Our Solution(s) Run Code

Solution 1 Solution 2

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

import java.util.*;
import java.util.stream.*;

class Program {
  // O(w * n * log(n) + n * w * log(w)) time | O(wn) space - where w is the number
  // n is the length of the longest word
  public static List<List<String>> groupAnagrams(List<String> words) {
    if (words.size() == 0) return new ArrayList<List<String>>();

    List<String> sortedWords = new ArrayList<String>();
    for (String word : words) {
      char[] charArray = word.toCharArray();
      Arrays.sort(charArray);
      String sortedWord = new String(charArray);
      sortedWords.add(sortedWord);
    }

    List<Integer> indices = IntStream.range(0, words.size()).boxed().collect(Colle
    indices.sort((a, b) -> sortedWords.get(a).compareTo(sortedWords.get(b)));

    List<List<String>> result = new ArrayList<List<String>>();
    List<String> currentAnagramGroup = new ArrayList<String>();
    String currentAnagram = sortedWords.get(indices.get(0));
    for (Integer index : indices) {
      String word = words.get(index);
      String sortedWord = sortedWords.get(index);

      if (sortedWord.equals(currentAnagram)) {
        currentAnagramGroup.add(word);
        continue;
      }

      result.add(currentAnagramGroup);
      currentAnagramGroup = new ArrayList<String>(Arrays.asList(word));
      currentAnagram = sortedWord;
    }

    result.add(currentAnagramGroup);

    return result;
  }
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```java
import java.util.*;

class Program {
  public static List<List<String>> groupAnagrams(List<String> words) {
    // Write your code here.
    return null;
  }
}

```

Custom Output Raw Output Submit Code

Run or submit code when you're ready.