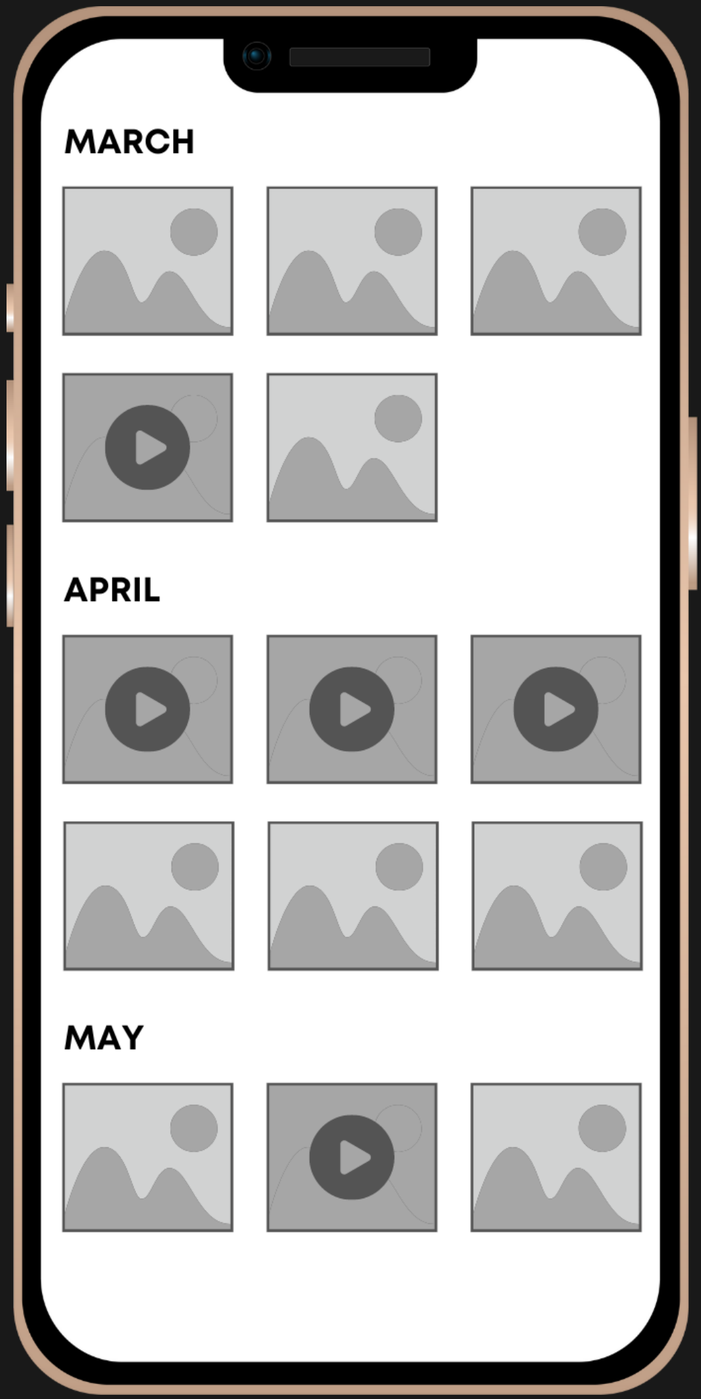
.chunked { $0.date.month == $1.date.month }



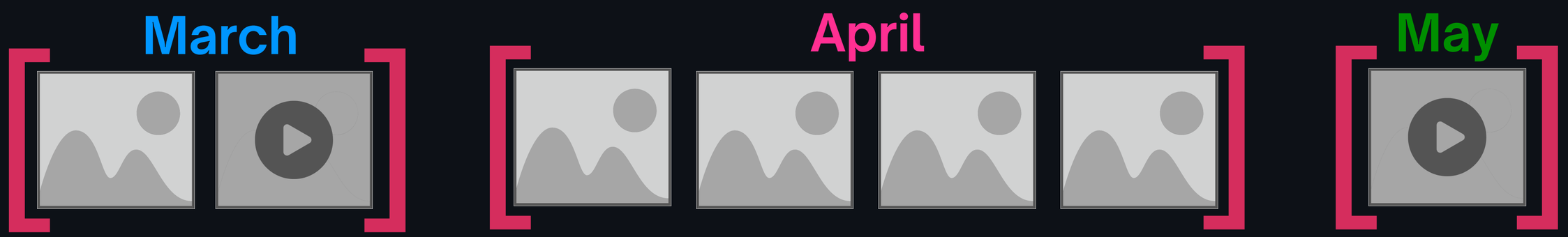
.chunked { $0.date.month == $1.date.month }



```swift
import Algorithms

extension Media {
    func createCollectionElements() -> [MediaElement]
}


media
  .flatMap { $0.createCollectionElements() }
  .chunked { $0.date.month != $1.date.month }
```



.chunked { $0.date.month == $1.date.month }



.chunked { $0.date.month == $1.date.month }

.joined(separator: ???)

Joined

[Source | Tests]

Concatenate a sequence of sequences, inserting a separator between each element.

The separator can be either a single element or a sequence of elements, and it can optionally depend on the sequences right before and after it by returning it from a closure:

```
for number in [[1], [2, 3], [4, 5, 6]].joined(by: 100) {
    print(number)
}
// 1, 100, 2, 3, 100, 4, 5, 6

for number in [[10], [20, 30], [40, 50, 60]].joined(by: { [$0.count, $1.count] }) {
    print(number)
}
// 10, 1, 2, 20, 30, 2, 3, 40, 50, 60
```

Detailed Design

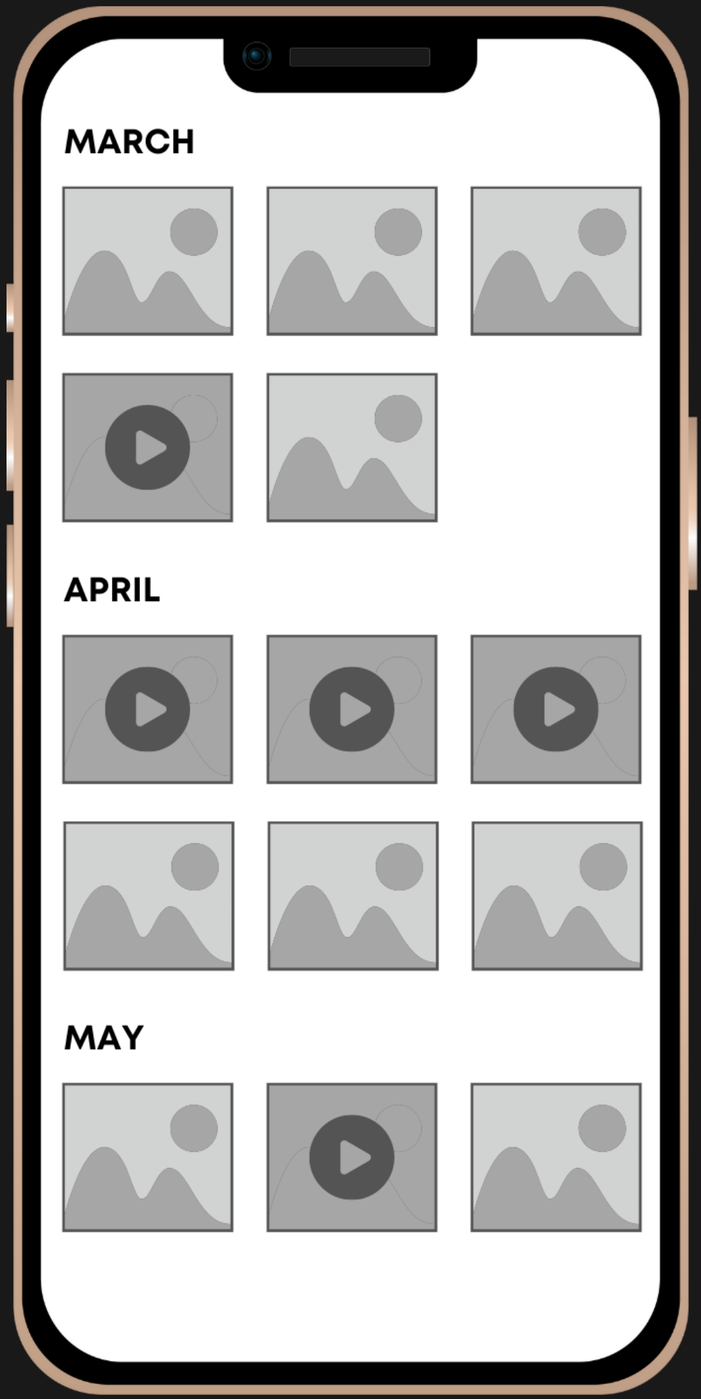
The versions that take a closure are executed eagerly and are defined on `Sequence`:

```
extension Sequence where Element: Sequence {
    public func joined(
        by separator: (Element, Element) throws -> Element.Element
    ) rethrows -> [Element.Element]
```

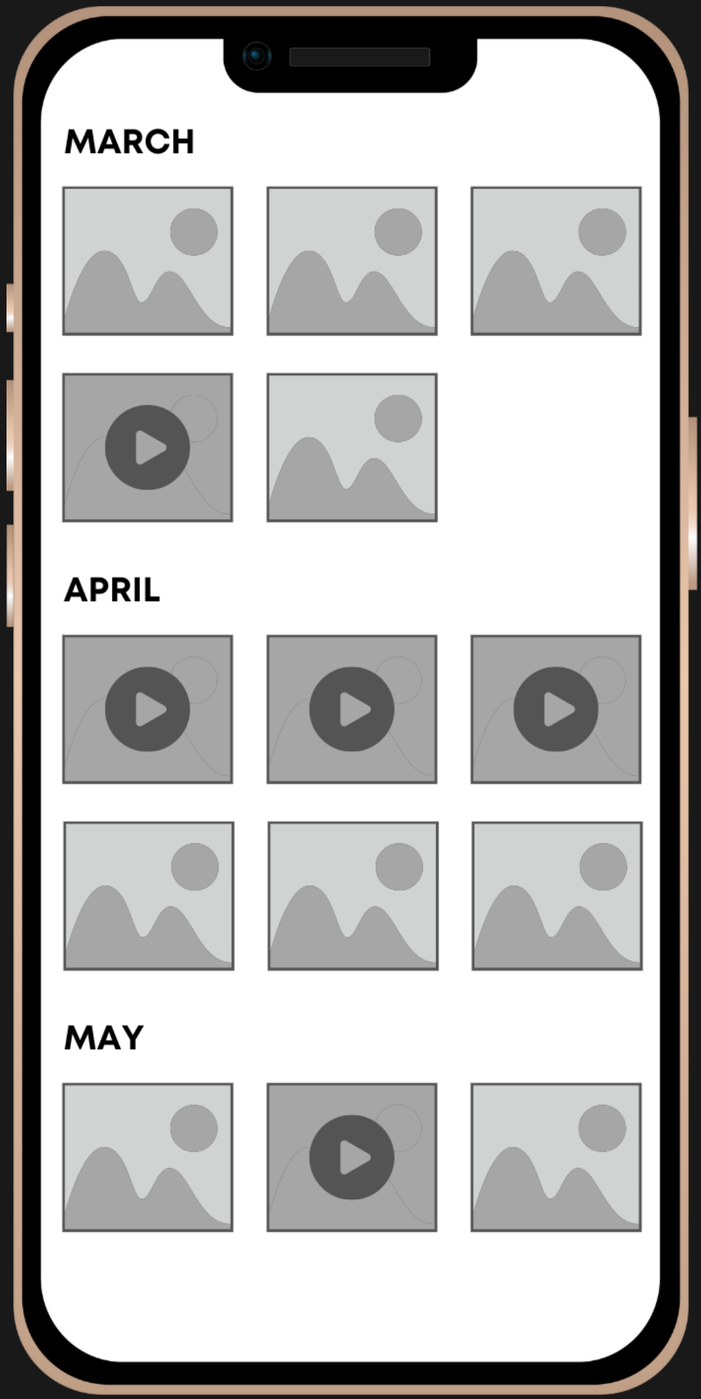
.chunked { $0.date.month == $1.date.month }

.joined { $0.first!.date.month }

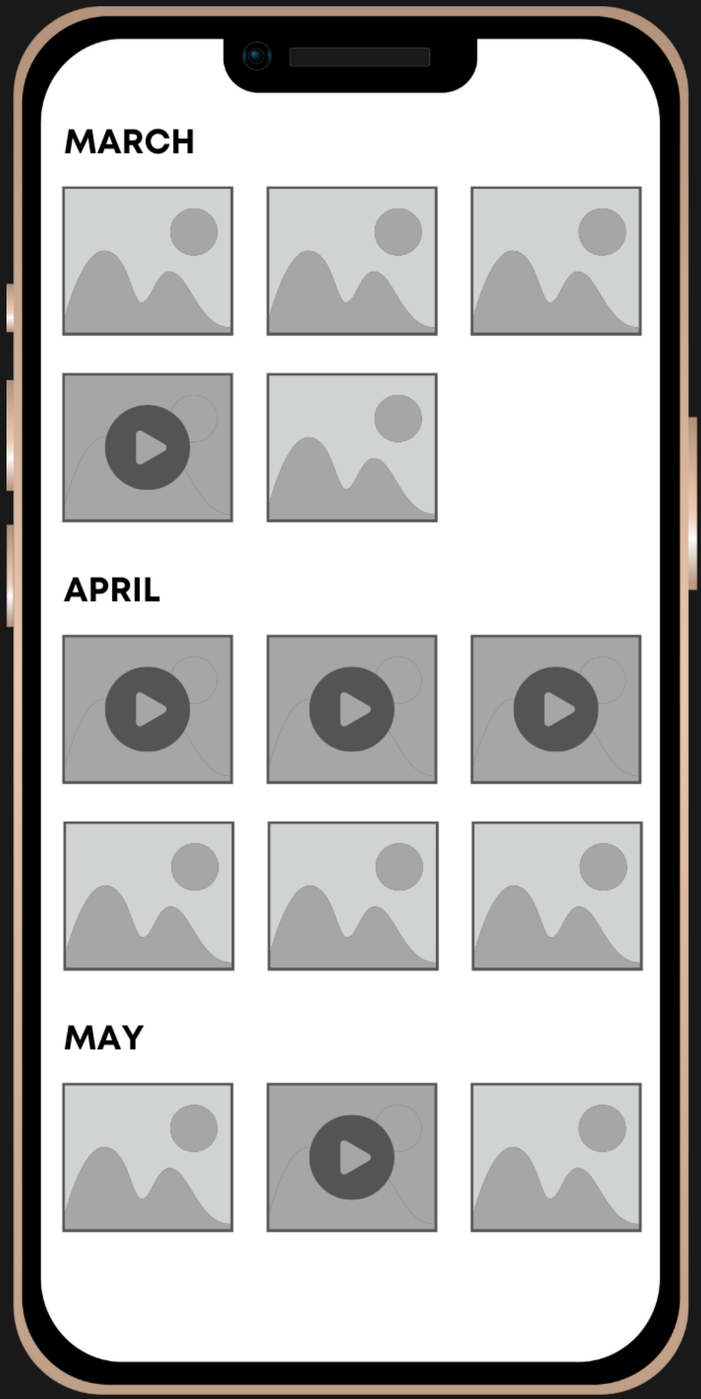
```swift
import Algorithms

extension Media {
    func createCollectionElements() -> [MediaElement]
}


media
  .flatMap { $0.createCollectionElements() }
  .chunked { $0.date.month < $1.date.month }
  .joined { $0.first!.date.month }
```



```swift
import Algorithms

extension Media {
    func createCollectionElements() -> [MediaElement]
}


media
  .lazy
  .flatMap { $0.createCollectionElements() }
  .chunked { $0.date.month < $1.date.month }
  .joined { $0.first!.date.month }
```



```swift
import Algorithms


extension Media {
    func createCollectionElements() -> [MediaElement]
}



result = Array(
    media
      .lazy
      .flatMap { $0.createCollectionElements() }
      .chunked { $0.date.month < $1.date.month }
      .joined { $0.first!.date.month }
)
```



Use algorithms to make your code faster

better

concise

One more thing

Home

PUBLIC

Questions

Tags

Users

Companies

COLLECTIVES

Explore Collectives

TEAMS

How can I pair socks from a pile efficiently?

Ask Question

Asked 10 years, 3 months ago Modified 4 months ago Viewed 436k times

4160

Yesterday I was pairing the socks from the clean laundry and figured out the way I was doing it is not very efficient. I was doing a naive search — picking one sock and "iterating" the pile in order to find its pair. This requires iterating over n/2 * n/4 = $n^2/8$ socks on average.

As a computer scientist I was thinking what I could do? Sorting (according to size/color/...) of course came to mind to achieve an O(NlogN) solution.

Hashing or other not-in-place solutions are not an option, because I am not able to duplicate my socks (though it could be nice if I could).

So, the question is basically:

Given a pile of `n` pairs of socks, containing `2n` elements (assume each sock has exactly one matching pair), what is the best way to pair them up efficiently with up to logarithmic extra space? (I believe I can remember that amount of info if needed.)

I will appreciate an answer that addresses the following aspects:

- A general *theoretical* solution for a huge number of socks.
- The actual number of socks is not that large, I don't believe my spouse and I have more than 30 pairs. (And it is fairly easy to distinguish between my socks and hers; can this be used as well?)
- Is it equivalent to the element distinctness problem?

algorithm sorting language-agnostic matching

The Overflow Blog

Looking for job perks? How about saving the world?

How a top-ranked engineering school reimagined CS curriculum (Ep. 565)

Featured on Meta

Improving the copy in the close modal and post notices - 2023 edition

New blog post from our CEO Prashanth: Community is the future of AI

Content Discovery initiative April 13 update: Related questions using a...

Temporary policy: ChatGPT is banned

Review our technical responses for the 2023 Developer Survey

Linked

2 What does this syntax mean in Keras Dropout(0.5)(X)?

25556 How do I undo the most recent local commits in Git?

Why is processing a sorted array faster

THANK YOU!

@dvrzan

iosdev.space/@dvrzan

danijelavrzan.com