Our Solution(s) Run Code

Solution 1 Solution 2

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

package main

import "sort"

// O(w * n * log(n)) time | O(wn) space - where w is the number
// n is the length of the longest word
func GroupAnagrams(words []string) [][]string {
  anagrams := map[string][]string{}

  for _, word := range words {
    sortedWord := sortWord(word)
    anagrams[sortedWord] = append(anagrams[sortedWord], word)
  }

  result := [][]string{}
  for _, group := range anagrams {
    result = append(result, group)
  }
  return result
}

func sortWord(word string) string {
  wordBytes := []byte(word)
  sort.Slice(wordBytes, func(i, j int) bool {
    return wordBytes[i] < wordBytes[j]
  })
  return string(wordBytes)
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

func GroupAnagrams(words []string) [][]string {
  // Write your code here.
  return nil
}

```

Our Tests Custom Output Submit Code

```
import (
    "sort"


    "github.com/stretchr/testify/require"
)

func (s *TestSuite) TestCase1() *TestCase {
    words := []string{}
    expected := [][]string{}
    output := GroupAnagrams(words)
    compare(t, expected, output)
}

func (s *TestSuite) TestCase2() *TestCase {
    words := []string{"test"}
    expected := [][]string{{"test"}}
    output := GroupAnagrams(words)
    compare(t, expected, output)
}
```

Run or submit code when you're ready.