Program Structures and Algorithms

Spring 2023(SEC – 03)

NAME: Changyu Wu

NUID: 002111594

**Task: Assignment 2 (3-SUM)**

1. **Evidence:**

All test cases were passed. Screenshots were attached below.

文本

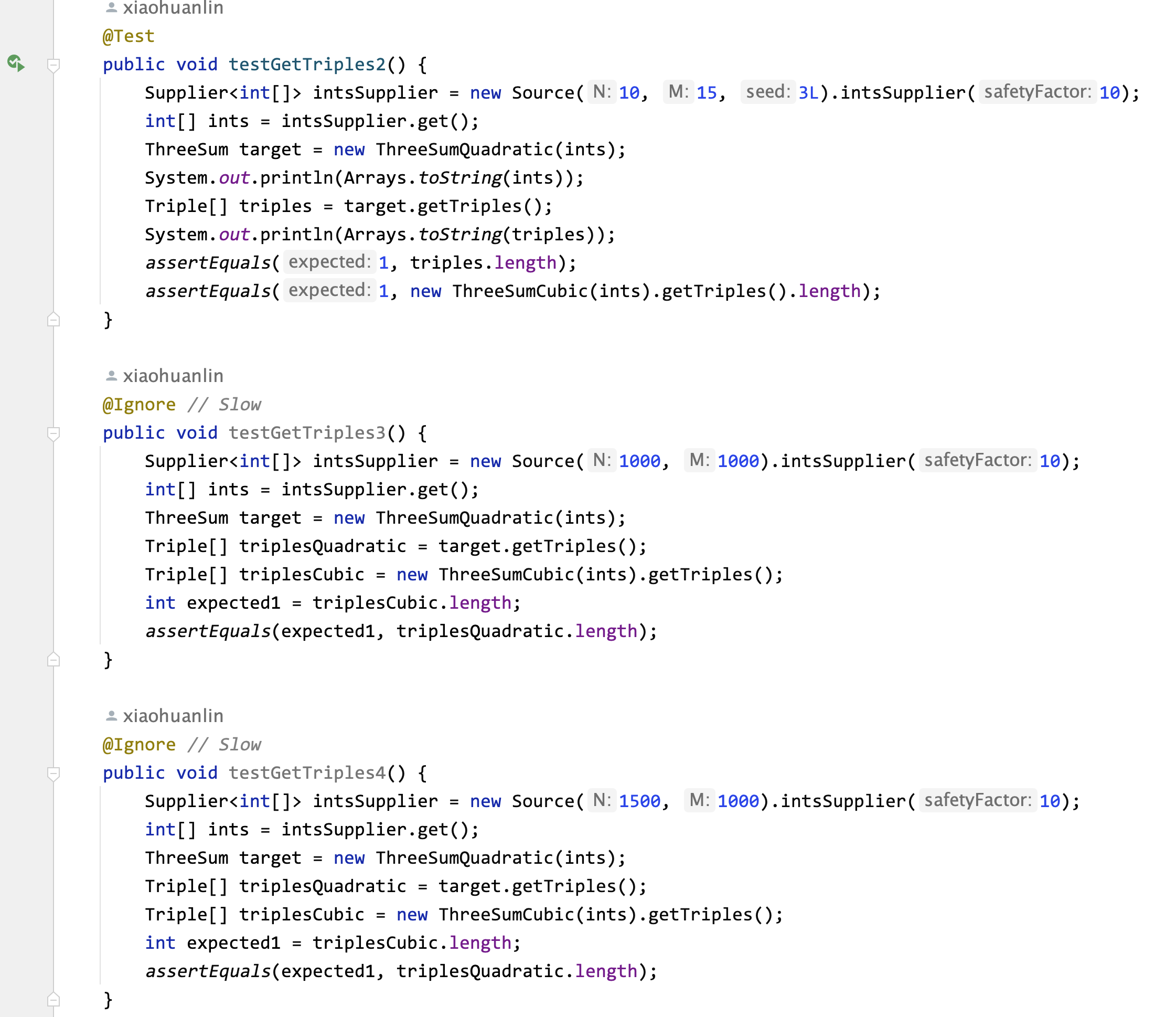
描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成



图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

1. **Spreadsheet:**

The spreadsheet is in the folder.

表格

描述已自动生成

1. **Explanation:**
2. Quadratic:

public List<Triple> getTriples(int j) {  
 List<Triple> triples = new ArrayList<>();  
 *// FIXME : for each candidate, test if a[i] + a[j] + a[k] = 0.* int i = j - 1, k = j + 1;  
 while (i >= 0 && k < length) {  
 int sum = a[i] + a[j] + a[k];  
 if (sum == 0) { *// Match* triples.add(new Triple(a[i], a[j], a[k]));  
 i--; *// Continue* } else if (sum < 0) {  
 k++;  
 } else { *// sum > 0* i--;  
 }  
 }  
  
 *// END* return triples;  
}

In the first loop, we traverse each number from 0 to length-1 in the given sorted array, and mark the index of this number as j. Each j represents the index of the middle of the triples. Based on that, we keep traverse the rest of the array with other two indices i and k. Starting from j, i always goes to the left side, while k goes to the right side. In each step, we calculate the summation of three numbers. If the sum equals to zero, we will create a new Triple object and add it to triples. If the sum is smaller than zero, we move the right index k one step to the right side to get a potentially larger summation. On the contrary, when the sum is larger than zero, we move the left index i one step to the left. Until i < 0 or k >= length, we stop the second loop.

1. QuadraticWithCalipers:

public static List<Triple> calipers(int[] a, int i, Function<Triple, Integer> function) {  
 List<Triple> triples = new ArrayList<>();  
 *// FIXME : use function to qualify triples and to navigate otherwise.* int j = i + 1, k = a.length - 1;  
 while (j < k) {  
 Triple triple = new Triple(a[i], a[j], a[k]); *// Create a new triple* int sum = function.apply(triple);  
 if (sum == 0) {  
 triples.add(triple);  
 j++;  
 } else if (sum > 0) {  
 k--;  
 } else { *// sum < 0* j++;  
 }  
 }  
  
 *// END* return triples;  
}

Given the index name i, the first loop traverse from 0 to length-2. In the second loop, j begins from i+1 and k begins from length-1. Like a caliper, indices j and k gradually become closer until j >= k. To be specific, each step, we calculate the summation of three numbers. If the sum equals zero, we will add it to triples and move j one step to the right side. If the sum is larger than 0, we move k one step to the left. If the sum is smaller than 0, we move j one step to the right.