PRANAV KASETTI

SWIFT ALGORITHMS

WHAT IS SWIFT ALGORITHMS?

- Swift Algorithms is a new open-source package of Sequence and Collection algorithms, along with their related types.
- Apple is trialling this Evolution format for exploring experimental features.
- Evolution Pitches still happen as usual.

WHY SHOULD I CARE?

- Don't have to reinvent the wheel for well-known algorithms.
- Makes code easier to write.
- Makes cleaner code that's easier to read.
- Avoids correctness traps with algorithmic code.
- Avoids performance traps (memory and runtime).

DEMO!

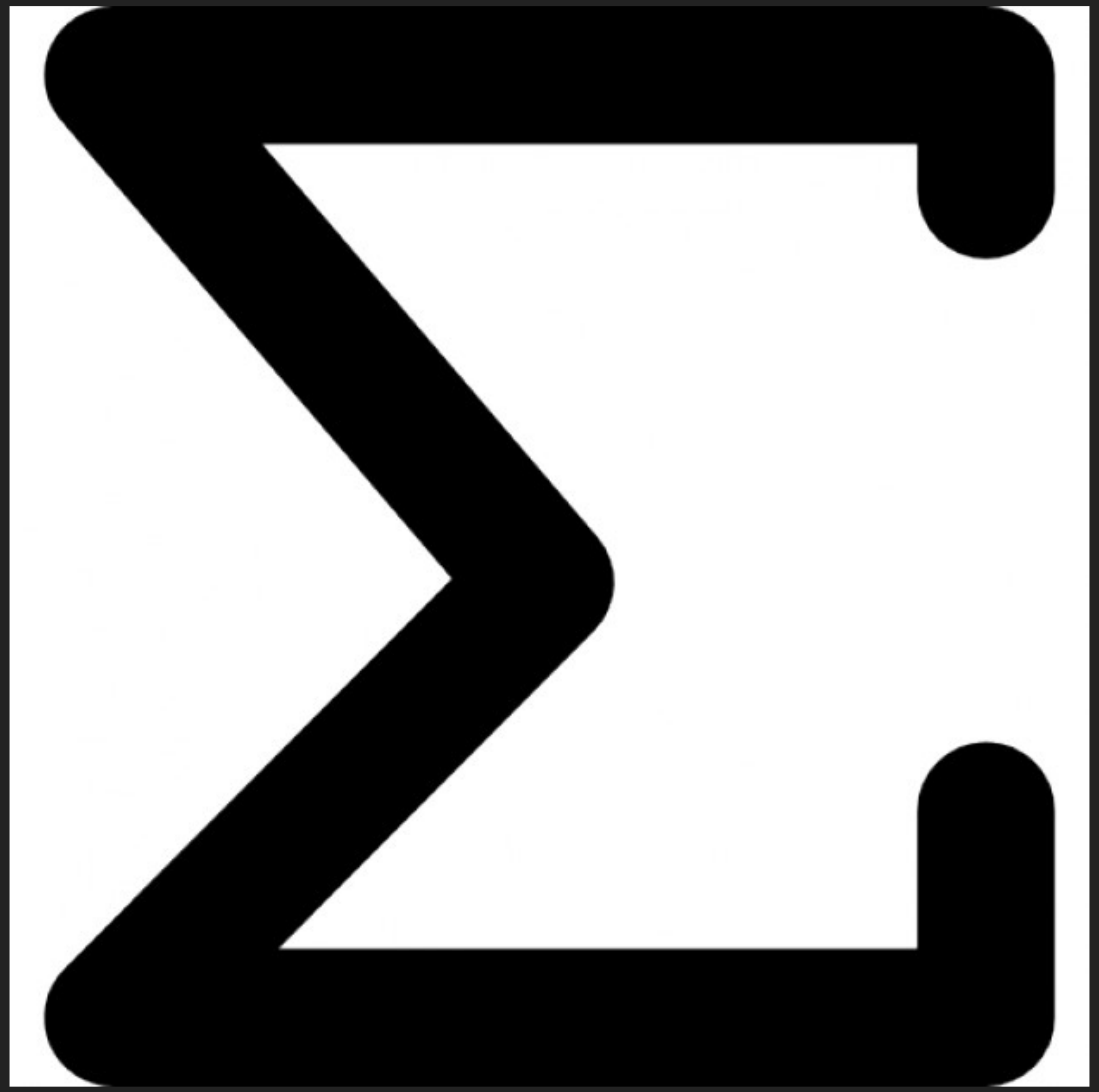
FOUR SUM

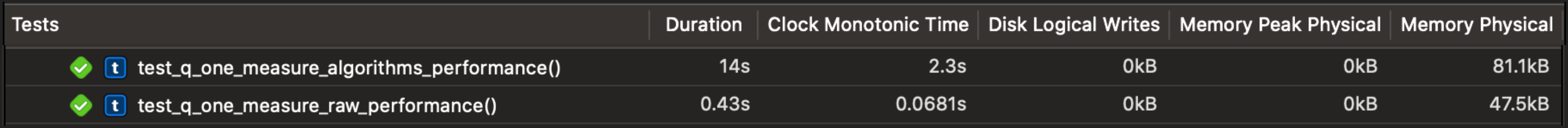
- Given an array nums of n integers and an integer target, are there elements a, b, c, and d in nums such that a + b + c + d = target? Find all unique quadruplets in the array which gives the sum of target.

- Notice that the solution set must not contain duplicate quadruplets.

- Example 1:

- Input: nums = [1,0,-1,0,-2,2], target = 0
- Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
- Constraints: 0 <= nums.length <= 200, -109 <= nums[i] <= 109, -109 <= target <= 109





Tests	Duration	Clock Monotonic Time	Disk Logical Writes	Memory Peak Physical	Memory Physical
✓ t test_q_one_measure_algorithms_performance()	14s	2.3s	0kB	0kB	81.1kB
✓ t test_q_one_measure_raw_performance()	0.43s	0.0681s	0kB	0kB	47.5kB

Four Sum Comparison of Solution Performances

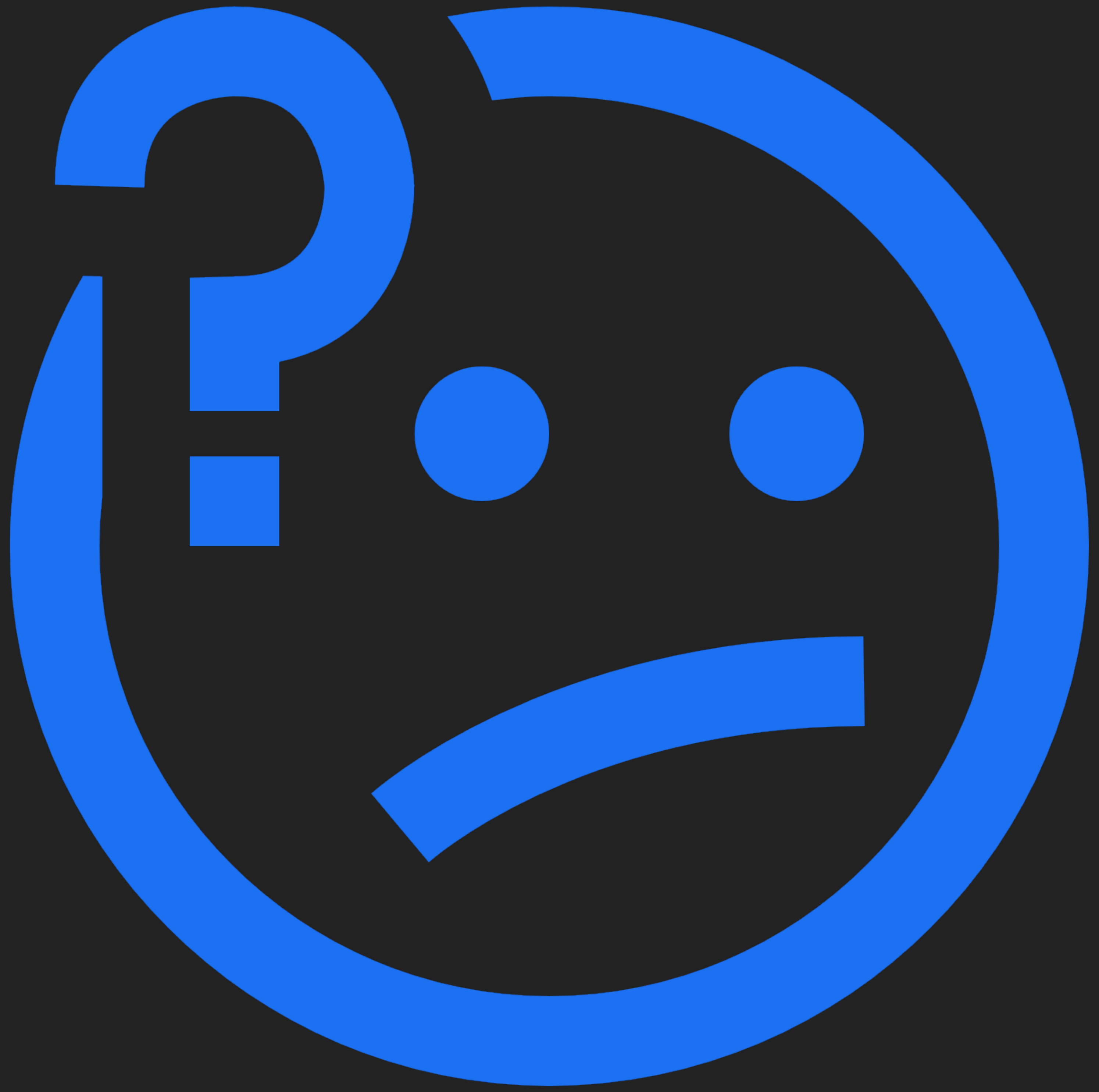
CORRECTNESS TRAPS

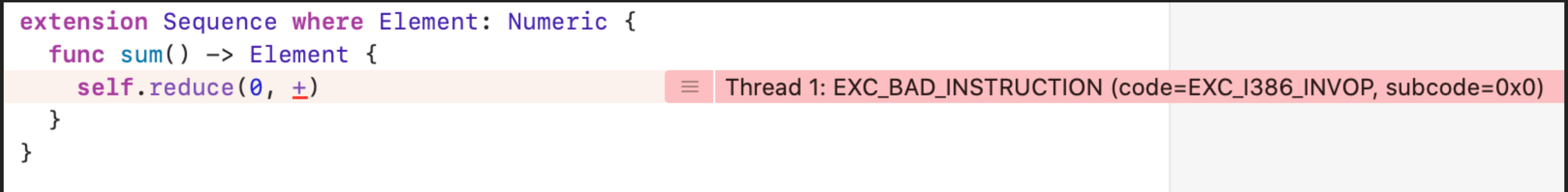
EXAMPLE:

```swift
extension Sequence where Element: Numeric {
  func sum() -> Element {
    self.reduce(0, +)
  }
}
```

- `[100 as Int8, 28, -100, -100].sum()`

- What happens?





```
extension Sequence where Element: Numeric {
  func sum() -> Element {
    self.reduce(0, +)
  }
}
```

Thread 1: EXC_BAD_INSTRUCTION (code=EXC_I386_INVOP, subcode=0x0)

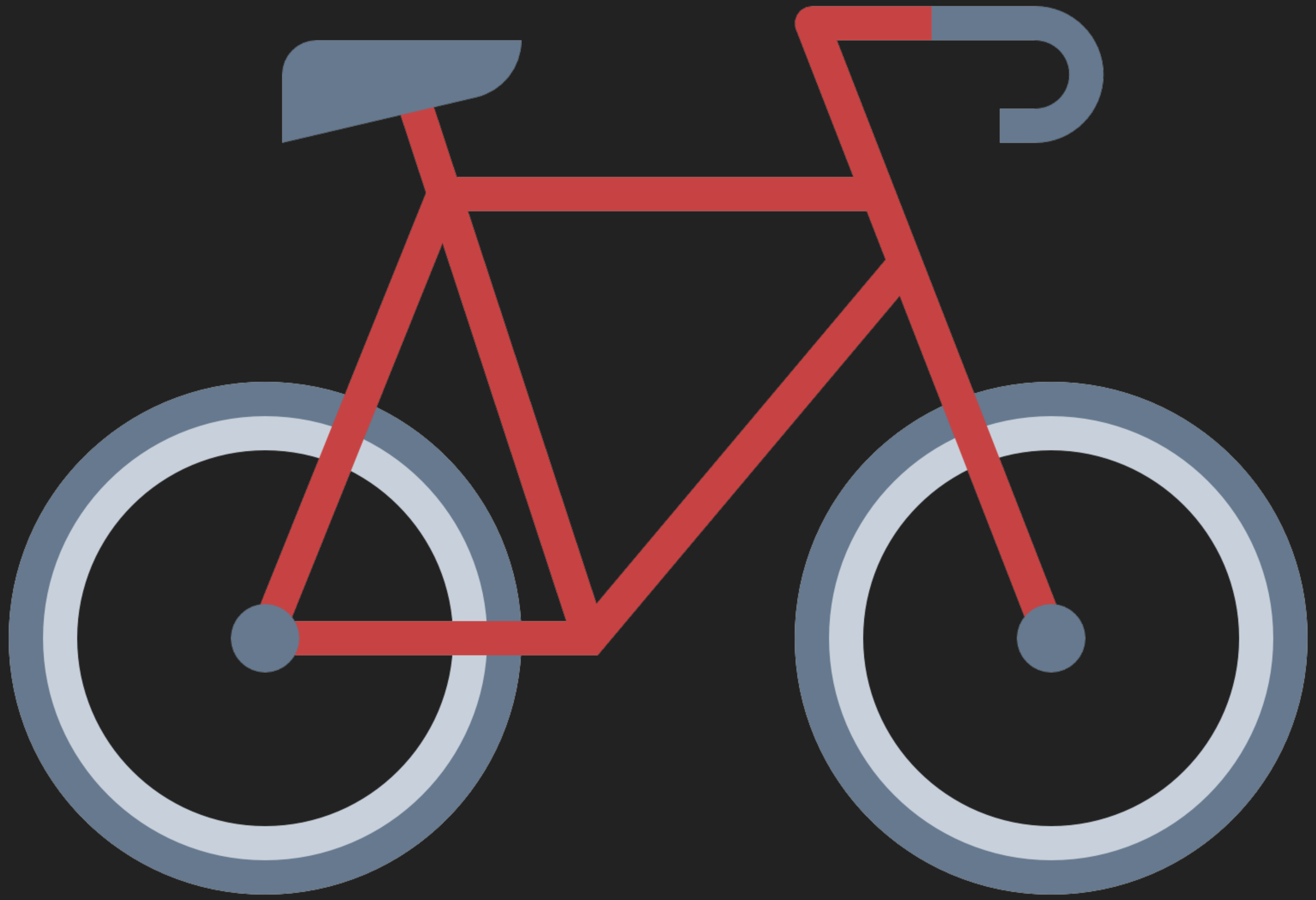
```
(result, overflow) = result.addingReportingOverflow(element)
```

POTENTIAL SOLUTION (INTEGERS)

CYCLES

```
for x in (1...3).cycled(times: 3) {
  print(x)
}
```

Prints 1 through 3 three times without nested for loops.



UNIQUED

```
let numbers = [1, 2, 3, 3, 2, 3, 3, 2, 2, 2, 1]
let unique = numbers.uniqued()
// unique == [1, 2, 3]
```

- This PRESERVES the initial ordering of the elements, unlike using:

```
let unique = Array(Set(numbers))
```

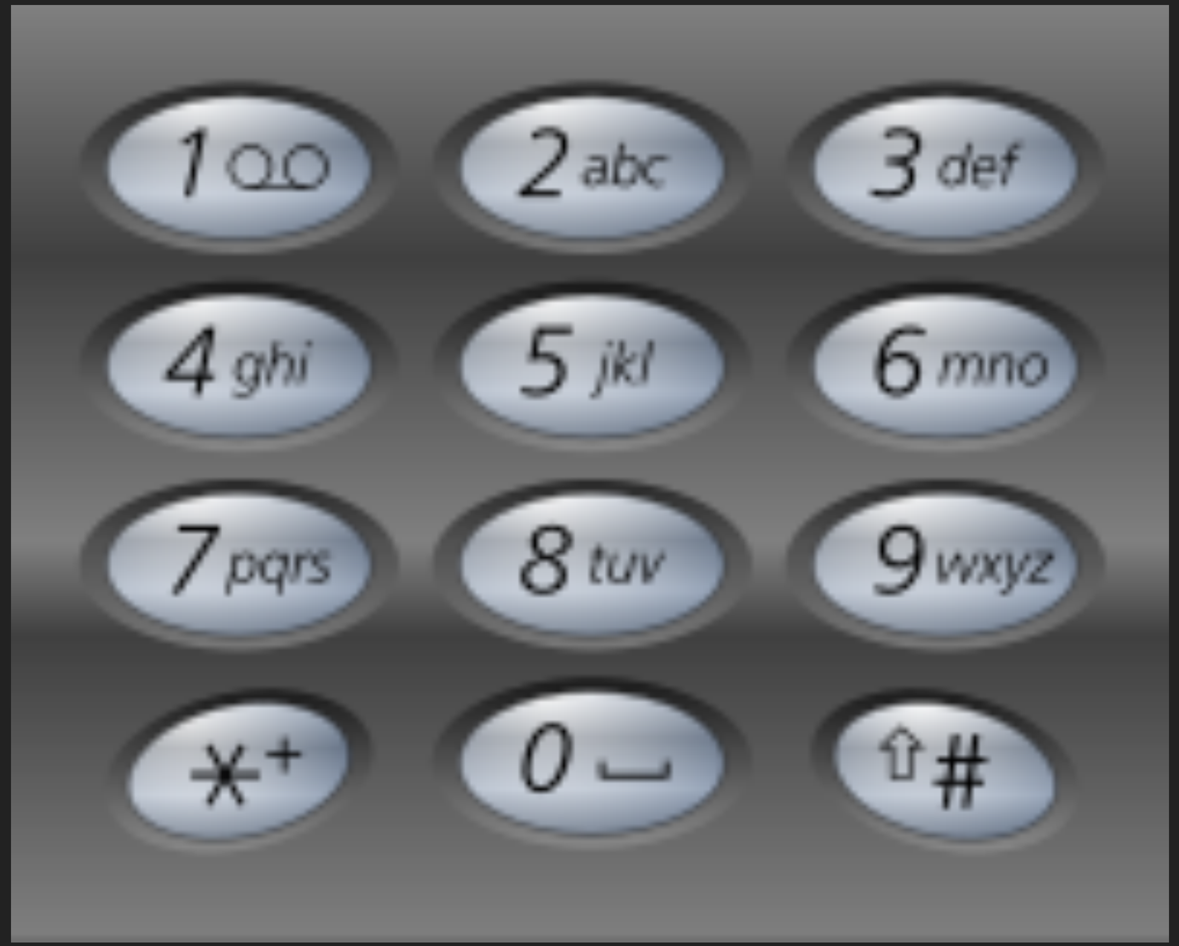
COMBINATIONS

```
let numbers = [10, 20, 30, 40]
for combo in numbers.combinations(ofCount: 2) {
  print(combo)
}
// [10, 20], [10, 30], [10, 40], [20, 30], [20, 40], [30, 40]
```

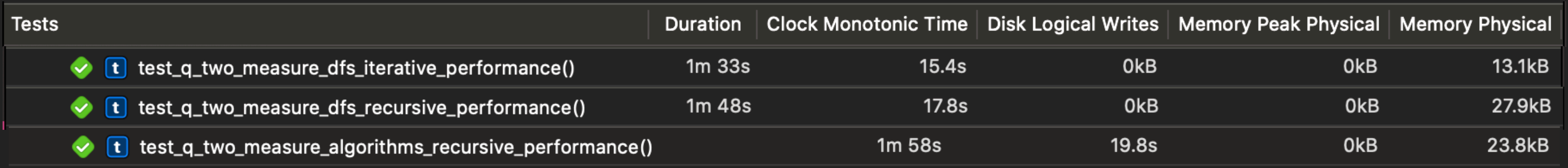
DEMO!

LETTER COMBINATIONS OF A PHONE NUMBER

- Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.
- A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.
- Example 1:
- Input: digits = "23"
- Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]



SWIFT ALGORITHMS



Tests	Duration	Clock Monotonic Time	Disk Logical Writes	Memory Peak Physical	Memory Physical
✓ t test_q_two_measure_dfs_iterative_performance()	1m 33s	15.4s	0kB	0kB	13.1kB
✓ t test_q_two_measure_dfs_recursive_performance()	1m 48s	17.8s	0kB	0kB	27.9kB
✓ t test_q_two_measure_algorithms_recursive_performance()		1m 58s	19.8s	0kB	23.8kB

SUMMARY

- Swift Algorithms can make your code cleaner, safer and faster for many things, but is not always the best tool.
- Can be used in projects right now using the Swift Package Manager, but will be available in the standard library later.
- You can contribute to this package online at: https://github.com/apple/swift-algorithms.

@SPIDEY_VITAMINS
WWW.GITHUB.COM/LEMONSPIKE

THANK YOU!