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

ASSESSMENT SUMMARY

Compilation: FAILED (0 errors, 8 warnings)

API: PASSED

Findbugs: FAILED (3 warnings)

PMD: FAILED (8 warnings)

Checkstyle: FAILED (0 errors, 23 warnings)

Correctness: 11/27 tests passed

Memory: 8/8 tests passed

Timing: 14/42 tests passed

Aggregate score: 46.11%

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

ASSESSMENT DETAILS

The following files were submitted:

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

10K Jun 23 15:11 KdTree.java

4.0K Jun 23 15:11 PointSET.java

% pmd \*.java

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

KdTree.java:16: Avoid unused private fields such as 'key'. [UnusedPrivateField]

KdTree.java:26: Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes. [FieldDeclarationsShouldBeAtStartOfClass]

KdTree.java:27: Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes. [FieldDeclarationsShouldBeAtStartOfClass]

KdTree.java:28: Avoid unused private fields such as 'mVertical'. [UnusedPrivateField]

KdTree.java:28: Fields should be declared at the top of the class, before any method declarations, constructors, initializers or inner classes. [FieldDeclarationsShouldBeAtStartOfClass]

KdTree.java:28: Possible unsafe assignment to a non-final static field in a constructor. [AssignmentToNonFinalStatic]

KdTree.java:69: Avoid throwing null pointer exceptions. [AvoidThrowingNullPointerException]

KdTree.java:184: Avoid throwing null pointer exceptions. [AvoidThrowingNullPointerException]

PMD ends with 8 warnings.

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

% checkstyle \*.java

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

[WARN] PointSET.java:1:3: '//' or '/\*' is not followed by whitespace. [IllegalTokenText]

[WARN] PointSET.java:3: Do not use .\* in import statements. [AvoidStarImport]

[WARN] PointSET.java:100:74: ',' is not followed by whitespace. [WhitespaceAfter]

[WARN] PointSET.java:101:96: ',' is preceded with whitespace. [NoWhitespaceBefore]

[WARN] PointSET.java:101:98: ',' is not followed by whitespace. [WhitespaceAfter]

[WARN] KdTree.java:1:3: '//' or '/\*' is not followed by whitespace. [IllegalTokenText]

[WARN] KdTree.java:3: Do not use .\* in import statements. [AvoidStarImport]

[WARN] KdTree.java:28:5: Declare static variables before instance variables, constructors, and methods. [DeclarationOrder]

Checkstyle ends with 0 errors and 8 warnings.

% custom checkstyle checks for PointSET.java

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

[WARN] PointSET.java:41:30: The numeric literal '0.01' appears to be unnecessary. [NumericLiteral]

[WARN] PointSET.java:68:29: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] PointSET.java:69:50: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

Checkstyle ends with 0 errors and 3 warnings.

% custom checkstyle checks for KdTree.java

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

[WARN] KdTree.java:3: Declaring 1 non-final static variables ('mVertical') suggests poor design in this class. [StaticVariableCount]

[WARN] KdTree.java:134:30: The numeric literal '0.001' appears to be unnecessary. [NumericLiteral]

[WARN] KdTree.java:157:9: You should not need to use the 'TreeSet' data type in this program. [IllegalType]

[WARN] KdTree.java:164:48: You should not need to use the 'TreeSet' data type in this program. [IllegalType]

[WARN] KdTree.java:193:30: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] KdTree.java:193:67: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] KdTree.java:197:37: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] KdTree.java:197:67: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] KdTree.java:211:34: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] KdTree.java:211:67: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] KdTree.java:221:46: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

[WARN] KdTree.java:221:80: Do not call 'distanceTo()' in this program; instead use 'distanceSquaredTo()'. [IllegalMethodCall]

Checkstyle ends with 0 errors and 12 warnings.

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

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

\* TESTING CORRECTNESS

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

Testing correctness of PointSET

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

Running 8 total tests.

A point in an m-by-m grid means that it is of the form (i/m, j/m),

where i and j are integers between 0 and m

Test 1: Test size() by inserting n random points

(size may be less than n because of duplicates)

\* 100000 random points in a 100000-by-100000 grid

\* 100000 random points in a 10000-by-10000 grid

\* 100000 random points in a 1000-by-1000 grid

\* 100000 random points in a 100-by-100 grid

\* 100000 random points in a 10-by-10 grid

==> passed

Test 2: Test isEmpty() by checking for n = 0, 1, and 2 points

\* 0 points

\* 1 point

\* 2 points

==> passed

Test 3: Insert n random points and check contains() for random query points

\* 100000 random points in a 100000-by-100000 grid

\* 100000 random points in a 10000-by-10000 grid

\* 100000 random points in a 1000-by-1000 grid

\* 100000 random points in a 100-by-100 grid

\* 100000 random points in a 10-by-10 grid

==> passed

Test 4: Insert n random points and check nearest() for random query points

\* 1000 random points in a 100000-by-100000 grid

\* 1000 random points in a 10000-by-10000 grid

\* 1000 random points in a 1000-by-1000 grid

\* 1000 random points in a 100-by-100 grid

\* 1000 random points in a 10-by-10 grid

==> passed

Test 5: Insert n random points and check range() for random query rectangles

\* 1000 random rectangles and points in a 100000-by-100000 grid

\* 1000 random rectangles and points in a 10000-by-10000 grid

\* 1000 random rectangles and points in a 1000-by-1000 grid

\* 1000 random rectangles and points in a 100-by-100 grid

\* 1000 random rectangles and points in a 10-by-10 grid

==> passed

Test 6: Intermixed sequence of calls to isEmpty(), size(), insert(),

contains(), range(), and nearest() with probabilities

p1, p2, p3, p4, p5, and p6, respectively

\* 10000 calls with random points in a 10000-by-10000 grid

and probabilities 0.05, 0.05, 0.3, 0.2, 0.2, 0.2

\* 10000 calls with random points in a 1000-by-1000 grid

and probabilities 0.05, 0.05, 0.3, 0.2, 0.2, 0.2

\* 10000 calls with random points in a 100-by-100 grid

and probabilities 0.05, 0.05, 0.3, 0.2, 0.2, 0.2

\* 10000 calls with random points in a 10-by-10 grid

and probabilities 0.05, 0.05, 0.3, 0.2, 0.2, 0.2

\* 10000 calls with random points in a 1-by-1 grid

and probabilities 0.05, 0.05, 0.3, 0.2, 0.2, 0.2

==> passed

Test 7: Intermixed sequence of calls to isEmpty(), size(), insert(),

contains(), range(), and nearest() with probabilities

p1, p2, p3=0, p4, p5, and p6, respectively

(data structure with 0 points)

\* 1000 calls with random points in a 1000-by-1000 grid

and probabilities 0.5, 0.5, 0.0, 0.0, 0.0, 0.0

\* 1000 calls with random points in a 1000-by-1000 grid

and probabilities 0.2, 0.2, 0.0, 0.6, 0.0, 0.0

\* 1000 calls with random points in a 1000-by-1000 grid

and probabilities 0.2, 0.2, 0.0, 0.0, 0.6, 0.0

\* 1000 calls with random points in a 1000-by-1000 grid

and probabilities 0.2, 0.2, 0.0, 0.0, 0.0, 0.6

- failed on trial 1 of 1000

- student nearest() = (-1.0, -1.0)

- reference nearest() = null

- (-1.0, -1.0) was not inserted into data structure

- student distanceTo() = 2.438974579613326

\* 1000 calls with random points in a 1000-by-1000 grid

and probabilities 0.2, 0.2, 0.0, 0.2, 0.2, 0.2

- failed on trial 8 of 1000

- student nearest() = (-1.0, -1.0)

- reference nearest() = null

- (-1.0, -1.0) was not inserted into data structure

- student distanceTo() = 2.32510859961422

==> FAILED

Test 8: Test whether two PointSET objects can be created at the same time

==> passed

Total: 7/8 tests passed!

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

Testing correctness of KdTree

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

In the tests below, we consider three classes of points and rectangles.

\* Non-degenerate points: no two points (or rectangles) share either an

x-coordinate or a y-coordinate

\* Distinct points: no two points (or rectangles) share both an

x-coordinate and a y-coordinate

\* General points: no restrictions on the x-coordinates or y-coordinates

of the points (or rectangles)

A point in an GRID-by-GRID grid means that it is of the form (i/GRID, j/GRID),

where i and j are integers between 0 and GRID

Running 19 total tests.

Test 1a: Insert n non-degenerate points and check size() after each insertion

\* 50000 random non-degenerate points in a 100000-by-100000 grid

- failed on trial 1 of 50000

- student size() = 0

- reference size() = 1

\* 5000 random non-degenerate points in a 10000-by-10000 grid

- failed on trial 1 of 5000

- student size() = 0

- reference size() = 1

\* 500 random non-degenerate points in a 1000-by-1000 grid

- failed on trial 1 of 500

- student size() = 0

- reference size() = 1

\* 50 random non-degenerate points in a 100-by-100 grid

- failed on trial 1 of 50

- student size() = 0

- reference size() = 1

\* 10 random non-degenerate points in a 10-by-10 grid

- failed on trial 1 of 10

- student size() = 0

- reference size() = 1

\* 1 random non-degenerate points in a 1-by-1 grid

- failed on trial 1 of 1

- student size() = 0

- reference size() = 1

==> FAILED

Test 1b: Insert n distinct points and check size() after each insertion

\* 100000 random distinct points in a 100000-by-100000 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 100000 random distinct points in a 10000-by-10000 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 100000 random distinct points in a 1000-by-1000 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 10000 random distinct points in a 100-by-100 grid

- failed on trial 1 of 10000

- student size() = 0

- reference size() = 1

\* 100 random distinct points in a 10-by-10 grid

- failed on trial 1 of 100

- student size() = 0

- reference size() = 1

\* 1 random distinct points in a 1-by-1 grid

- failed on trial 1 of 1

- student size() = 0

- reference size() = 1

==> FAILED

Test 1c: Insert n general points and check size() after each insertion

\* 100000 random general points in a 100000-by-100000 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 100000 random general points in a 10000-by-10000 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 100000 random general points in a 1000-by-1000 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 100000 random general points in a 100-by-100 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 100000 random general points in a 10-by-10 grid

- failed on trial 1 of 100000

- student size() = 0

- reference size() = 1

\* 10 random general points in a 1-by-1 grid

- failed on trial 1 of 10

- student size() = 0

- reference size() = 1

==> FAILED

Test 2: Test size() and isEmpty() for n = 0, 1, and 2 points

\* 0 points

\* 1 point

- student isEmpty() = true

- student size() = 0

- reference isEmpty() = false

- reference size() = 1

\* 2 points

- student isEmpty() = true

- student size() = 0

- reference isEmpty() = false

- reference size() = 2

==> FAILED

Test 3a: Insert n non-degenerate points and call contains() with random query points

\* 50000 random non-degenerate points in a 100000-by-100000 grid

- failed on trial 186 of 50000

- student contains() = true

- reference contains() = false

\* 5000 random non-degenerate points in a 10000-by-10000 grid

- failed on trial 21 of 5000

- student contains() = true

- reference contains() = false

\* 500 random non-degenerate points in a 1000-by-1000 grid

- failed on trial 78 of 500

- student contains() = true

- reference contains() = false

\* 50 random non-degenerate points in a 100-by-100 grid

- failed on trial 8 of 50

- student contains() = true

- reference contains() = false

\* 5 random non-degenerate points in a 10-by-10 grid

\* 1 random non-degenerate points in a 1-by-1 grid

==> FAILED

Test 3b: Insert n distinct points and call contains() with random query points

\* 100000 random distinct points in a 100000-by-100000 grid

- failed on trial 34 of 100000

- student contains() = true

- reference contains() = false

\* 100000 random distinct points in a 10000-by-10000 grid

- failed on trial 8 of 100000

- student contains() = true

- reference contains() = false

\* 100000 random distinct points in a 1000-by-1000 grid

- failed on trial 1 of 100000

- student contains() = true

- reference contains() = false

\* 10000 random distinct points in a 100-by-100 grid

- failed on trial 88 of 10000

- student contains() = true

- reference contains() = false

\* 100 random distinct points in a 10-by-10 grid

- failed on trial 10 of 100

- student contains() = true

- reference contains() = false

\* 1 random distinct points in a 1-by-1 grid

==> FAILED

Test 3c: Insert n general points and call contains() with random query points

\* 10000 random general points in a 1000-by-1000 grid

- failed on trial 2 of 10000

- student contains() = true

- reference contains() = false

\* 10000 random general points in a 100-by-100 grid

- failed on trial 3 of 10000

- student contains() = true

- reference contains() = false

\* 10000 random general points in a 10-by-10 grid

\* 10000 random general points in a 1-by-1 grid

==> FAILED

Test 4: Test whether two KdTree objects can be created at the same time

==> passed

Test 5a: Insert n non-degenerate points and call range() for n random query rectangles

\* 5000 random non-degenerate points and rectangles in a 100000-by-100000 grid

\* 5000 random non-degenerate points and rectangles in a 10000-by-10000 grid

\* 500 random non-degenerate points and rectangles in a 1000-by-1000 grid

\* 50 random non-degenerate points and rectangles in a 100-by-100 grid

\* 5 random non-degenerate points and rectangles in a 10-by-10 grid

==> passed

Test 5b: Insert n distinct points and call range() for n random query rectangles

\* 5000 random distinct points and rectangles in a 100000-by-100000 grid

- number of entries in student solution: 390

- number of entries in reference solution: 391

- 1 missing entry in student solution: '(0.81368, 0.77682)'

- failed on trial 4 of 5000

- rectangle = [0.7139, 0.98678] x [0.66428, 0.94093]

\* 5000 random distinct points and rectangles in a 10000-by-10000 grid

- number of entries in student solution: 92

- number of entries in reference solution: 95

- 3 missing entries in student solution, including: '(0.4071, 0.8546)'

- failed on trial 2 of 5000

- rectangle = [0.402, 0.5136] x [0.6951, 0.8948]

\* 1000 random distinct points and rectangles in a 1000-by-1000 grid

- number of entries in student solution: 266

- number of entries in reference solution: 284

- 18 missing entries in student solution, including: '(0.794, 0.709)'

- failed on trial 1 of 1000

- rectangle = [0.563, 0.946] x [0.069, 0.8]

\* 1000 random distinct points and rectangles in a 100-by-100 grid

- number of entries in student solution: 29

- number of entries in reference solution: 54

- 25 missing entries in student solution, including: '(0.55, 0.76)'

- failed on trial 1 of 1000

- rectangle = [0.47, 0.63] x [0.52, 0.76]

\* 5 random distinct points and rectangles in a 10-by-10 grid

==> FAILED

Test 5c: Insert n general points and call range() for n random query rectangles

\* 5000 random general points and rectangles in a 10000-by-10000 grid

- number of entries in student solution: 99

- number of entries in reference solution: 101

- 2 missing entries in student solution, including: '(0.4612, 0.5105)'

- failed on trial 1 of 5000

- rectangle = [0.3386, 0.4738] x [0.3736, 0.5176]

\* 5000 random general points and rectangles in a 1000-by-1000 grid

- number of entries in student solution: 410

- number of entries in reference solution: 476

- 66 missing entries in student solution, including: '(0.752, 0.819)'

- failed on trial 1 of 5000

- rectangle = [0.305, 0.895] x [0.653, 0.819]

\* 5000 random general points and rectangles in a 100-by-100 grid

- number of entries in student solution: 228

- number of entries in reference solution: 545

- 317 missing entries in student solution, including: '(0.43, 0.55)'

- failed on trial 1 of 5000

- rectangle = [0.16, 0.43] x [0.05, 0.55]

\* 5000 random general points and rectangles in a 10-by-10 grid

- number of entries in student solution: 24

- number of entries in reference solution: 72

- 48 missing entries in student solution, including: '(0.7, 0.8)'

- failed on trial 1 of 5000

- rectangle = [0.0, 0.7] x [0.0, 0.8]

\* 5000 random general points and rectangles in a 1-by-1 grid

- number of entries in student solution: 0

- number of entries in reference solution: 1

- 1 missing entry in student solution: '(1.0, 1.0)'

- failed on trial 3 of 5000

- rectangle = [1.0, 1.0] x [1.0, 1.0]

==> FAILED

Test 5d: Insert n points and call range() for tiny rectangles enclosing each point.

\* 4000 tiny rectangles and 4000 points in a 100000-by-100000 grid

- failed on trial 1640 of 4000

- rectangle query = [0.2379899999, 0.23799000010000002] x [0.4070199999, 0.4070200001]

- student range() size = 0

- reference range() size = 1

\* 4000 tiny rectangles and 4000 points in a 10000-by-10000 grid

- failed on trial 4 of 4000

- rectangle query = [0.7473999999, 0.7474000001] x [6.999999E-4, 7.000001E-4]

- student range() size = 0

- reference range() size = 1

\* 4000 tiny rectangles and 4000 points in a 1000-by-1000 grid

- failed on trial 1 of 3988

- rectangle query = [0.0139999999, 0.0140000001] x [9.999999E-4, 0.0010000001]

- student range() size = 0

- reference range() size = 1

\* 4000 tiny rectangles and 4000 points in a 100-by-100 grid

- failed on trial 1 of 3280

- rectangle query = [0.0, 1.0E-10] x [0.0, 1.0E-10]

- student range() size = 0

- reference range() size = 1

\* 4000 tiny rectangles and 4000 points in a 10-by-10 grid

- failed on trial 1 of 121

- rectangle query = [0.0, 1.0E-10] x [0.0, 1.0E-10]

- student range() size = 0

- reference range() size = 1

==> FAILED

Test 6a: Insert n non-degenerate points and call nearest() with random query points

\* 50000 random non-degenerate points in a 100000-by-100000 grid

\* 5000 random non-degenerate points in a 10000-by-10000 grid

\* 500 random non-degenerate points in a 1000-by-1000 grid

\* 50 random non-degenerate points in a 100-by-100 grid

\* 5 random non-degenerate points in a 10-by-10 grid

==> passed

Test 6b: Insert n distinct points and call nearest() with random query points

\* 50000 random distinct points in a 100000-by-100000 grid

- failed on trial 42 of 50000

- student nearest() = (0.34662, 0.39305)

- reference nearest() = (0.34662, 0.39379)

- student distanceTo() = 0.002531600284405069

- reference distanceTo() = 0.0020102238681301045

\* 10000 random distinct points in a 10000-by-10000 grid

- failed on trial 17 of 10000

- student nearest() = (0.2635, 0.0555)

- reference nearest() = (0.2655, 0.0522)

- student distanceTo() = 0.004788527957525137

- reference distanceTo() = 0.0024166091947188935

\* 10000 random distinct points in a 1000-by-1000 grid

- failed on trial 1 of 10000

- student nearest() = (0.102, 0.288)

- reference nearest() = (0.112, 0.272)

- student distanceTo() = 0.011401754250991356

- reference distanceTo() = 0.009055385138137424

\* 5000 random distinct points in a 100-by-100 grid

- failed on trial 1 of 5000

- student nearest() = (0.08, 0.47)

- reference nearest() = (0.07, 0.47)

- student distanceTo() = 0.014142135623730914

- reference distanceTo() = 0.009999999999999953

\* 50 random distinct points in a 10-by-10 grid

- failed on trial 17 of 50

- student nearest() = (0.0, 0.2)

- reference nearest() = (0.2, 0.3)

- student distanceTo() = 0.1414213562373095

- reference distanceTo() = 0.1

==> FAILED

Test 6c: Insert n general points and call nearest() with random query points

\* 10000 random general points in a 1000-by-1000 grid

- failed on trial 4 of 10000

- student nearest() = (0.145, 0.877)

- reference nearest() = (0.146, 0.875)

- student distanceTo() = 0.002236067977499792

- reference distanceTo() = 0.0014142135623730963

\* 10000 random general points in a 100-by-100 grid

- failed on trial 3 of 10000

- student nearest() = (0.83, 0.77)

- reference nearest() = (0.82, 0.76)

- student distanceTo() = 0.014142135623730963

- reference distanceTo() = 0.0

\* 10000 random general points in a 10-by-10 grid

- failed on trial 2 of 10000

- student nearest() = (0.6, 0.6)

- reference nearest() = (0.6, 0.5)

- student distanceTo() = 0.09999999999999998

- reference distanceTo() = 0.0

==> FAILED

Test 7a: Intermixed sequence of calls to insert(), isEmpty(), size(),

contains(), range(), and nearest() with probabilities

(p1, p2, p3, p4, p5, p6), respectively

\* 20000 calls with non-degenerate points in a 100000-by-100000 grid

and probabilities (0.3, 0.05, 0.05, 0.1, 0.2, 0.2)

- failed on trial 5 of 20000

- student size() = 0

- reference size() = 2

- sequence of dequeue operations was:

set.insert((0.81776, 0.71886))

set.insert((0.30541, 0.17243))

set.nearest((0.35469, 0.97975)) ==> (0.81776, 0.71886)

set.nearest((0.30165, 0.10363)) ==> (0.30541, 0.17243)

set.size() ==> 0

\* 20000 calls with non-degenerate points in a 10000-by-10000 grid

and probabilities (0.3, 0.05, 0.05, 0.1, 0.2, 0.2)

- failed on trial 5 of 20000

- student isEmpty() = true

- reference isEmpty() = false

- sequence of dequeue operations was:

set.insert((0.7337, 0.3495))

set.insert((0.26, 0.4558))

set.isEmpty() ==> true

\* 20000 calls with non-degenerate points in a 1000-by-1000 grid

and probabilities (0.3, 0.05, 0.05, 0.1, 0.2, 0.2)

- failed on trial 5 of 20000

- student isEmpty() = true

- reference isEmpty() = false

- sequence of dequeue operations was:

set.insert((0.779, 0.582))

set.insert((0.892, 0.608))

set.isEmpty() ==> true

\* 20000 calls with non-degenerate points in a 100-by-100 grid

and probabilities (0.3, 0.05, 0.05, 0.1, 0.2, 0.2)

- failed on trial 11 of 20000

- student isEmpty() = true

- reference isEmpty() = false

- sequence of dequeue operations was:

set.size() ==> 0

set.insert((0.1, 0.84))

set.insert((0.76, 0.09))

set.range([0.21, 0.96] x [0.37, 0.4])

set.contains((0.77, 0.78)) ==> false

set.insert((0.45, 0.58))

set.insert((0.85, 0.41))

set.insert((0.23, 0.82))

set.isEmpty() ==> true

\* 20000 calls with non-degenerate points in a 10-by-10 grid

and probabilities (0.3, 0.05, 0.05, 0.1, 0.2, 0.2)

- failed on trial 4 of 20000

- student isEmpty() = true

- reference isEmpty() = false

- sequence of dequeue operations was:

set.isEmpty() ==> true

set.insert((0.2, 0.2))

set.isEmpty() ==> true

\* 20000 calls with non-degenerate points in a 1-by-1 grid

and probabilities (0.3, 0.05, 0.05, 0.1, 0.2, 0.2)

- failed on trial 4 of 20000

- student size() = 0

- reference size() = 2

- sequence of dequeue operations was:

set.insert((1.0, 1.0))

set.insert((0.0, 0.0))

set.size() ==> 0

==> FAILED

Test 7b: Intermixed sequence of calls to insert(), isEmpty(), size(),

contains(), range(), and nearest() with probabilities

(p1, p2, p3, p4, p5, p6), respectively

\* 20000 calls with distinct points in a 100000-by-100000 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 6 of 20000

- student isEmpty() = true

- reference isEmpty() = false

- sequence of dequeue operations was:

set.insert((0.98926, 0.95272))

set.range([0.31751, 0.40714] x [0.81967, 0.94156])

set.contains((0.40325, 0.54841)) ==> false

set.insert((0.16203, 0.42516))

set.insert((0.82385, 0.84345))

set.isEmpty() ==> true

\* 20000 calls with distinct points in a 10000-by-10000 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 2 of 20000

- student size() = 0

- reference size() = 1

- sequence of dequeue operations was:

set.insert((0.7502, 0.0142))

set.size() ==> 0

\* 20000 calls with distinct points in a 1000-by-1000 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 2 of 20000

- student isEmpty() = true

- reference isEmpty() = false

- sequence of dequeue operations was:

set.insert((0.841, 0.884))

set.isEmpty() ==> true

\* 20000 calls with distinct points in a 100-by-100 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 8 of 20000

- student size() = 0

- reference size() = 2

- sequence of dequeue operations was:

set.insert((0.01, 0.94))

set.contains((0.92, 0.76)) ==> false

set.range([0.53, 0.63] x [0.64, 0.98])

set.insert((0.08, 0.1))

set.contains((0.22, 0.35)) ==> false

set.range([0.37, 0.37] x [0.2, 0.31])

set.nearest((0.33, 0.44)) ==> (0.08, 0.1)

set.size() ==> 0

\* 20000 calls with distinct points in a 10-by-10 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 8 of 20000

- student size() = 0

- reference size() = 3

- sequence of dequeue operations was:

set.insert((0.8, 0.5))

set.insert((0.5, 0.7))

set.nearest((0.0, 0.0)) ==> (0.5, 0.7)

set.insert((0.6, 0.6))

set.nearest((0.5, 0.5)) ==> (0.6, 0.6)

set.nearest((0.2, 0.7)) ==> (0.5, 0.7)

set.nearest((0.2, 0.5)) ==> (0.5, 0.7)

set.size() ==> 0

\* 20000 calls with distinct points in a 1-by-1 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 7 of 20000

- student isEmpty() = true

- reference isEmpty() = false

- sequence of dequeue operations was:

set.size() ==> 0

set.insert((0.0, 1.0))

set.nearest((1.0, 1.0)) ==> (0.0, 1.0)

set.contains((1.0, 0.0)) ==> false

set.range([0.0, 1.0] x [0.0, 0.0])

set.nearest((0.0, 0.0)) ==> (0.0, 1.0)

set.isEmpty() ==> true

==> FAILED

Test 7c: Intermixed sequence of calls to insert(), isEmpty(), size(),

contains(), range(), and nearest() with probabilities

(p1, p2, p3, p4, p5, p6), respectively

\* 20000 calls with general points in a 100000-by-100000 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 3 of 20000

- student size() = 0

- reference size() = 1

- sequence of dequeue operations was:

set.insert((0.53187, 0.95454))

set.nearest((0.25784, 0.70035)) ==> (0.53187, 0.95454)

set.size() ==> 0

\* 20000 calls with general points in a 10000-by-10000 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 6 of 20000

- student size() = 0

- reference size() = 2

- sequence of dequeue operations was:

set.insert((0.3674, 0.4108))

set.insert((0.6711, 0.1485))

set.contains((0.9294, 0.975)) ==> false

set.range([0.0764, 0.4276] x [0.2387, 0.9861])

set.range([0.0773, 0.6749] x [0.3796, 0.6335])

set.size() ==> 0

\* 20000 calls with general points in a 1000-by-1000 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 3 of 20000

- student size() = 0

- reference size() = 1

- sequence of dequeue operations was:

set.insert((0.983, 0.268))

set.contains((0.864, 0.85)) ==> false

set.size() ==> 0

\* 20000 calls with general points in a 100-by-100 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 13 of 20000

- student isEmpty() = true

- reference isEmpty() = false

\* 20000 calls with general points in a 10-by-10 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 16 of 20000

- student isEmpty() = true

- reference isEmpty() = false

\* 20000 calls with general points in a 1-by-1 grid

and probabilities (0.3, 0.05, 0.05, 0.2, 0.2, 0.2)

- failed on trial 7 of 20000

- student contains() = true

- reference contains() = false

- sequence of dequeue operations was:

set.insert((0.0, 1.0))

set.nearest((0.0, 1.0)) ==> (0.0, 1.0)

set.nearest((1.0, 0.0)) ==> (0.0, 1.0)

set.range([0.0, 1.0] x [1.0, 1.0])

set.insert((1.0, 0.0))

set.contains((0.0, 1.0)) ==> true

set.contains((0.0, 0.0)) ==> true

==> FAILED

Test 8: Intermixed sequence of calls to insert(), isEmpty(), size(),

contains(), range(), and nearest() with probabilities

(p1=0, p2, p3, p4, p5, p6), respectively

(data structure with 0 points)

\* 1000 calls with no points in a 1000-by-1000 grid

and probabilities (0.0, 0.5, 0.5, 0.0, 0.0, 0.0)

\* 1000 calls with no points in a 1000-by-1000 grid

and probabilities (0.0, 0.2, 0.2, 0.6, 0.0, 0.0)

\* 1000 calls with no points in a 1000-by-1000 grid

and probabilities (0.0, 0.2, 0.2, 0.0, 0.6, 0.0)

\* 1000 calls with no points in a 1000-by-1000 grid

and probabilities (0.0, 0.2, 0.2, 0.0, 0.0, 0.6)

\* 1000 calls with no points in a 1000-by-1000 grid

and probabilities (0.0, 0.2, 0.2, 0.2, 0.2, 0.2)

==> passed

Total: 4/19 tests passed!

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

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

\* MEMORY

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

Computing memory of Point2D

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

Memory of Point2D object = 32 bytes

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

Computing memory of RectHV

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

Memory of RectHV object = 48 bytes

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

Computing memory of KdTree

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

Running 8 total tests.

Memory usage of a KdTree with n points (including Point2D and RectHV objects).

Maximum allowed memory is 312n + 192 bytes.

n student (bytes) reference (bytes)

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

=> passed 1 168 160

=> passed 2 304 288

=> passed 5 712 672

=> passed 10 1392 1312

=> passed 25 3432 3232

=> passed 100 13632 12832

=> passed 400 54432 51232

=> passed 800 108832 102432

==> 8/8 tests passed

Total: 8/8 tests passed!

Estimated student memory (bytes) = 136.00 n + 32.00 (R^2 = 1.000)

Estimated reference memory (bytes) = 128.00 n + 32.00 (R^2 = 1.000)

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

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

\* TIMING

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

Timing PointSET

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

Running 14 total tests.

Inserting n points into a PointSET

n ops per second

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

=> passed 160000 1393109

=> passed 320000 1317365

=> passed 640000 1042623

=> passed 1280000 848336

==> 4/4 tests passed

Performing contains() queries after inserting n points into a PointSET

n ops per second

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

=> passed 160000 473874

=> passed 320000 485133

=> passed 640000 422238

=> passed 1280000 366482

==> 4/4 tests passed

Performing range() queries after inserting n points into a PointSET

n ops per second

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

=> passed 10000 4356

=> passed 20000 3363

=> passed 40000 960

==> 3/3 tests passed

Performing nearest() queries after inserting n points into a PointSET

n ops per second

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

=> passed 10000 4792

=> passed 20000 2252

=> passed 40000 997

==> 3/3 tests passed

Total: 14/14 tests passed!

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

Timing KdTree

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

Running 28 total tests.

Inserting n points into a 2d tree. The table gives the average number of calls to methods

in RectHV and Point per call to insert().

Point2D

n ops per second RectHV() x() y() equals()

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

=> FAILED 160000 291217 22.6 (11.3x) 55.4 52.9 0.0

=> FAILED 320000 542730 23.0 (11.5x) 56.3 53.8 0.0

=> FAILED 640000 390483 24.5 (12.3x) 60.1 57.6 0.0

=> FAILED 1280000 351262 26.6 (13.3x) 65.4 62.9 0.0

==> 0/4 tests passed

Performing contains() queries after inserting n points into a 2d tree. The table gives

the average number of calls to methods in RectHV and Point per call to contain().

Point2D

n ops per second x() y() equals()

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

=> passed 10000 519849 18.5 17.5 0.0

=> passed 20000 533895 19.7 18.7 0.0

=> passed 40000 506162 21.8 20.8 0.0

=> passed 80000 518002 22.0 21.0 0.0

=> passed 160000 420854 23.2 22.2 0.0

=> passed 320000 431355 25.0 24.0 0.0

=> passed 640000 401335 25.7 24.7 0.0

=> passed 1280000 341000 27.2 26.2 0.0

==> 8/8 tests passed

Performing range() queries after inserting n points into a 2d tree. The table gives

the average number of calls to methods in RectHV and Point per call to range().

n ops per second intersects() contains() x() y()

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

=> passed 10000 319926 50.4 31.1 50.1 12.1

=> passed 20000 320347 52.7 32.6 53.3 16.2

=> passed 40000 282414 64.9 39.3 63.1 14.1

=> passed 80000 258975 67.1 40.7 65.2 14.9

=> passed 160000 248958 70.0 42.5 70.9 20.4

=> passed 320000 243152 67.0 40.2 65.2 15.7

=> passed 640000 207725 72.0 43.3 70.7 19.2

Total: 0/28 tests passed: Could not complete tests, which results in a reported score of 0.

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