[陈](https://www.coursera.org/" \o "主页)



返回到第 3 周**课程**

[上一个](https://www.coursera.org/learn/algorithms-part1/quiz/LojjQ/interview-questions-optional)

[下一个](https://www.coursera.org/learn/algorithms-part1/supplement/efbDN/lecture-slides)

* **Mergesort**

[Overview](https://www.coursera.org/learn/algorithms-part1/supplement/mCXVI/overview)

[Lecture Slides](https://www.coursera.org/learn/algorithms-part1/supplement/4E9fa/lecture-slides)

[Mergesort](https://www.coursera.org/learn/algorithms-part1/lecture/ARWDq/mergesort)

[23 分](https://www.coursera.org/learn/algorithms-part1/lecture/ARWDq/mergesort)

[Bottom-up Mergesort](https://www.coursera.org/learn/algorithms-part1/lecture/PWNEl/bottom-up-mergesort)

[3 分](https://www.coursera.org/learn/algorithms-part1/lecture/PWNEl/bottom-up-mergesort)

[Sorting Complexity](https://www.coursera.org/learn/algorithms-part1/lecture/xAltF/sorting-complexity)

[9 分](https://www.coursera.org/learn/algorithms-part1/lecture/xAltF/sorting-complexity)

[Comparators](https://www.coursera.org/learn/algorithms-part1/lecture/9FYhS/comparators)

[6 分](https://www.coursera.org/learn/algorithms-part1/lecture/9FYhS/comparators)

[Stability](https://www.coursera.org/learn/algorithms-part1/lecture/pvvLZ/stability)

[5 分](https://www.coursera.org/learn/algorithms-part1/lecture/pvvLZ/stability)

**[练习测验:](https://www.coursera.org/learn/algorithms-part1/quiz/LojjQ/interview-questions-optional)**

[Interview Questions (optional)](https://www.coursera.org/learn/algorithms-part1/quiz/LojjQ/interview-questions-optional)

[3 questions](https://www.coursera.org/learn/algorithms-part1/quiz/LojjQ/interview-questions-optional)

**[编程作业:](https://www.coursera.org/learn/algorithms-part1/programming/prXiW/collinear-points)**

[Collinear Points](https://www.coursera.org/learn/algorithms-part1/programming/prXiW/collinear-points)

[5小时](https://www.coursera.org/learn/algorithms-part1/programming/prXiW/collinear-points)

* **Quicksort**

编程作业: Collinear Points

您还未提交解答。您必须获得 80/100 分才能通过。

|  |  |
| --- | --- |
| **截止时间** | Pass this assignment by 二月 12, 11:59 晚上 PST |

1. [说明](https://www.coursera.org/learn/algorithms-part1/programming/prXiW/collinear-points)
2. [**我提交的作业**](https://www.coursera.org/learn/algorithms-part1/programming/prXiW/collinear-points/submission)
3. [讨论](https://www.coursera.org/learn/algorithms-part1/programming/prXiW/collinear-points/discussions)

  Create submission

Your Submissions

**日期**

**分数**

**通过了吗？**

23 一月 2017 在 3:33 下午

59/100

否

Collinear Points

59/100

隐藏 评分反馈

See the Assessment Guide for information on how to interpret this report.

ASSESSMENT SUMMARY

Compilation: FAILED (0 errors, 4 warnings)

API: PASSED

Findbugs: FAILED (5 warnings)

Checkstyle: FAILED (14 warnings)

Correctness: 22/41 tests passed

Memory: 1/1 tests passed

Timing: 23/41 tests passed

Aggregate score: 58.41%

[Compilation: 5%, API: 5%, Findbugs: 0%, Checkstyle: 0%, Correctness: 60%, Memory: 10%, Timing: 20%]

ASSESSMENT DETAILS

The following files were submitted:

----------------------------------

2.7K Jan 23 07:33 BruteCollinearPoints.java

4.0K Jan 23 07:33 FastCollinearPoints.java

4.8K Jan 23 07:33 Point.java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* COMPILING

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% javac Point.java

\*-----------------------------------------------------------

% javac BruteCollinearPoints.java

\*-----------------------------------------------------------

BruteCollinearPoints.java:15: warning: [rawtypes] found raw type: ArrayList

private List<LineSegment> lslist = new ArrayList();

^

missing type arguments for generic class ArrayList<E>

where E is a type-variable:

E extends Object declared in class ArrayList

BruteCollinearPoints.java:15: warning: [unchecked] unchecked conversion

private List<LineSegment> lslist = new ArrayList();

^

required: List<LineSegment>

found: ArrayList

2 warnings

% javac FastCollinearPoints.java

\*-----------------------------------------------------------

FastCollinearPoints.java:16: warning: [rawtypes] found raw type: ArrayList

private List<LineSegment> lslist = new ArrayList();

^

missing type arguments for generic class ArrayList<E>

where E is a type-variable:

E extends Object declared in class ArrayList

FastCollinearPoints.java:16: warning: [unchecked] unchecked conversion

private List<LineSegment> lslist = new ArrayList();

^

required: List<LineSegment>

found: ArrayList

2 warnings

================================================================

Checking the APIs of your programs.

\*-----------------------------------------------------------

Point:

BruteCollinearPoints:

FastCollinearPoints:

================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* CHECKING STYLE AND COMMON BUG PATTERNS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% findbugs \*.class

\*-----------------------------------------------------------

M V EI\_EXPOSE\_REP EI: Returns a reference to the mutable object stored in the instance variable 'ls', which exposes the internal representation of the class 'BruteCollinearPoints'. Instead, create a defensive copy of the object referenced by 'ls' and return the copy. At BruteCollinearPoints.java:[line 50]

L D FE\_FLOATING\_POINT\_EQUALITY FE: Tests for exact floating-point equality. Because floating-point calculations may involve rounding, the calculated values may be imprecise. At BruteCollinearPoints.java:[line 32]

L B NM\_CLASS\_NAMING\_CONVENTION Nm: The class name 'Point$slopeComparator' doesn't start with an upper-case letter. At Point.java:[lines 108-116]

M V EI\_EXPOSE\_REP EI: Returns a reference to the mutable object stored in the instance variable 'ls', which exposes the internal representation of the class 'FastCollinearPoints'. Instead, create a defensive copy of the object referenced by 'ls' and return the copy. At FastCollinearPoints.java:[line 80]

L D FE\_FLOATING\_POINT\_EQUALITY FE: Tests for exact floating-point equality. Because floating-point calculations may involve rounding, the calculated values may be imprecise. At FastCollinearPoints.java:[line 54]

Warnings generated: 5

================================================================

% checkstyle \*.java

\*-----------------------------------------------------------

Point.java:13:8: Unused import statement for 'edu.princeton.cs.algs4.StdOut'. [UnusedImports]

Point.java:108:19: The class 'slopeComparator' must start with an uppercase letter and use CamelCase. [TypeName]

BruteCollinearPoints.java:1:8: Unused import statement for 'edu.princeton.cs.algs4.In'. [UnusedImports]

BruteCollinearPoints.java:2:8: Unused import statement for 'edu.princeton.cs.algs4.StdDraw'. [UnusedImports]

BruteCollinearPoints.java:3:8: Unused import statement for 'edu.princeton.cs.algs4.StdOut'. [UnusedImports]

BruteCollinearPoints.java:6:8: Unused import statement for 'java.util.Comparator'. [UnusedImports]

BruteCollinearPoints.java:29:16: 'for' is not followed by whitespace. [WhitespaceAfter]

BruteCollinearPoints.java:63:3: The comment is empty. [IllegalTokenText]

BruteCollinearPoints.java:72:3: The comment is empty. [IllegalTokenText]

FastCollinearPoints.java:6:8: Unused import statement for 'edu.princeton.cs.algs4.In'. [UnusedImports]

FastCollinearPoints.java:7:8: Unused import statement for 'edu.princeton.cs.algs4.StdOut'. [UnusedImports]

FastCollinearPoints.java:8:8: Unused import statement for 'edu.princeton.cs.algs4.StdDraw'. [UnusedImports]

FastCollinearPoints.java:93:3: The comment is empty. [IllegalTokenText]

FastCollinearPoints.java:102:3: The comment is empty. [IllegalTokenText]

Checkstyle ends with 14 errors.

================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TESTING CORRECTNESS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Testing correctness of Point

\*-----------------------------------------------------------

Running 3 total tests.

Test 1: p.slopeTo(q)

\* positive infinite slope, where p and q have coordinates in [0, 500)

\* positive infinite slope, where p and q have coordinates in [0, 32768)

\* negative infinite slope, where p and q have coordinates in [0, 500)

\* negative infinite slope, where p and q have coordinates in [0, 32768)

\* positive zero slope, where p and q have coordinates in [0, 500)

\* positive zero slope, where p and q have coordinates in [0, 32768)

\* symmetric for random points p and q with coordinates in [0, 500)

\* symmetric for random points p and q with coordinates in [0, 32768)

\* transitive for random points p, q, and r with coordinates in [0, 500)

\* transitive for random points p, q, and r with coordinates in [0, 32768)

\* slopeTo(), where p and q have coordinates in [0, 500)

\* slopeTo(), where p and q have coordinates in [0, 32768)

\* slopeTo(), where p and q have coordinates in [0, 10)

\* throw a java.lang.NullPointerException if argument is null

==> passed

Test 2: p.compareTo(q)

\* reflexive, where p and q have coordinates in [0, 500)

\* reflexive, where p and q have coordinates in [0, 32768)

\* antisymmetric, where p and q have coordinates in [0, 500)

\* antisymmetric, where p and q have coordinates in [0, 32768)

\* transitive, where p, q, and r have coordinates in [0, 500)

\* transitive, where p, q, and r have coordinates in [0, 32768)

\* sign of compareTo(), where p and q have coordinates in [0, 500)

\* sign of compareTo(), where p and q have coordinates in [0, 32768)

\* sign of compareTo(), where p and q have coordinates in [0, 10)

\* throw java.lang.NullPointerException exception if argument is null

==> passed

Test 3: p.slopeOrder().compare(q, r)

\* reflexive, where p and q have coordinates in [0, 500)

\* reflexive, where p and q have coordinates in [0, 32768)

\* antisymmetric, where p, q, and r have coordinates in [0, 500)

\* antisymmetric, where p, q, and r have coordinates in [0, 32768)

\* transitive, where p, q, r, and s have coordinates in [0, 500)

\* transitive, where p, q, r, and s have coordinates in [0, 32768)

\* sign of compare(), where p, q, and r have coordinates in [0, 500)

\* sign of compare(), where p, q, and r have coordinates in [0, 32768)

\* sign of compare(), where p, q, and r have coordinates in [0, 10)

\* throw java.lang.NullPointerException if either argument is null

==> passed

Total: 3/3 tests passed!

================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TESTING CORRECTNESS (substituting reference Point and LineSegment)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Testing correctness of BruteCollinearPoints

\*-----------------------------------------------------------

Running 17 total tests.

The inputs satisfy the following conditions:

- no duplicate points

- no 5 (or more) points are collinear

- all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file

\* filename = input8.txt

\* filename = equidistant.txt

\* filename = input40.txt

\* filename = input48.txt

==> passed

Test 2a: Points from a file with horizontal line segments

\* filename = horizontal5.txt

\* filename = horizontal25.txt

==> passed

Test 2b: Random horizontal line segments

\* 1 random horizontal line segment

\* 5 random horizontal line segments

\* 10 random horizontal line segments

\* 15 random horizontal line segments

==> passed

Test 3a: Points from a file with vertical line segments

\* filename = vertical5.txt

\* filename = vertical25.txt

==> passed

Test 3b: Random vertical line segments

\* 1 random vertical line segment

\* 5 random vertical line segments

\* 10 random vertical line segments

\* 15 random vertical line segments

==> passed

Test 4a: Points from a file with no line segments

\* filename = random23.txt

\* filename = random38.txt

==> passed

Test 4b: Random points with no line segments

\* 5 random points

\* 10 random points

\* 20 random points

\* 50 random points

==> passed

Test 5: Points from a file with fewer than 4 points

\* filename = input1.txt

\* filename = input2.txt

\* filename = input3.txt

==> passed

Test 6: Check for dependence on either compareTo() or compare()

returning { -1, +1, 0 } instead of { negative integer,

positive integer, zero }

\* filename = equidistant.txt

\* filename = input40.txt

\* filename = input48.txt

==> passed

Test 7: Check for fragile dependence on return value of toString()

\* filename = equidistant.txt

\* filename = input40.txt

\* filename = input48.txt

==> passed

Test 8: Random line segments, none vertical or horizontal

\* 1 random line segment

\* 5 random line segments

\* 10 random line segments

\* 15 random line segments

==> passed

Test 9: Random line segments

\* 1 random line segment

\* 5 random line segments

\* 10 random line segments

\* 15 random line segments

==> passed

Test 10: Check that data type is immutable by testing whether each method

returns the same value, regardless of any intervening operations

\* input8.txt

- failed after 15 operations involving BruteCollinearPoints

- first and last call to segments() returned different arrays

- sequence of operations was:

BruteCollinearPoints collinear = new BruteCollinearPoints(points);

collinear.segments()

collinear.numberOfSegments() -> 2

collinear.numberOfSegments() -> 2

collinear.numberOfSegments() -> 2

collinear.numberOfSegments() -> 2

mutate points[] array that was passed to constructor

collinear.segments()

collinear.numberOfSegments() -> 2

mutate points[] array that was passed to constructor

mutate points[] array that was passed to constructor

collinear.segments()

collinear.numberOfSegments() -> 2

mutate array returned by last call to segments()

collinear.segments()

- failed on trial 1 of 100

\* equidistant.txt

- failed after 16 operations involving BruteCollinearPoints

- first and last call to segments() returned different arrays

- sequence of operations was:

BruteCollinearPoints collinear = new BruteCollinearPoints(points);

collinear.numberOfSegments() -> 4

mutate points[] array that was passed to constructor

collinear.segments()

collinear.numberOfSegments() -> 4

collinear.numberOfSegments() -> 4

mutate points[] array that was passed to constructor

mutate points[] array that was passed to constructor

collinear.segments()

collinear.segments()

collinear.numberOfSegments() -> 4

collinear.numberOfSegments() -> 4

collinear.numberOfSegments() -> 4

mutate points[] array that was passed to constructor

mutate array returned by last call to segments()

collinear.segments()

- failed on trial 1 of 100

==> FAILED

Test 11: Check that data type does not mutate the constructor argument

\* input8.txt

- data type mutated the points[] array

- data type should have no side effects unless documented in API

\* equidistant.txt

- data type mutated the points[] array

- data type should have no side effects unless documented in API

==> FAILED

Test 12: numberOfSegments() is consistent with segments()

\* filename = input8.txt

\* filename = equidistant.txt

\* filename = input40.txt

\* filename = input48.txt

\* filename = horizontal5.txt

\* filename = vertical5.txt

\* filename = random23.txt

==> passed

Test 13: Throws exception either if argument to constructor is null

or if any entry in array is null

\* argument is null

\* Point[] of length 10, number of null entries = 1

\* Point[] of length 10, number of null entries = 10

\* Point[] of length 4, number of null entries = 1

\* Point[] of length 3, number of null entries = 1

\* Point[] of length 2, number of null entries = 1

\* Point[] of length 1, number of null entries = 1

==> passed

Test 14: Check that the constructor throws an exception if duplicate points

\* 50 points

- failed on trial 1 of 5

- does not throw a java.lang.IllegalArgumentException

\* 25 points

- failed on trial 1 of 10

- does not throw a java.lang.IllegalArgumentException

\* 5 points

- failed on trial 1 of 100

- does not throw a java.lang.IllegalArgumentException

5

11824 17720

22702 27146

6206 17142

25068 14868

6206 17142

\* 4 points

- failed on trial 1 of 100

- does not throw a java.lang.IllegalArgumentException

4

24656 16880

24420 3022

25540 25060

25540 25060

\* 3 points

- failed on trial 1 of 100

- does not throw a java.lang.IllegalArgumentException

3

22527 7

24261 25067

24261 25067

\* 2 points

- failed on trial 1 of 100

- does not throw a java.lang.IllegalArgumentException

2

13317 28660

13317 28660

==> FAILED

Total: 14/17 tests passed!

================================================================

Testing correctness of FastCollinearPoints

\*-----------------------------------------------------------

Running 21 total tests.

The inputs satisfy the following conditions:

- no duplicate points

- all x- and y-coordinates between 0 and 32,767

Test 1: Points from a file

\* filename = input8.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (10000, 0) -> (7000, 3000)

- reference segment 0: (10000, 0) -> (7000, 3000) -> (3000, 7000) -> (0, 10000)

- student solution has 3 non-null entries

- reference solution has 2 non-null entries

- 3 extra entries in student solution, including: (6000, 7000) -> (14000, 15000)

- 2 missing entries in student solution, including: (3000, 4000) -> (6000, 7000) -> (14000, 15000) -> (20000, 21000)

\* filename = equidistant.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 1: (10000, 0) -> (13000, 0)

- reference segment 3: (10000, 0) -> (13000, 0) -> (20000, 0) -> (30000, 0)

- student solution has 3 non-null entries

- reference solution has 4 non-null entries

- 1 extra entry in student solution: (10000, 0) -> (13000, 0)

- 2 missing entries in student solution, including: (30000, 0) -> (20000, 10000) -> (10000, 20000) -> (0, 30000)

\* filename = input40.txt

- student solution has 0 non-null entries

- reference solution has 4 non-null entries

- 4 missing entries in student solution, including: (2000, 29000) -> (4000, 29000) -> (22000, 29000) -> (28000, 29000)

\* filename = input48.txt

- student solution has 0 non-null entries

- reference solution has 6 non-null entries

- 6 missing entries in student solution, including: (1000, 26000) -> (9000, 26000) -> (11000, 26000) -> (18000, 26000)

\* filename = input299.txt

- student solution has 0 non-null entries

- reference solution has 6 non-null entries

- 6 missing entries in student solution, including: (3250, 17450) -> (8500, 17450) -> (13400, 17450) -> (17250, 17450)

==> FAILED

Test 2a: Points from a file with horizontal line segments

\* filename = horizontal5.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (4750, 4652) -> (5766, 4652)

- reference segment 0: (4750, 4652) -> (5766, 4652) -> (9972, 4652) -> (16307, 4652)

- student solution has 3 non-null entries

- reference solution has 5 non-null entries

- 3 extra entries in student solution, including: (2682, 14118) -> (5067, 14118)

- 5 missing entries in student solution, including: (2682, 14118) -> (5067, 14118) -> (7453, 14118) -> (7821, 14118)

\* filename = horizontal25.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (10021, 1256) -> (16419, 1256)

- reference segment 0: (10021, 1256) -> (13505, 1256) -> (16419, 1256) -> (19615, 1256)

- student solution has 5 non-null entries

- reference solution has 25 non-null entries

- 5 extra entries in student solution, including: (6268, 18593) -> (11710, 18593)

- 25 missing entries in student solution, including: (8784, 20913) -> (9880, 20913) -> (16352, 20913) -> (19666, 20913)

\* filename = horizontal50.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 1: (12567, 11991) -> (15839, 11991)

- reference segment 29: (12567, 11991) -> (15839, 11991) -> (16171, 11991) -> (19205, 11991)

- student solution has 3 non-null entries

- reference solution has 50 non-null entries

- 2 extra entries in student solution, including: (11973, 17473) -> (15363, 17473)

- 49 missing entries in student solution, including: (5249, 20754) -> (5559, 20754) -> (14800, 20754) -> (17428, 20754)

\* filename = horizontal75.txt

- segments() contains the same segment more than once

- segment 1: (20739, 9681) -> (19367, 9681)

- segment 2: (20739, 9681) -> (19367, 9681)

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (7903, 1061) -> (17515, 1061)

- reference segment 0: (7903, 1061) -> (8084, 1061) -> (17515, 1061) -> (20959, 1061)

- student solution has 8 non-null entries

- reference solution has 75 non-null entries

- 7 extra entries in student solution, including: (5184, 18549) -> (14760, 18549)

- 74 missing entries in student solution, including: (1536, 20976) -> (6545, 20976) -> (14178, 20976) -> (14591, 20976)

\* filename = horizontal100.txt

- segments() contains the same segment more than once

- segment 3: (20409, 10339) -> (3798, 10339)

- segment 4: (20409, 10339) -> (3798, 10339)

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (1090, 1279) -> (8690, 1279)

- reference segment 0: (1090, 1279) -> (7354, 1279) -> (8690, 1279) -> (13975, 1279)

- student solution has 5 non-null entries

- reference solution has 100 non-null entries

- 2 extra entries in student solution, including: (20409, 10339) -> (3798, 10339)

- 97 missing entries in student solution, including: (5835, 20698) -> (7673, 20698) -> (16154, 20698) -> (19642, 20698)

==> FAILED

Test 2b: Random horizontal line segments

\* 1 random horizontal line segment

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (2578, 19675) -> (15337, 19675)

- reference segment 0: (2578, 19675) -> (14368, 19675) -> (15337, 19675) -> (17415, 19675)

- student solution has 1 non-null entries

- reference solution has 1 non-null entries

- 1 extra entry in student solution: (2578, 19675) -> (15337, 19675)

- 1 missing entry in student solution: (2578, 19675) -> (14368, 19675) -> (15337, 19675) -> (17415, 19675)

- failed on trial 1 of 500

4

2578 19675

14368 19675

17415 19675

15337 19675

\* 5 random horizontal line segments

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (2767, 1714) -> (3125, 1714)

- reference segment 0: (2767, 1714) -> (3125, 1714) -> (8795, 1714) -> (19566, 1714)

- student solution has 3 non-null entries

- reference solution has 5 non-null entries

- 3 extra entries in student solution, including: (6252, 8645) -> (14591, 8645)

- 5 missing entries in student solution, including: (1479, 14203) -> (3996, 14203) -> (15016, 14203) -> (17677, 14203)

- failed on trial 1 of 250

20

16236 8645

8795 1714

15016 14203

6252 8645

14591 8645

19566 1714

15910 7078

3125 1714

17677 14203

20388 7078

6030 8472

19300 8472

2767 1714

1479 14203

18324 8472

16841 8645

14103 7078

8825 8472

13777 7078

3996 14203

\* 10 random horizontal line segments

- segments() contains a subsegment of a segment in reference solution

- student segment 1: (20150, 4291) -> (10150, 4291)

- reference segment 1: (9038, 4291) -> (9355, 4291) -> (10150, 4291) -> (20150, 4291)

- student solution has 2 non-null entries

- reference solution has 10 non-null entries

- 1 extra entry in student solution: (20150, 4291) -> (10150, 4291)

- 9 missing entries in student solution, including: (2897, 18093) -> (9797, 18093) -> (12512, 18093) -> (17658, 18093)

- failed on trial 1 of 50

\* 15 random horizontal line segments

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (6178, 2133) -> (7531, 2133)

- reference segment 0: (6178, 2133) -> (7531, 2133) -> (9665, 2133) -> (16762, 2133)

- student solution has 5 non-null entries

- reference solution has 15 non-null entries

- 4 extra entries in student solution, including: (8370, 13398) -> (12503, 13398)

- 14 missing entries in student solution, including: (1636, 19048) -> (2580, 19048) -> (15539, 19048) -> (16460, 19048)

- failed on trial 1 of 5

==> FAILED

Test 3a: Points from a file with vertical line segments

\* filename = vertical5.txt

- segments() contains the same segment more than once

- segment 1: (14407, 10367) -> (14407, 19953)

- segment 4: (14407, 10367) -> (14407, 19953)

- segments() contains a subsegment of a segment in reference solution

- student segment 2: (14407, 10367) -> (14407, 17188)

- reference segment 3: (14407, 10367) -> (14407, 17188) -> (14407, 17831) -> (14407, 19953)

- student solution has 5 non-null entries

- reference solution has 5 non-null entries

- 2 extra entries in student solution, including: (14407, 10367) -> (14407, 19953)

- 2 missing entries in student solution, including: (5757, 3426) -> (5757, 13581) -> (5757, 16647) -> (5757, 20856)

\* filename = vertical25.txt

- student solution has 4 non-null entries

- reference solution has 25 non-null entries

- 21 missing entries in student solution, including: (7674, 9568) -> (7674, 11299) -> (7674, 17451) -> (7674, 18802)

\* filename = vertical50.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (10209, 1143) -> (10209, 8678)

- reference segment 25: (10209, 1143) -> (10209, 8678) -> (10209, 12125) -> (10209, 12199)

- student solution has 4 non-null entries

- reference solution has 50 non-null entries

- 2 extra entries in student solution, including: (5991, 15606) -> (5991, 20227)

- 48 missing entries in student solution, including: (5991, 15606) -> (5991, 20227) -> (5991, 20340) -> (5991, 20424)

\* filename = vertical75.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 1: (9242, 3732) -> (9242, 9054)

- reference segment 24: (9242, 3732) -> (9242, 9054) -> (9242, 12301) -> (9242, 17315)

- student solution has 3 non-null entries

- reference solution has 75 non-null entries

- 2 extra entries in student solution, including: (9242, 3732) -> (9242, 9054)

- 74 missing entries in student solution, including: (13780, 16365) -> (13780, 17494) -> (13780, 18253) -> (13780, 18908)

\* filename = vertical100.txt

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (13469, 1049) -> (13469, 4082)

- reference segment 60: (13469, 1049) -> (13469, 4082) -> (13469, 10837) -> (13469, 20173)

- student solution has 2 non-null entries

- reference solution has 100 non-null entries

- 2 extra entries in student solution, including: (1709, 1114) -> (1709, 16277)

- 100 missing entries in student solution, including: (15135, 15588) -> (15135, 17398) -> (15135, 19476) -> (15135, 20439)

==> FAILED

Test 3b: Random vertical line segments

\* 1 random vertical line segment

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (20033, 5613) -> (20033, 8244)

- reference segment 0: (20033, 5613) -> (20033, 6693) -> (20033, 8244) -> (20033, 13892)

- student solution has 1 non-null entries

- reference solution has 1 non-null entries

- 1 extra entry in student solution: (20033, 5613) -> (20033, 8244)

- 1 missing entry in student solution: (20033, 5613) -> (20033, 6693) -> (20033, 8244) -> (20033, 13892)

- failed on trial 2 of 500

4

20033 13892

20033 6693

20033 5613

20033 8244

\* 5 random vertical line segments

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (18592, 1541) -> (18592, 15101)

- reference segment 4: (18592, 1541) -> (18592, 15101) -> (18592, 15235) -> (18592, 15483)

- student solution has 2 non-null entries

- reference solution has 5 non-null entries

- 1 extra entry in student solution: (18592, 1541) -> (18592, 15101)

- 4 missing entries in student solution, including: (4750, 11337) -> (4750, 14904) -> (4750, 15494) -> (4750, 17589)

- failed on trial 1 of 250

20

4750 14904

3193 1664

18592 15235

9423 2713

9423 15016

18592 15483

4750 15494

10947 5720

3193 10326

10947 11335

4750 17589

18592 1541

4750 11337

3193 17380

9423 10611

10947 20913

9423 17752

3193 3478

10947 12769

18592 15101

\* 10 random vertical line segments

- student solution has 2 non-null entries

- reference solution has 10 non-null entries

- 8 missing entries in student solution, including: (17887, 11211) -> (17887, 12656) -> (17887, 12728) -> (17887, 16150)

- failed on trial 1 of 50

\* 15 random vertical line segments

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (19367, 1445) -> (19367, 5975)

- reference segment 12: (19367, 1445) -> (19367, 3282) -> (19367, 5975) -> (19367, 11048)

- student solution has 1 non-null entries

- reference solution has 15 non-null entries

- 1 extra entry in student solution: (19367, 1445) -> (19367, 5975)

- 15 missing entries in student solution, including: (18323, 12128) -> (18323, 16837) -> (18323, 17559) -> (18323, 18295)

- failed on trial 1 of 5

==> FAILED

Test 4a: Points from a file with no line segments

\* filename = random23.txt

\* filename = random38.txt

\* filename = random91.txt

\* filename = random152.txt

==> passed

Test 4b: Random points with no line segments

\* 5 random points

\* 10 random points

\* 20 random points

\* 50 random points

==> passed

Test 5a: Points from a file with 5 or more on some line segments

\* filename = input9.txt

- segments() contains the same segment more than once

- segment 0: (1000, 1000) -> (2000, 2000)

- segment 1: (1000, 1000) -> (2000, 2000)

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (1000, 1000) -> (2000, 2000)

- reference segment 0: (1000, 1000) -> (2000, 2000) -> (3000, 3000) -> (4000, 4000) -> (5000, 5000) -> (6000, 6000) -> (7000, 7000) -> (8000, 8000) -> (9000, 9000)

- student solution has 36 non-null entries

- reference solution has 1 non-null entries

- 35 extra entries in student solution, including: (7000, 7000) -> (9000, 9000)

\* filename = input10.txt

- segments() contains the same segment more than once

- segment 1: (1000, 18000) -> (4000, 30000)

- segment 2: (1000, 18000) -> (4000, 30000)

- segments() contains a subsegment of a segment in reference solution

- student segment 0: (3500, 28000) -> (1000, 18000)

- reference segment 1: (1000, 18000) -> (2000, 22000) -> (3000, 26000) -> (3500, 28000) -> (4000,

...

WARNING: the grading output was truncated due to excessive length.

Typically, this is because you have a method that has an unanticipated side effect

(such as printing to standard output or throwing an exception). A large amount of output

can also arise from failing many tests.

23 一月 2017 在 3:26 下午

38/100

否