Home
Syllabus
Schedule
Booksite
Lectures
Exercises
Programming Assignments
Job Interview Questions
Discussion Forums

Course Wiki
Join a Meetup

Programming Assignment 1: Percolation | percolation.zip

Submission	
Submission time	Wed-29-Aug 08:49:20
Raw Score	76.79 / 100.00
Feedback	See the Assessment Guide for information on how to read this report.

Assessment Summary

```
Compilation: PASSED
Style: PASSED
API: PASSED

Correctness: 9/14 tests passed
Memory: 4/4 tests passed
Timing: 5/5 tests passed

Raw score: 76.79% [Correctness: 65%, Memory: 10%, Timing: 25%, Style: 0%]
```

Assessment Details

```
files submitted
--------------------------------
total 12K
-rw-r--r-- 1 1.7K Aug 29 15:49 Percolation.java
-rw-r--r-- 1 1.7K Aug 29 15:49 PercolationStats.java
-rw-r--r-- 1 1.5K Aug 29 15:49 studentSubmission.zip


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


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

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



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


% findbugs *.class
*------------------------------------------------------------
/Applications/findbugs-2.0.1/bin/findbugs: Command not found.
================================================================
```

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

PercolationStats:

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


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

Testing methods in Percolation
*-----------------------------------------------------------
Running 9 total tests.

Test 1: Check whether exception is called if (i, j) are out of bounds
  *  N = 10, (i, j) = (-1, 5)
  *  N = 10, (i, j) = (0, 5)
  *  N = 10, (i, j) = (11, 5)
  *  N = 10, (i, j) = (5, -1)
  *  N = 10, (i, j) = (5, 0)
  *  N = 10, (i, j) = (5, 11)
==> passed

Tests 2 through 6 create a Percolation object using your code, then repeatedly open
sites
using open(i, j). After each call to open, we check that isOpen(), percolates()
and isFull() return corrrect results.

Test 2: Open predetermined list of sites using files
  *  filename = input6.txt
  *  filename = input8.txt
  *  filename = input8-no.txt
  *  filename = input10-no.txt
  *  filename = greeting57.txt
  *  filename = heart25.txt
==> passed

Test 3: Open random sites until system percolates (then test is terminated)
  *  N = 5
  *  N = 5
  *  N = 10
  *  N = 10
  *  N = 20
  *  N = 20
  *  N = 50
  *  N = 50
==> passed

Test 4: Opens predetermined sites, but where N = 1 and N = 2 (corner case test)
  *  filename = input1.txt
   percolates() returns wrong value [after 0 total calls to open()]
     - student   = true
     - reference = false
  *  filename = input1-no.txt
   percolates() returns wrong value [after 0 total calls to open()]
     - student   = true
     - reference = false
  *  filename = input2.txt
  *  filename = input2-no.txt
==> FAILED

Test 5a: Predetermined sites which are prone to backwash
  *  filename = input20.txt
   isFull(18, 1) returns wrong value [after 231 total calls to open()]
     - student   = true
```

```
     - reference = false
  *  filename = input10.txt
   isFull(9, 1) returns wrong value [after 56 total calls to open()]
     - student   = true
     - reference = false
==> FAILED

Test 5b: Predetermined sites with multiple percolating paths
  *  filename = input3.txt
   isFull(3, 1) returns wrong value [after 4 total calls to open()]
     - student   = true
     - reference = false
  *  filename = input4.txt
   isFull(4, 4) returns wrong value [after 7 total calls to open()]
     - student   = true
     - reference = false
  *  filename = input50.txt
   isFull(22, 28) returns wrong value [after 1412 total calls to open()]
     - student   = true
     - reference = false
  *  filename = input7.txt
   isFull(6, 1) returns wrong value [after 12 total calls to open()]
     - student   = true
     - reference = false
==> FAILED

Test 6: Opens every site
  *  filename = input5.txt
==> passed

Test 7: Create multiple Percolation objects at the same time (to make sure you
didn't store data in static variables)
==> passed

Test 8: Call all methods in random order
  *  N = 10
  isFull(5, 2) returns wrong value
     - student   = true
     - reference = false
  *  N = 20
  isFull(17, 20) returns wrong value
     - student   = true
     - reference = false
  *  N = 50
  isFull(38, 36) returns wrong value
     - student   = true
     - reference = false
==> FAILED


Total: 5/9 tests passed!
================================================================

Testing methods in PercolationStats
*-----------------------------------------------------------
Running 5 total tests.

Test 1: Test mean and standard deviation of percolation threshold

% java PercolationStats 100 50
-----------------------------------------------

PercolationStats reports:
        mean():    0.594 (PASSED)
        stddev():  0.015 (PASSED)

% java PercolationStats 200 10
-----------------------------------------------
```

```
PercolationStats reports:
        mean():    0.588 (PASSED)
        stddev():  0.008 (PASSED)


Test 2: Check whether exception is called if N, T are out of bounds

  *  N = -23, T = 42
  *  N = 23, T = 0
     - IllegalArgumentException NOT thrown for PercolationStats()
  *  N = -42, T = 0
     - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED

Total: 4/5 tests passed!

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


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

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

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

                 N        bytes
-------------------------------------------
=> passed       64        37040
=> passed      256       590000
=> passed      512      2359472
=> passed     1024      9437360
==> 4/4 tests passed


Estimated student memory = 9.00 N^2 + -0.00 N + 176.00 bytes  (R^2 = 1.000)

Total: 4/4 tests passed!

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



********************************************************************************
*  timing
********************************************************************************

Timing Percolation
*-----------------------------------------------------------
Running 5 total tests.

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

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


                 N   seconds     # union()              # find()
---------------------------------------------------------------------------
=> passed        8     0.00           67                   164
=> passed       32     0.01          781                  1866
=> passed      128     0.03        11399                 28834
=> passed      512     0.12       185772                474148
=> passed     1024     0.29       729779               1864838
```

```
==> 5/5 tests passed

Running time in seconds depends on the machine on which the script runs, and may
vary each time that you submit. If one of the values in the table violates the
performance limits, the factor by which you failed the test appears in parentheses,
e.g. (9.6x) in the union() column would indicate your test uses 9.6x too many calls.

Total: 5/5 tests passed!
================================================================
```