Programming Assignment 1: Percolation | percolation.zip

Help

Submission	
Submission time	Sat-28-Jun 18:48:17
Raw Score	58.64 / 100.00
Feedback	See the Assessment Guide for information on how to read this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     No potential bugs found.
API:          PASSED

Correctness:  8/22 tests passed
Memory:       8/8 tests passed
Timing:       9/9 tests passed

Raw score: 58.64% [Correctness: 65%, Memory: 10%, Timing: 25%, Sty
le: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------
total 12K
-rw-r--r-- 1 3.3K Jun 28 15:41 Percolation.java
-rw-r--r-- 1 2.2K Jun 28 15:41 PercolationStats.java
-rw-r--r-- 1 1.9K Jun 28 15:41 studentSubmission.zip


*********************************************************************
************
*  compiling
*********************************************************************
************



% javac Percolation.java
*-----------------------------------------------------------
=============================================================
```

```
% javac PercolationStats.java
*-----------------------------------------------------------
============================================================



% checkstyle *.java
*-----------------------------------------------------------
============================================================


% findbugs *.class
*-----------------------------------------------------------
============================================================


Testing the APIs of your programs.
*-----------------------------------------------------------
Percolation:

PercolationStats:

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


******************************************************************
************
*  executing
******************************************************************
************

Testing methods in Percolation
*-----------------------------------------------------------
Running 14 total tests.

Tests 1 through 7 create a Percolation object using your code, the
n repeatedly
open sites using open(i, j). After each call to open, we check tha
t isFull(),
isOpen(), and percolates() return the corrrect results.


Test 1: Open predetermined list of sites using files
  *  filename = input6.txt
     isFull(1, 6) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt
```

```
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8-no.txt
     isFull(1, 6) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input10-no.txt
     isFull(1, 4) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
  *  filename = greeting57.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = heart25.txt
     isFull(1, 7) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
==> FAILED

Test 2: Open random sites until system percolates (then test is te
rminated)
  *  N = 3
     isFull(1, 2) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  N = 5
     isFull(1, 2) returns wrong value [after 7 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 10

     isFull(1, 4) returns wrong value [after 3 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 10
     isFull(1, 2) returns wrong value [after 17 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 3) returns wrong value [after 4 total calls to open
```

```
()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 1) returns wrong value [after 2 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 15) returns wrong value [after 21 total calls to op
en()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 30) returns wrong value [after 35 total calls to op
en()]
     - student   = false
     - reference = true
==> FAILED

Test 3: Opens predetermined sites for N = 1 and N = 2 (corner case
 test)
  *  filename = input1.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input1-no.txt
  *  filename = input2.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input2-no.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(

)]
     - student   = false
     - reference = true
==> FAILED

Test 4: Check for backwash with predetermined sites
  *  filename = input20.txt
     isFull(1, 1) returns wrong value [after 6 total calls to open
()]
     - student   = false
     - reference = true
  *  filename = input10.txt
     isFull(1, 4) returns wrong value [after 5 total calls to open
()]
     - student   = false
```

```
     - reference = true
  *  filename = input50.txt
     isFull(1, 27) returns wrong value [after 6 total calls to ope
n()]
     - student   = false
     - reference = true
==> FAILED

Test 5: Check for backwash with predetermined sites that have
        multiple percolating paths
  *  filename = input3.txt
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input4.txt
     isFull(1, 1) returns wrong value [after 4 total calls to open
()]
     - student   = false
     - reference = true
  *  filename = input7.txt
     isFull(1, 1) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
==> FAILED

Test 6: Predetermined sites with very long percolating path
  *  filename = snake13.txt
     isFull(1, 1) returns wrong value [after 85 total calls to ope
n()]

     - student   = false
     - reference = true
  *  filename = snake101.txt
     isFull(1, 1) returns wrong value [after 5101 total calls to o
pen()]
     - student   = false
     - reference = true
==> FAILED

Test 7: Opens every site
  *  filename = input5.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
==> FAILED

Test 8: Check whether exception is called if (i, j) are out of bou
```

```
nds
  *  N = 10, (i, j) = (0, 6)
  *  N = 10, (i, j) = (12, 6)
  *  N = 10, (i, j) = (11, 6)
  *  N = 10, (i, j) = (6, 0)
  *  N = 10, (i, j) = (6, 12)
  *  N = 10, (i, j) = (6, 11)
==> passed

Test 9: Check that IllegalArgumentException is thrown if N <= 0 in
 constructor
  *  N = -10
     - IllegalArgumentException NOT thrown
  *  N = -1
     - IllegalArgumentException NOT thrown
  *  N = 0
     - IllegalArgumentException NOT thrown
==> FAILED

Test 10: Create multiple Percolation objects at the same time
         (to make sure you didn't store data in static variables)
     isFull(1, 9) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
     isFull(1, 16) returns wrong value [after 4 total calls to ope
n()]
     - student   = false

     - reference = true
     isFull(1, 5) returns wrong value [after 6 total calls to open
()]
     - student   = false
     - reference = true
==> FAILED

Test 11: Open predetermined list of sites using file
         but change the order in which methods are called
  *  filename = input8.txt;  order =     isFull(),     isOpen(), p
ercolates()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order =     isFull(), percolates(),
   isOpen()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order =     isOpen(),     isFull(), p
```

```
ercolates()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order =     isOpen(), percolates(),
   isFull()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order = percolates(),     isOpen(),
   isFull()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order = percolates(),     isFull(),
   isOpen()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
==> FAILED


Test 12: Call all methods in random order until just before system
 percolates
  *  N = 3
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  N = 5
     percolates() returns wrong value [after 7 total calls to open
()]
     - student   = true
     - reference = false
  *  N = 7
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  N = 10
     isFull(1, 2) returns wrong value [after 2 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 16) returns wrong value [after 34 total calls to op
en()]
```

```
      - student   = false
      - reference = true
  *   N = 50
      isFull(1, 48) returns wrong value [after 72 total calls to op
en()]
      - student   = false
      - reference = true
==> FAILED

Test 13: Call all methods in random order with inputs not prone to
 backwash
  *   N = 3
      isFull(1, 3) returns wrong value [after 3 total calls to open
()]
      - student   = false
      - reference = true
  *   N = 5
      isFull(1, 5) returns wrong value [after 5 total calls to open
()]
      - student   = false

      - reference = true
  *   N = 7
      isFull(1, 2) returns wrong value [after 13 total calls to ope
n()]
      - student   = false
      - reference = true
  *   N = 10
      isFull(1, 3) returns wrong value [after 4 total calls to open
()]
      - student   = false
      - reference = true
  *   N = 20
      isFull(1, 5) returns wrong value [after 24 total calls to ope
n()]
      - student   = false
      - reference = true
  *   N = 50
      isFull(1, 3) returns wrong value [after 69 total calls to ope
n()]
      - student   = false
      - reference = true
==> FAILED

Test 14: Call all methods in random order until all sites are open
  *   N = 3
      isFull(1, 3) returns wrong value [after 1 total call to open(
)]
      - student   = false
      - reference = true
```

```
  *  N = 5
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  N = 7
     isFull(1, 3) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 10
     isFull(1, 4) returns wrong value [after 16 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 4) returns wrong value [after 12 total calls to ope

n()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 37) returns wrong value [after 117 total calls to o
pen()]
     - student   = false
     - reference = true
==> FAILED


Total: 1/14 tests passed!

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

*****************************************************************
************
*  executing PercolationStats with reference Percolation
*****************************************************************
************

Testing methods in PercolationStats
*-----------------------------------------------------------
Running 8 total tests.

Test 1: Test that PercolateStats creates T Percolation objects, ea
ch of size N-by-N
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed
```

```
Test 2: Test that percolates() is called exactly T times, once per
 experiment
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed

Test 3: Test that mean() is consistent with the number of intercep
ted calls to open()
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50

  *  N =  64, T = 150
==> passed

Test 4: Test that stddev() is consistent with the number of interc
epted calls to open()
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed

Test 5: Test that confidenceLo() and confidenceHigh() are consiste
nt with mean() and stddev()
  *  N =  20, T =  10
     -  PercolationStats confidence high = 0.5589496111133156
     -  PercolationStats mean            = 0.5932499999999999
     -  PercolationStats stddev          = 0.05534048648543348
     -  T                                = 10
     -  mean + 1.96 * stddev / sqrt(T)   = 0.6275503888866842
  *  N =  50, T =  20
     -  PercolationStats confidence high = 0.587520190651893
     -  PercolationStats mean            = 0.5993999999999999
     -  PercolationStats stddev          = 0.027106184961331145
     -  T                                = 20
     -  mean + 1.96 * stddev / sqrt(T)   = 0.6112798093481069
  *  N = 100, T =  50
     -  PercolationStats confidence high = 0.5904520705008804
     -  PercolationStats mean            = 0.5944099999999999
     -  PercolationStats stddev          = 0.01427897340962063
     -  T                                = 50
     -  mean + 1.96 * stddev / sqrt(T)   = 0.5983679294991193
  *  N =  64, T = 150
     -  PercolationStats confidence high = 0.589879791895359
     -  PercolationStats mean            = 0.5935123697916667
     -  PercolationStats stddev          = 0.022698883410373964
     -  T                                = 150
     -  mean + 1.96 * stddev / sqrt(T)   = 0.5971449476879743
```

==> **FAILED**

```
Test 6: Check whether exception is thrown if either N or T is out
of bounds
  *  N = -23, T =  42
  *  N =  23, T =   0
  *  N = -42, T =   0
  *  N =  42, T =  -1
==> passed


Test 7: Create two PercolationStats objects at the same time and c
heck mean()
        (to make sure you didn't store data in static variables)
  *  N1 =  50, T1 =  10, N2 =  50, T2 =   5
  *  N1 =  50, T1 =   5, N2 =  50, T2 =  10
  *  N1 =  50, T1 =  10, N2 =  25, T2 =  10
  *  N1 =  25, T1 =  10, N2 =  50, T2 =  10
  *  N1 =  50, T1 =  10, N2 =  15, T2 = 100
  *  N1 =  15, T1 = 100, N2 =  50, T2 =  10
==> passed

Test 8: Check that the methods return the same value, regardless o
f the order in which they are called
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed


Total: 7/8 tests passed!

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


*****************************************************************
************
*  memory usage
*****************************************************************
************

Computing memory of Percolation
*-----------------------------------------------------------
Running 4 total tests.

Test 1a-1d: Measuring total memory usage as a function of grid siz
e (max allowed: 17 N^2 + 128 N + 1024 bytes)

                N        bytes
-----------------------------------------
```

```
=> passed       64        67848
=> passed      256      1057032
=> passed      512      4210952
=> passed     1024     16810248
==> 4/4 tests passed


Estimated student memory = 16.00 N^2 + 32.00 N + 264.00  (R^2 = 1.
000)

Total: 4/4 tests passed!

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



Computing memory of PercolationStats
*-------------------------------------------------------------
Running 4 total tests.

Test 1a-1d: Measuring total memory usage as a function of T (max a
llowed: 8 T + 128 bytes)

                 T        bytes
--------------------------------------------
=> passed       16          184
=> passed       32          312
=> passed       64          568
=> passed      128         1080
==> 4/4 tests passed


Estimated student memory = 8.00 T + 56.00  (R^2 = 1.000)

Total: 4/4 tests passed!

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



*****************************************************************
************
*  executing PercolationStats with reference Percolation
*****************************************************************
************

Timing Percolation
*-------------------------------------------------------------
Running 9 total tests.
```

```
Tests 1a-1e: Measuring runtime and counting calls to connected(),
union() and

             find() in WeightedQuickUnionUF.


For each N, a percolation object is generated and sites are random
ly opened
until the system percolates. If you do not pass the correctness te
sts, these
results may be meaningless.

                                                    2 * connected()
                 N   seconds       union()              + find()
      constructor
-----------------------------------------------------------------
----------------------------
=> passed        8     0.01           58                    64
               1
=> passed       32     0.00          935                   956
               1
=> passed      128     0.03        11028                 13422
               1
=> passed      512     0.06       174585                213870
               1
=> passed     1024     0.34       692632                851554
               1
==> 5/5 tests passed

Running time in seconds depends on the machine on which the script
 runs,
and  varies each time that you submit. If one of the values in the
 table
violates the performance limits, the factor by which you failed th
e test
appears in parentheses. For example, (9.6x) in the union() column
indicates that it uses 9.6x too many calls.


Tests 2a-2d: This test checks whether you use a constant number of
 calls to
union(), connected(), and find() per call to open(), isFull(), and
 percolates().
The table below shows max(union(), connected(), find()) calls made
 during a
single call to open(), isFull(), and percolates().

                 N     per open()      per isOpen()    per isFull(
)    per percolates()

-----------------------------------------------------------------
```

```
---------------------------
=> passed       32        8               0               0
        1
=> passed      128        8               0               0
        1
=> passed      512       10               0               0
        1
=> passed     1024        9               0               0
        1
==> 4/4 tests passed

Total: 9/9 tests passed!
================================================================
```

Submission	
Submission time	Sat-28-Jun 18:48:15
Raw Score	58.64 / 100.00
Feedback	See the Assessment Guide for information on how to read this report.

Assessment Summary

```
Compilation:  PASSED
Style:        PASSED
Findbugs:     No potential bugs found.
API:          PASSED

Correctness:  8/22 tests passed
Memory:       8/8 tests passed
Timing:       9/9 tests passed

Raw score: 58.64% [Correctness: 65%, Memory: 10%, Timing: 25%, Sty
le: 0%]
```

Assessment Details

```
The following files were submitted:
----------------------------------

total 12K
-rw-r--r-- 1 3.3K Jun 28 15:41 Percolation.java
-rw-r--r-- 1 2.2K Jun 28 15:41 PercolationStats.java
-rw-r--r-- 1 1.9K Jun 28 15:41 studentSubmission.zip
```

```
*****************************************************************
************
*  compiling
*****************************************************************
************


% javac Percolation.java
*-----------------------------------------------------------
================================================================

% javac PercolationStats.java
*-----------------------------------------------------------
================================================================



% checkstyle *.java
*-----------------------------------------------------------
================================================================


% findbugs *.class
*-----------------------------------------------------------
================================================================


Testing the APIs of your programs.
*-----------------------------------------------------------
Percolation:

PercolationStats:

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


*****************************************************************
************
*  executing
*****************************************************************

************


Testing methods in Percolation
*-----------------------------------------------------------
Running 14 total tests.

Tests 1 through 7 create a Percolation object using your code, the
n repeatedly
```

```
open sites using open(i, j). After each call to open, we check tha
t isFull(),
isOpen(), and percolates() return the corrrect results.

Test 1: Open predetermined list of sites using files
  *  filename = input6.txt
     isFull(1, 6) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8-no.txt
     isFull(1, 6) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input10-no.txt
     isFull(1, 4) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
  *  filename = greeting57.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = heart25.txt
     isFull(1, 7) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
==> FAILED

Test 2: Open random sites until system percolates (then test is te

rminated)
  *  N = 3
     isFull(1, 3) returns wrong value [after 2 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 5
     isFull(1, 1) returns wrong value [after 2 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 10
```

```
     isFull(1, 4) returns wrong value [after 3 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 10
     isFull(1, 9) returns wrong value [after 27 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 10) returns wrong value [after 32 total calls to op
en()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 11) returns wrong value [after 23 total calls to op
en()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 10) returns wrong value [after 52 total calls to op
en()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 3) returns wrong value [after 8 total calls to open
()]
     - student   = false
     - reference = true
==> FAILED

Test 3: Opens predetermined sites for N = 1 and N = 2 (corner case
 test)
  *  filename = input1.txt

     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input1-no.txt
  *  filename = input2.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input2-no.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
==> FAILED
```

```
Test 4: Check for backwash with predetermined sites
  *  filename = input20.txt
     isFull(1, 1) returns wrong value [after 6 total calls to open
()]
     - student   = false
     - reference = true
  *  filename = input10.txt
     isFull(1, 4) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
  *  filename = input50.txt
     isFull(1, 27) returns wrong value [after 6 total calls to ope
n()]
     - student   = false
     - reference = true
==> FAILED

Test 5: Check for backwash with predetermined sites that have
        multiple percolating paths
  *  filename = input3.txt
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input4.txt
     isFull(1, 1) returns wrong value [after 4 total calls to open
()]
     - student   = false

     - reference = true
  *  filename = input7.txt
     isFull(1, 1) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
==> FAILED

Test 6: Predetermined sites with very long percolating path
  *  filename = snake13.txt
     isFull(1, 1) returns wrong value [after 85 total calls to ope
n()]
     - student   = false
     - reference = true
  *  filename = snake101.txt
     isFull(1, 1) returns wrong value [after 5101 total calls to o
pen()]
     - student   = false
     - reference = true
```

```
==> FAILED

Test 7: Opens every site
  *  filename = input5.txt
     isFull(1, 1) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
==> FAILED

Test 8: Check whether exception is called if (i, j) are out of bou
nds
  *  N = 10, (i, j) = (0, 6)
  *  N = 10, (i, j) = (12, 6)
  *  N = 10, (i, j) = (11, 6)
  *  N = 10, (i, j) = (6, 0)
  *  N = 10, (i, j) = (6, 12)
  *  N = 10, (i, j) = (6, 11)
==> passed

Test 9: Check that IllegalArgumentException is thrown if N <= 0 in
 constructor
  *  N = -10
     - IllegalArgumentException NOT thrown
  *  N = -1
     - IllegalArgumentException NOT thrown
  *  N = 0

     - IllegalArgumentException NOT thrown
==> FAILED

Test 10: Create multiple Percolation objects at the same time
         (to make sure you didn't store data in static variables)
     isFull(1, 1) returns wrong value [after 2 total calls to open
()]
     - student   = false
     - reference = true
     isFull(1, 8) returns wrong value [after 3 total calls to open
()]
     - student   = false
     - reference = true
     isFull(1, 7) returns wrong value [after 6 total calls to open
()]
     - student   = false
     - reference = true
==> FAILED

Test 11: Open predetermined list of sites using file
         but change the order in which methods are called
  *  filename = input8.txt;  order =     isFull(),     isOpen(), p
ercolates()
```

```
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order =     isFull(), percolates(),
   isOpen()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order =     isOpen(),     isFull(), p
ercolates()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order =     isOpen(), percolates(),
   isFull()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order = percolates(),     isOpen(),
   isFull()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  filename = input8.txt;  order = percolates(),     isFull(),
   isOpen()
     isFull(1, 3) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
==> FAILED

Test 12: Call all methods in random order until just before system
 percolates
  *  N = 3
     isFull(1, 1) returns wrong value [after 4 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 5
     isFull(1, 5) returns wrong value [after 3 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 7
     isFull(1, 1) returns wrong value [after 4 total calls to open
```

```
()]
     - student   = false
     - reference = true
  *  N = 10
     isFull(1, 10) returns wrong value [after 5 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 7) returns wrong value [after 7 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 25) returns wrong value [after 6 total calls to ope
n()]
     - student   = false
     - reference = true
==> FAILED


Test 13: Call all methods in random order with inputs not prone to
 backwash
  *  N = 3
     percolates() returns wrong value [after 3 total calls to open
()]
     - student   = true
     - reference = false
  *  N = 5
     isFull(1, 2) returns wrong value [after 1 total call to open(
)]
     - student   = false
     - reference = true
  *  N = 7
     isFull(1, 2) returns wrong value [after 4 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 10
     isFull(1, 3) returns wrong value [after 15 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 20) returns wrong value [after 66 total calls to op
en()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 32) returns wrong value [after 15 total calls to op
```

```
en()]
     - student   = false
     - reference = true
==> FAILED

Test 14: Call all methods in random order until all sites are open
  *  N = 3
     isFull(1, 3) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true
  *  N = 5
     isFull(1, 4) returns wrong value [after 5 total calls to open
()]
     - student   = false
     - reference = true

  *  N = 7
     isFull(1, 2) returns wrong value [after 10 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 10
     isFull(1, 6) returns wrong value [after 11 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 20
     isFull(1, 16) returns wrong value [after 8 total calls to ope
n()]
     - student   = false
     - reference = true
  *  N = 50
     isFull(1, 27) returns wrong value [after 10 total calls to op
en()]
     - student   = false
     - reference = true
==> FAILED


Total: 1/14 tests passed!

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


*****************************************************************
************
*  executing PercolationStats with reference Percolation
*****************************************************************
************

Testing methods in PercolationStats
```

```
*-----------------------------------------------------------
Running 8 total tests.

Test 1: Test that PercolateStats creates T Percolation objects, ea
ch of size N-by-N
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed

Test 2: Test that percolates() is called exactly T times, once per

 experiment
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed

Test 3: Test that mean() is consistent with the number of intercep
ted calls to open()
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed

Test 4: Test that stddev() is consistent with the number of interc
epted calls to open()
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed

Test 5: Test that confidenceLo() and confidenceHigh() are consiste
nt with mean() and stddev()
  *  N =  20, T =  10
     -  PercolationStats confidence high = 0.5565776912634465
     -  PercolationStats mean            = 0.6014999999999999
     -  PercolationStats stddev          = 0.07247796600040289
     -  T                                = 10
     -  mean + 1.96 * stddev / sqrt(T)   = 0.6464223087365534
  *  N =  50, T =  20
     -  PercolationStats confidence high = 0.5841035955622262
     -  PercolationStats mean            = 0.59412
     -  PercolationStats stddev          = 0.022854450217339343
     -  T                                = 20
     -  mean + 1.96 * stddev / sqrt(T)   = 0.6041364044377737
  *  N = 100, T =  50
```

```
     -  PercolationStats confidence high = 0.588867767985409
     -  PercolationStats mean           = 0.5938220000000001
     -  PercolationStats stddev         = 0.017873321699433172
     -  T                               = 50
     -  mean + 1.96 * stddev / sqrt(T)  = 0.5987762320145912
  *  N =  64, T = 150
     -  PercolationStats confidence high = 0.585732376977724
     -  PercolationStats mean           = 0.5893961588541666

     -  PercolationStats stddev         = 0.022893867668727783
     -  T                               = 150
     -  mean + 1.96 * stddev / sqrt(T)  = 0.5930599407306093
==> FAILED

Test 6: Check whether exception is thrown if either N or T is out
of bounds
  *  N = -23, T =  42
  *  N =  23, T =   0
  *  N = -42, T =   0
  *  N =  42, T =  -1
==> passed

Test 7: Create two PercolationStats objects at the same time and c
heck mean()
       (to make sure you didn't store data in static variables)
  *  N1 =  50, T1 =  10, N2 =  50, T2 =   5
  *  N1 =  50, T1 =   5, N2 =  50, T2 =  10
  *  N1 =  50, T1 =  10, N2 =  25, T2 =  10
  *  N1 =  25, T1 =  10, N2 =  50, T2 =  10
  *  N1 =  50, T1 =  10, N2 =  15, T2 = 100
  *  N1 =  15, T1 = 100, N2 =  50, T2 =  10
==> passed

Test 8: Check that the methods return the same value, regardless o
f the order in which they are called
  *  N =  20, T =  10
  *  N =  50, T =  20
  *  N = 100, T =  50
  *  N =  64, T = 150
==> passed


Total: 7/8 tests passed!

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

*****************************************************************
************
*  memory usage
*****************************************************************
************
```

```
************

Computing memory of Percolation
*-----------------------------------------------------------
Running 4 total tests.


Test 1a-1d: Measuring total memory usage as a function of grid siz
e (max allowed: 17 N^2 + 128 N + 1024 bytes)

                N        bytes
-----------------------------------------------
=> passed       64        67848
=> passed      256      1057032
=> passed      512      4210952
=> passed     1024     16810248
==> 4/4 tests passed


Estimated student memory = 16.00 N^2 + 32.00 N + 264.00  (R^2 = 1.
000)

Total: 4/4 tests passed!

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




Computing memory of PercolationStats
*-----------------------------------------------------------
Running 4 total tests.

Test 1a-1d: Measuring total memory usage as a function of T (max a
llowed: 8 T + 128 bytes)

                T        bytes
-----------------------------------------------
=> passed       16          184
=> passed       32          312
=> passed       64          568
=> passed      128         1080
==> 4/4 tests passed


Estimated student memory = 8.00 T + 56.00  (R^2 = 1.000)

Total: 4/4 tests passed!

================================================================
```

```
*****************************************************************
************
*  executing PercolationStats with reference Percolation
*****************************************************************
************

Timing Percolation
*-----------------------------------------------------------
Running 9 total tests.

Tests 1a-1e: Measuring runtime and counting calls to connected(),
union() and
             find() in WeightedQuickUnionUF.


For each N, a percolation object is generated and sites are random
ly opened
until the system percolates. If you do not pass the correctness te
sts, these
results may be meaningless.

                                                  2 * connected()
                 N   seconds       union()              + find()
      constructor
-----------------------------------------------------------------
---------------------------
=> passed        8     0.00           58                    64
               1
=> passed       32     0.00          935                   956
               1
=> passed      128     0.05        11028                 13422
               1
=> passed      512     0.08       174585                213870
               1
=> passed     1024     0.38       692632                851554
               1
==> 5/5 tests passed

Running time in seconds depends on the machine on which the script
 runs,
and  varies each time that you submit. If one of the values in the
 table
violates the performance limits, the factor by which you failed th
e test
appears in parentheses. For example, (9.6x) in the union() column
indicates that it uses 9.6x too many calls.
```

```
Tests 2a-2d: This test checks whether you use a constant number of
 calls to
union(), connected(), and find() per call to open(), isFull(), and
 percolates().
The table below shows max(union(), connected(), find()) calls made
 during a
single call to open(), isFull(), and percolates().

                 N     per open()      per isOpen()    per isFull(
)    per percolates()
-------------------------------------------------------------------
---------------------------
=> passed       32        8               0               0
        1
=> passed      128        8               0               0
        1
=> passed      512       10               0               0
        1
=> passed     1024        9               0               0
        1
==> 4/4 tests passed

Total: 9/9 tests passed!
===================================================================
```