Prompt | Scratchpad | Our Solution(s) | Video Explanation | Run Code

Solution 1 | Solution 2

```swift
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
    // O(w * n * log(n) + n * w * log(w)) time | O(wn) space - where w is the number of wor
    // n is the length of the longest word
    func groupAnagrams(_ words: [String]) -> [[String]] {
        if words.count == 0 {
            return [[String]]()
        }

        var sortedWords = [String]()
        var indices = [Int]()
        for i in 0 ..< words.count {
            sortedWords.append(sortWord(words[i]))
            indices.append(i)
        }
        indices = indices.sorted {
            return sortedWords[$0] < sortedWords[$1]
        }

        var result = [[String]]()
        var currentAnagramGroup = [String]()
        var currentAnagram = sortedWords[indices[0]]
        for index in indices {
            let word = words[index]
            let sortedWord = sortedWords[index]
            if currentAnagramGroup.count == 0 {
                currentAnagramGroup.append(word)
                currentAnagram = sortedWord
                continue
            }
            if sortedWord == currentAnagram {
                currentAnagramGroup.append(word)
                continue
            }

            result.append(currentAnagramGroup)
            currentAnagramGroup = [word]
            currentAnagram = sortedWord
        }

        result.append(currentAnagramGroup)
        return result
    }

    func sortWord(_ word: String) -> String {
        return String(word.sorted())
    }
}

```

Your Solutions | Run Code

Solution 1 | Solution 2 | Solution 3

```swift
class Program {
    func groupAnagrams(_ words: [String]) -> [[String]] {
        // Write your code here.
        return []
    }
}

```

Custom Output | Raw Output | Submit Code

**Run or submit code when you're ready.**