Programming Assignment 1: Percolation | percolation.zip

Submission		
Submission time	Sat-26-Oct 03:16:	12
Raw Score	75.50 / 100.00	
Feedback		ment Summary
		PASSED PASSED No potential bugs found. PASSED
	Memory: Timing:	14/20 tests passed 4/8 tests passed 9/9 tests passed .50% [Correctness: 65%, Memory: 10%, Timing: 25%, St

Assessment Details

_				
9	ınn	าiร	\sim	n
Ju	IDII	113	310	

% javac PercolationStats.java *	% javac F *	ercolation.java
*	======	
% checkstyle *.java *	% javac F	ercolationStats.java
* findbugs *.class *		
*	% checkst	vle *.iava
% findbugs *.class *		
*	======	
Testing the APIs of your programs. * Percolation: PercolationStats: ************** * executing ************** Testing methods in Percolation *	% findbuç	s *.class
Testing the APIs of your programs. * Percolation: PercolationStats: ************** * executing ************** Testing methods in Percolation *	*	
* Percolation: PercolationStats:	=======	=======================================
PercolationStats: ***********************************		

******** * executing ***********************************	*	
******** * executing ***********************************	* Percolati	on:
* executing ***********************************	* Percolati	on:
************* Testing methods in Percolation *	* Percolati Percolati	on: onStats:
********** Testing methods in Percolation *Running 13 total tests.	*Percolati Percolati ========	on: onStats:
*Running 13 total tests.	*Percolati Percolati ======= ******** * execut	on: onStats:
Running 13 total tests.	* Percolati Percolati ====== ******** * execut *******	on: onStats:
Test 1: Check whether excention is called if (i i) are out of	*Percolati Percolati ====== ******** * execut ******* Testing m	onstats: ***********************************
	* Percolati Percolati ====== ******* * execut ******* Testing m *	on: onStats:

```
* 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
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the 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 t
erminated)
  * N = 3
  * 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 (cor
ner case test)
 * filename = input1.txt
 * filename = input1-no.txt
  * filename = input2.txt
  * filename = input2-no.txt
==> passed
```

```
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
     isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
     - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
     isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
  * filename = snake101.txt
==> passed
Test 8: Opens every site
```

```
* filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
       (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
 * filename = input8.txt; order =
                                        isFull(), percolates(),
   isOpen()
  * filename = input8.txt; order =
                                        isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order = isOpen(), percolates(),
   isFull()
  * filename = input8.txt; order = percolates(),
                                                     isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
    isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
  * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
  * N = 5
     isFull(5, 1) returns wrong value [after 17 total calls to op
en()]
     - student = true
     - reference = false
  * N = 7
  * N = 10
```

```
isFull(10, 1) returns wrong value [after 69 total calls to o
pen()]
    - student = true
    - reference = false
  * N = 20
    isFull(20, 1) returns wrong value [after 365 total calls to
open()]
    - student = true
    - reference = false
  * N = 50
    isFull(50, 2) returns wrong value [after 1555 total calls to
open()]
    - student = true
    - reference = false
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
    isFull(3, 1) returns wrong value [after 7 total calls to ope
n()]
     - student = true
    - reference = false
  * N = 5
  * N = 7
    isFull(7, 1) returns wrong value [after 32 total calls to op
en()]
    - student = true
    - reference = false
  * N = 10
    isFull(8, 2) returns wrong value [after 61 total calls to op
en()]
     - student = true
    - reference = false
  * N = 20
  * N = 50
    isFull(41, 1) returns wrong value [after 1543 total calls to
open()]
    - student = true
    - reference = false
==> FAILED
```

```
Total: 9/13 tests passed!
______
Testing methods in PercolationStats
*_____
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold.
Creating new PercolationStats(100, 50)
-----
PercolationStats reports:
      mean(): 0.602 (passed)
      stddev(): 0.019 (passed)
      Overall result: passed
Creating new PercolationStats(200, 10)
PercolationStats reports:
      mean(): 0.669 (FAILED, outside of range)
      stddev(): 0.071 (FAILED, outside of range)
      Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
PercolationStats reports:
      confidenceLo(): 0.587 (passed)
      confidenceHi(): 0.607 (passed)
==> passed
Creating new PercolationStats(200, 10)
-----
 * confidenceLo() = 0.5857135008255461
 * confidenceHi() = 0.6900908395876016
==> FAILED
```

```
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
 * N = 23, T = 0
* N = -42, T = 0
==> passed
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
==> passed
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * order = stddev(), mean()
==> passed
Total: 5/7 tests passed!
______
******************
*****
* memory usage
*****************
Computing memory of Percolation
*_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
           N
                  bytes
-----
=> passed 64
                 41864
=> passed
         256
                 609032
=> passed 512 2397448
=> passed 1024 9513224
==> 4/4 tests passed
```

Submission Estimated student memory = $9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (R}^2 = 1.)$ 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 allowed: 8 T + 128bytes) T bytes => FAILED 16 97400 (380.5x) => **FAILED** 32 => **FAILED** 64 97464 (253.8x) 97592 (152.5x) => FAILED 97848 (84.9x) 128 ==> 0/4 tests passed Estimated student memory = $4.00 \text{ T} + 97336.00 \text{ (R}^2 = 1.000)$ Total: 0/4 tests passed! ***** timing 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 rando mly opened

until the system percolates. If you do not pass the correctness \boldsymbol{t} ests, these

results may be meaningless.

const	N tructor	seconds	union()	2 * connected() + find()
=> passed	8	0.00	86	250
	1			
=> passed	32	0.00	828	3092
	1			
=> passed	128	0.03	11554	48006
	1			
=> passed	512	0.10	186371	785726
	1			
=> passed	1024	0.29	730968	3100964
	1			

==> 5/5 tests passed

Running time in seconds depends on the machine on which the scrip t 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 \boldsymbol{t} he 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 o f calls to

union(), connected(), and find() per call to open(), isFull(), an
d percolates().

The table below shows max(union(), connected(), find()) calls mad

Submission					
	e during a single call	to open(),	, isFull(), and	percolates().	
	() per pe			per isOpen()	per isFull
	=> passed	32	4	0	1
	=> passed	128	4	0	1
	=> passed	512	4	0	1
	=> passed	1024	4	0	1
	==> 4/4 test				
	Total: 9/9 to	ests passe =======	ed! ========	:=========	=======

Submission		
Submission time	Sat-26-Oct 03:15:	37
Raw Score	75.50 / 100.00	
Feedback	See the Assessme	ent Guide for information on how to read this report.
		ment Summary
	Compilation:	PASSED
		PASSED No notantial bugs found
	-	No potential bugs found.
	API:	PASSED
	'	14/20 tests passed
	Correctness:	

Raw score: 75.50% [Correctness: 65%, Memory: 10%, Timing: 25%, St yle: 0%]

Assessment Details

The following files were submitted:
total 12K
-rw-rr 1 2.0K Oct 26 10:15 Percolation.java
-rw-rr 1 2.6K Oct 26 10:15 PercolationStats.java
-rw-rr 1 1.7K Oct 26 10:15 studentSubmission.zip

* compiling

% javac Percolation.java
*
<pre>% javac PercolationStats.java *</pre>
=======================================
% checkstyle *.java
% findbugs *.class
*
====================================
Testing the APIs of your programs.

	lation:
Perco:	lationStats:
====:	
****	*******************
****	* * * * * * *
	ecuting

****	*****
Tooti	ng methods in Percolation
	ng 13 total tests.
	3
Test :	1: Check whether exception is called if (i, j) are out of bo
unds	
* 1	N = 10, (i, j) = (0, 6)
* 1	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)
==> pa	assed
Tests	2 through 8 create a Percolation object using your code, th
	peatedly
-	sites using open(i, j). After each call to open, we check th
	Full(),
isOpe	n(), and percolates() return the corrrect results.
	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
	TITOTIANO = HOUT CESTONE
==> pa	assed

```
Test 3: Open random sites until system percolates (then test is t
erminated)
  * N = 3
  * 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 (cor
ner case test)
 * filename = input1.txt
  * filename = input1-no.txt
  * filename = input2.txt
  * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
     isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
    - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
 * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
```

```
- student = true
     - reference = false
  * filename = input4.txt
    isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
 * filename = snake13.txt
 * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
       (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                       isFull(),
                                                     isOpen(),
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
   isOpen()
  * filename = input8.txt; order = isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order = isOpen(), percolates(),
   isFull()
  * filename = input8.txt; order = percolates(),
                                                     isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
    isOpen()
==> passed
```

```
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
 * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
 * N = 5
  * N = 7
    isFull(7, 1) returns wrong value [after 31 total calls to op
en()]
     - student = true
     - reference = false
  * N = 10
    isFull(10, 1) returns wrong value [after 82 total calls to o
pen()]
     - student = true
    - reference = false
  * N = 20
     isFull(20, 1) returns wrong value [after 244 total calls to
open()]
     - student = true
     - reference = false
  * N = 50
    isFull(50, 1) returns wrong value [after 1916 total calls to
 open()]
     - student = true
     - reference = false
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
  * N = 5
    isFull(5, 5) returns wrong value [after 14 total calls to op
en()]
     - student = true
```

```
- reference = false
 * N = 7
    isFull(7, 7) returns wrong value [after 20 total calls to op
en()]
    - student = true
    - reference = false
 * N = 10
    isFull(10, 1) returns wrong value [after 79 total calls to o
pen()]
    - student = true
    - reference = false
 * N = 20
    isFull(13, 13) returns wrong value [after 261 total calls to
open()]
    - student = true
    - reference = false
 * N = 50
    isFull(41, 2) returns wrong value [after 1515 total calls to
open()]
    - student = true
    - reference = false
==> FAILED
Total: 9/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
______
PercolationStats reports:
       mean(): 0.602 (passed)
       stddev(): 0.019 (passed)
      Overall result: passed
```

```
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 0.656 (passed)
       stddev(): 0.070 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
PercolationStats reports:
       confidenceLo(): 0.588 (passed)
       confidenceHi(): 0.609 (passed)
==> passed
Creating new PercolationStats(200, 10)
-----
  * confidenceLo() = 0.5867731242576526
 * confidenceHi() = 0.6928921653193472
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
 * N = 23, T = 0
 * N = -42, T = 0
==> passed
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
==> passed
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
  * order = stddev(), mean()
==> passed
Total: 5/7 tests passed!
```

```
_____
memory usage
******************
Computing memory of Percolation
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
             N
                   bytes
-----
=> passed 64
                   41864
=> passed 256 609032
=> passed 512 2397448
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (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
allowed: 8 T + 128 bytes)
            T
                   bytes
=> FAILED 16
                  97400 (380.5x)
```

```
=> FAILED
          32
                  97464 (253.8x)
=> FAILED
          64
                  97592 (152.5x)
                  97848 (84.9x)
=> FAILED
         128
==> 0/4 tests passed
Estimated student memory = 4.00 \text{ T} + 97336.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
*****************
  timing
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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                   2 * connected()
              seconds union()
                                        + find()
     constructor
-----
          8
                          86
=> passed
               0.00
                                          250
          1
=> passed 32
               0.00
                        828
                                         3092
```

	1	0.00	44554	40000
=> passed	128	0.02	11554	48006
=> passed	1 512	0.11	186371	785726
-> passeu	1	0.11	100371	763720
=> passed		0.30	730968	3100964
	1			
==> 5/5 test	s passed			
Running time	e in secon	ıds depends	s on the machine o	on which the scr
t runs,				
and varies	each time	that you	submit. If one of	f the values in
e table				
violates the	e performa	nce limits	s, the factor by w	which you failed
he test				
appears in p	arenthese	s. For exa	ample, $(9.6x)$ in $(9.6x)$	the union() colur
indicates th	nat it use	s 9.6x too	many calls.	
		and find(() per call to ope	en(), isFull(), a
d percolates The table be e during a	s(). elow shows	max(unior	n(), connected(),	find()) calls ma
d percolates The table be e during a	s(). elow shows	max(unior		find()) calls ma
d percolates The table be e during a	s(). elow shows to open()	max(unior	n(), connected(),	find()) calls ma
d percolates The table be e during a single call	s(). elow shows to open()	max(union , isFull() per oper	n(), connected(),	find()) calls ma
d percolates The table be e during a single call	s(). elow shows to open() N	max(union , isFull() per oper	n(), connected(),	find()) calls ma
d percolates The table be e during a single call () per pe	o(). elow shows to open() N ercolates(max(unior , isFull() per oper)	n(), connected(), , and percolates n() per isOpe	find()) calls ma
d percolates The table be e during a single call () per pe	o(). elow shows to open() N ercolates(max(union , isFull() per oper	n(), connected(),	find()) calls ma
d percolates The table be e during a single call () per pe	to open() Nercolates(max(unior , isFull() per oper) 4	n(), connected(), n, and percolates n() per isOpe	find()) calls ma (). en() per isFui
d percolates The table be e during a single call () per pe	to open() Nercolates(max(unior , isFull() per oper)	n(), connected(), , and percolates n() per isOpe	find()) calls ma
d percolates The table be e during a single call () per pe	to open() Nercolates(max(unior , isFull() per oper) 4	n(), connected(), n, and percolates n() per isOpe	find()) calls ma (). en() per isFui
d percolates The table be e during a single call () per pe => passed 1 => passed	to open() Nercolates(max(union , isFull() per oper) 4	n(), connected(), n, and percolates n() per isOpe	find()) calls ma (). en() per isFul 1
d percolates The table be e during a single call () per pe => passed 1 => passed 1 => passed	to open() Nercolates(max(union , isFull() per oper) 4	n(), connected(), n, and percolates n() per isOpe	find()) calls ma (). en() per isFul 1
d percolates The table be e during a single call () per percolates => passed 1 => passed 1 => passed	to open() Nercolates(per oper) 4 4	o(), connected(), o, and percolateso o() per isOpe	find()) calls ma (). en() per isFul 1 1
d percolates The table be e during a single call () per percolates => passed 1 => passed 1 => passed 1 => passed	(). elow shows to open() N ercolates(32 128 512 1024	per oper) 4 4	o(), connected(), o, and percolateso o() per isOpe	find()) calls ma (). en() per isFul 1 1
d percolates The table be e during a single call () per pe	(). elow shows to open() N ercolates(32 128 512 1024	per oper) 4 4	o(), connected(), o, and percolateso o() per isOpe	find()) calls ma (). en() per isFul 1 1
d percolates The table be e during a single call () per pe => passed 1 => passed 1 => passed 1 => passed	to open() Nercolates(per oper) 4 4 4	o(), connected(), o, and percolateso o() per isOpe	find()) calls ma (). en() per isFul 1 1

Submission	

Submission	
Submission time	Sat-26-Oct 03:13:46
Raw Score	75.50 / 100.00
Feedback	See the Assessment Guide for information on how to read this report.
	Assessment Summary

Compilation: PASSED

Findbugs: No potential bugs found.

API: PASSED

Style: PASSED

Correctness: 14/20 tests passed
Memory: 4/8 tests passed
Timing: 9/9 tests passed

Raw score: 75.50% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

Su		

_	c Percolation.java
	c PercolationStats.java
% chec	kstyle *.java
=====	
	bugs *.class
=====	
Testin	α the ΔPTs of your programs
	g the APIs of your programs.
*	
* Percol	
* Percol	ation:
* Percol Percol	ation: ationStats:
* Percol Percol	ation: ationStats:
* Percol Percol *****	ation: ationStats:
* Percol Percol *****	ation: ationStats:
* Percol Percol ***** * exe	ation: ationStats:
* Percol Percol ***** * exe *****	ation: ationStats:
* Percol Percol ***** * exe *****	ation: ationStats:
* Percol Percol ***** * exe ***** Testin *	ation: ationStats:
* Percol Percol ***** * exe ***** Testin *	ation: ationStats:

```
unds
 * 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
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the 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 t
erminated)
  * N = 3
  * 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 (cor
ner case test)
 * filename = input1.txt
  * filename = input1-no.txt
 * filename = input2.txt
  * filename = input2-no.txt
==> passed
```

```
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
     isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
    - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
 * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
     isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
  * filename = snake101.txt
==> passed
```

```
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
       (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                       isFull(),
                                                     isOpen(),
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
   isOpen()
  * filename = input8.txt; order = isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order = isOpen(), percolates(),
    isFull()
  * filename = input8.txt; order = percolates(),
                                                     isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
    isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
 * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
  * N = 5
    isFull(5, 1) returns wrong value [after 15 total calls to op
en()]
     - student = true
     - reference = false
  * N = 7
```

```
isFull(7, 1) returns wrong value [after 33 total calls to op
en()]
    - student = true
    - reference = false
  * N = 10
 * N = 20
    isFull(20, 2) returns wrong value [after 310 total calls to
open()]
    - student = true
    - reference = false
 * N = 50
    isFull(50, 2) returns wrong value [after 1705 total calls to
open()]
    - student = true
    - reference = false
==> FAILED
Test 13: Call all methods in random order until all sites are ope
n
 * N = 3
 * N = 5
    isFull(5, 5) returns wrong value [after 16 total calls to op
en()]
    - student = true
    - reference = false
 * N = 7
 * N = 10
 * N = 20
    isFull(20, 17) returns wrong value [after 273 total calls to
open()]
    - student = true
    - reference = false
 * N = 50
    isFull(48, 9) returns wrong value [after 1590 total calls to
open()]
    - student = true
    - reference = false
==> FAILED
Total: 9/13 tests passed!
______
```

```
Testing methods in PercolationStats
*_____
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
      mean(): 0.603 (passed)
      stddev(): 0.021 (passed)
      Overall result: passed
Creating new PercolationStats(200, 10)
PercolationStats reports:
      mean(): 0.654 (passed)
      stddev(): 0.070 (FAILED, outside of range)
      Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
PercolationStats reports:
      confidenceLo(): 0.590 (passed)
      confidenceHi(): 0.612 (passed)
==> passed
Creating new PercolationStats(200, 10)
-----
  * confidenceLo() = 0.5891444077640876
 * confidenceHi() = 0.6924945785113203
==> FAILED
Test 2: Check whether exception is called if N, \mathsf{T} are out of boun
ds
```

```
* N = -23, T = 42
      * N = 23, T = 0
     * N = -42, T = 0
==> passed
Test 3: Create multiple PercolationStats objects at the same time
  (to make sure you didn't store data in static variables)
==> passed
Test 4: Call the methods of PercolationStats in either order.
       * order = mean(), stddev()
      * order = stddev(), mean()
==> passed
Total: 5/7 tests passed!
______
* * * * * * * * * * * *
      memory usage
 ******************
 *****
Computing memory of Percolation
 *_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
                                                   N bytes
=> passed 64
                                                                                 41864
=> passed
                                           256
                                                                            609032
=> passed 512 2397448
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \, \text{N}^2 + 74.00 \, \text{N} + 264.00 \, (\text{R}^2 = 1.00 \, \text{N}^2 + 1.00 \, \text{N}^2 +
000)
```

29 of 296 2013年10月28日 21:20

```
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
allowed: 8 T + 128 bytes)
          T
                 bytes
-----
=> FAILED 16
                97400 (380.5x)
=> FAILED 32
=> FAILED 64
                97464 (253.8x)
               97592 (152.5x)
                97848 (84.9x)
=> FAILED
        128
==> 0/4 tests passed
Estimated student memory = 4.00 \text{ T} + 97336.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
  ****************
  timing
        ************
Timing Percolation
*----
Running 9 total tests.
Tests 1a-1e: Measuring runtime and counting calls to connected(),
union() and
        find() in WeightedQuickUnionUF.
```

30 of 296 2013年10月28日 21:20

For each N, a percolation object is generated and sites are rando mly opened

until the system percolates. If you do not pass the correctness \boldsymbol{t} ests, these

results may be meaningless.

	N	seconds	union()	2 * connected() + find()
cons	tructor			
		0.00	9.6	250
=> passed	8 1	0.00	86	250
=> passed	32	0.00	828	3092
-> passeu	1	0.00	020	3092
=> passed	128	0.02	11554	48006
passa	1	0.02		.5555
=> passed	512	0.11	186371	785726
•	1			
=> passed	1024	0.30	730968	3100964
	1			

==> 5/5 tests passed

Running time in seconds depends on the machine on which the scrip t 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 t

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 o f calls to

union(), connected(), and find() per call to open(), isFull(), an
d percolates().

The table below shows $\max(\text{union()}, \text{ connected()}, \text{ find()})$ calls \max e during a

single call to open(), isFull(), and percolates().

Submission					
	() per p			per isOpen()	per isFull
	=> passed	32	4	0	1
	=> passed	128	4	0	1
	=> passed	512	4	0	1
	=> passed	1024	4	0	1
	==> 4/4 tes	ts passed			
	Total: 9/9	tests passo	ed! =======	==========	

Submission						
Submission time	Sat-26-Oct 03:13:30					
Raw Score	75.50 / 100.00					
Feedback	See the Assessment Guide for information on how to read this report. Assessment Summary					
		PASSED PASSED				
		No potential bugs found.				
	Memory:	14/20 tests passed 4/8 tests passed				
		9/9 tests passed 5.50% [Correctness: 65%, Memory: 10%, Timing: 25%, St				

Assessment Details

The following files were submitted:
total 12K
-rw-rr- 1 2.0K Oct 26 10:13 Percolation.java
-rw-rr 1 2.6K Oct 26 10:13 PercolationStats.java
-rw-rr 1 1.7K Oct 26 10:13 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 13 total tests.
Test 1: Check whether exception is called if (i, j) are out of bo
unds
 * 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
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the 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 t
erminated)
 * N = 3
```

```
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 (cor
ner case test)
 * filename = input1.txt
 * filename = input1-no.txt
 * filename = input2.txt
 * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
    - student = true
    - reference = false
  * filename = input50.txt
    isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
    - student = true
    - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
 * filename = input3.txt
    isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
    - student = true
    - reference = false
  * filename = input4.txt
```

```
isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
 * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
       (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
         but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
    isOpen()
  * filename = input8.txt; order = isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order =
                                        isOpen(), percolates(),
   isFull()
  * filename = input8.txt; order = percolates(),
                                                     isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
    isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
```

```
* N = 5
  * N = 7
  * N = 10
  * N = 20
  * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
 * N = 5
 * N = 7
    isFull(7, 1) returns wrong value [after 38 total calls to op
en()]
     - student = true
     - reference = false
  * N = 10
    isFull(10, 1) returns wrong value [after 87 total calls to o
pen()]
     - student = true
    - reference = false
  * N = 20
    isFull(20, 1) returns wrong value [after 326 total calls to
open()]
     - student = true
     - reference = false
  * N = 50
    isFull(50, 1) returns wrong value [after 2189 total calls to
open()]
     - student = true
     - reference = false
==> FAILED
Test 13: Call all methods in random order until all sites are ope
n
  * N = 3
  * N = 5
  * N = 7
    isFull(6, 6) returns wrong value [after 25 total calls to op
en()]
     - student = true
    - reference = false
  * N = 10
```

```
* N = 20
    isFull(19, 16) returns wrong value [after 278 total calls to
open()]
    - student = true
    - reference = false
 * N = 50
    isFull(38, 49) returns wrong value [after 1487 total calls t
o open()]
    - student = true
    - reference = false
==> FAILED
Total: 9/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 0.604 (passed)
       stddev(): 0.020 (passed)
       Overall result: passed
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 0.660 (FAILED, outside of range)
       stddev(): 0.070 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
```

38 of 296 2013年10月28日 21:20

```
Creating new PercolationStats(100, 50)
-----
PercolationStats reports:
     confidenceLo(): 0.588 (passed)
     confidenceHi(): 0.609 (passed)
==> passed
Creating new PercolationStats(200, 10)
* confidenceLo() = 0.5790816683961664
 * confidenceHi() = 0.68402407198536
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
 * N = 23, T = 0
 * N = -42, T = 0
==> passed
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
==> passed
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * order = stddev(), mean()
==> passed
Total: 5/7 tests passed!
______
memory usage
*****
Computing memory of Percolation
*-----
Running 4 total tests.
```

39 of 296 2013年10月28日 21:20

Test 1a-1d: Measuring total memory usage as a function of grid si ze (max allowed: $17 \text{ N}^2 + 128 \text{ N} + 1024 \text{ bytes}$)

	N	bytes	
=> passed	64	41864	
=> passed	256	609032	
=> passed	512	2397448	
=> passed	1024	9513224	
==> 4/4 tests	passed		

Estimated student memory = $9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (R}^2 = 1.000)$

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 allowed: $8\ T + 128\ bytes)$

	ı	bytes
=> FAILED	16	97400 (380.5x)
=> FAILED	32	97464 (253.8x)
=> FAILED	64	97592 (152.5x)
=> FAILED	128	97848 (84.9x)
==> 0/4 tests	passed	

Estimated student memory = $4.00 \text{ T} + 97336.00 \text{ (R}^2 = 1.000)$

Total: 0/4 tests passed!

********* * timing ***********************************	* * * * * * * * * * * * * * * * * * *			*****************
Timing Perco				
Running 9 to	otal tests	S.		
union() and	d find():	in Weighted	JQuickUnionUF.	alls to connected(),
For each N, a percolation object is generated and sites are rando mly opened until the system percolates. If you do not pass the correctness t ests, these results may be meaningless.				
				2 * connected()
	N	seconds	union()	
const	tructor			+ find()
	tructor	0.00	86	+ find()
	8 1 32			
=> passed	8 1 32 1 128	0.00	86	250
=> passed => passed => passed	8 1 32 1	0.00	86 828	250 3092
=> passed => passed => passed => passed	8 1 32 1 128 1 512 1	0.00 0.00 0.02	86 828 11554	250 3092 48006
=> passed	8 1 32 1 128 1 512 1 1024	0.00 0.00 0.02 0.12	86 828 11554 186371	250 3092 48006 785726

Submission	
	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 the 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()

Submission	
Submission time	Sat-26-Oct 03:01:55
Raw Score	72.25 / 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

PASSED API:

Correctness: 13/20 tests passed Memory: 4/8 tests passed Timing: 9/9 tests passed

Raw score: 72.25% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

```
The following files were submitted:
_____
total 12K
-rw-r--r-- 1 2.0K Oct 26 10:02 Percolation.java
-rw-r--r-- 1 2.5K Oct 26 10:02 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 10:02 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 13 total tests.
Test 1: Check whether exception is called if (i, j) are out of bo unds * 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
Tests 2 through 8 create a Percolation object using your code, then repeatedly

```
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the 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 t
erminated)
  * N = 3
  * 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 (cor
ner case test)
 * filename = input1.txt
  * filename = input1-no.txt
 * filename = input2.txt
  * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
```

```
- reference = false
  * filename = input50.txt
     isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
    - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
    isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
 * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
         but change the order in which methods are called
  * filename = input8.txt; order = isFull(),
                                                      isOpen(),
```

```
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
   isOpen()
  * filename = input8.txt; order =
                                       isOpen(),
                                                     isFull(),
percolates()
 * filename = input8.txt; order =
                                       isOpen(), percolates(),
   isFull()
  * filename = input8.txt; order = percolates(),
                                                     isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
   isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
  * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
  * N = 5
  * N = 7
  * N = 10
    isFull(10, 1) returns wrong value [after 54 total calls to o
pen()]
    - student = true
    - reference = false
  * N = 20
    isFull(20, 1) returns wrong value [after 359 total calls to
open()]
    - student = true
     - reference = false
  * N = 50
    isFull(50, 1) returns wrong value [after 2288 total calls to
open()]
    - student = true
    - reference = false
```

```
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
  * N = 5
 * N = 7
 * N = 10
    isFull(8, 10) returns wrong value [after 42 total calls to o
pen()]
    - student = true
    - reference = false
  * N = 20
    isFull(18, 7) returns wrong value [after 210 total calls to
open()]
    - student = true
    - reference = false
  * N = 50
    isFull(39, 47) returns wrong value [after 1506 total calls t
o open()]
    - student = true
    - reference = false
==> FAILED
Total: 9/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 0.608 (passed)
       stddev(): 0.021 (passed)
       Overall result: passed
```

```
Creating new PercolationStats(200, 10)
-----
PercolationStats reports:
       mean(): 0.653 (passed)
       stddev(): 0.070 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
PercolationStats reports:
       confidenceLo(): 0.589 (passed)
       confidenceHi(): 0.612 (passed)
==> passed
Creating new PercolationStats(200, 10)
  * confidenceLo() = 0.5838202
 * confidenceHi() = 0.6894515133184068
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
  * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
==> passed
Test 4: Call the methods of PercolationStats in either order.
  * order = mean(), stddev()
  * order = stddev(), mean()
```

```
==> passed
Total: 4/7 tests passed!
______
*************
  memory usage
         ************
Computing memory of Percolation
*_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
          N
                bytes
-----
=> passed
         64
                41864
=> passed 256 609032
=> passed 512 2397448
=> passed 1024
                9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (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
allowed: 8 T + 128 bytes)
```

```
bytes
=> FAILED
           16
                    97400 (380.5x)
=> FAILED
           32
                    97464 (253.8x)
=> FAILED
           64
                    97592 (152.5x)
=> FAILED 128
                    97848 (84.9x)
==> 0/4 tests passed
Estimated student memory = 4.00 \text{ T} + 97336.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
  timina
        ***************
*****
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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                       2 * connected()
                           union()
                                            + find()
               seconds
     constructor
```

=> passed	8	0.00	86	250	
	1				
=> passed	32	0.00	828	3092	
	1				
=> passed	128	0.02	11554	48006	
	1				
=> passed	512	0.10	186371	785726	
	1				
=> passed	1024	0.30	730968	3100964	
	1				
==> 5/5 tests passed					
Running time	e in seco	nds depend	s on the machine	e on which the scrip	
t runs,					
and varies each time that you submit. If one of the values in th					
e table					
violates the performance limits, the factor by which you failed \boldsymbol{t}					
he test					
appears in parentheses. For example, (9.6x) in the union() column					

Tests 2a-2d: This test checks whether you use a constant number o f calls to

union(), connected(), and find() per call to open(), isFull(), an
d percolates().

The table below shows $\max(\text{union()}, \text{ connected()}, \text{ find()})$ calls mad e during a

single call to open(), isFull(), and percolates().

indicates that it uses 9.6x too many calls.

	N	per open()	per isOpen()	per isFull
() per pe	ercolates()		
=> passed	32	4	0	1
1				
=> passed	128	4	0	1
1				
=> passed	512	4	0	1
1				
=> passed	1024	4	0	1
1				

Submission				
Submission time	Sat-26-Oct 02:59:21			
Raw Score	72.25 / 100.00			
Feedback	See the Assessment Guide for information on how to read this report.			
	Assessment Summary			
	Compilation: PASSED Style: PASSED			

Style: PASSED

Findbugs: No potential bugs found.

API: PASSED

Correctness: 13/20 tests passed

Memory: 4/8 tests passed

Timing: 9/9 tests passed

Raw score: 72.25% [Correctness: 65%, Memory: 10%, Timing: 25%, St yle: 0%]

Assessment Details

```
The following files were submitted:

total 12K

-rw-r--r-- 1 2.0K Oct 26 09:59 Percolation.java

-rw-r--r-- 1 2.5K Oct 26 09:59 PercolationStats.java

-rw-r--r-- 1 1.7K Oct 26 09:59 studentSubmission.zip
```

Submission	

	* compiling

	% javac Percolation.java
	*
	% javac PercolationStats.java
	*
	% checkstyle *.java
	*
	% findbugs *.class
	*
	Testing the ADIs of your programs
	Testing the APIs of your programs.
	Percolation:
	PercolationStats:
	rei colacionstats.
	=====================================

	* executing

	Testing methods in Percolation

```
*_____
Running 13 total tests.
Test 1: Check whether exception is called if (i, j) are out of bo
unds
 * 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
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the 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 t
erminated)
 * N = 3
    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 (cor
ner case test)
 * filename = input1.txt
```

55 of 296

```
* filename = input1-no.txt
 * filename = input2.txt
 * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
     isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
     - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
     isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
```

```
* filename = snake13.txt
 * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                       isFull(),
                                                     isOpen(),
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
    isOpen()
  * filename = input8.txt; order =
                                       isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order =
                                       isOpen(), percolates(),
   isFull()
  * filename = input8.txt; order = percolates(), isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
    isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
  * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
  * N = 5
    isFull(5, 1) returns wrong value [after 20 total calls to op
```

```
en()]
    - student = true
    - reference = false
  * N = 7
  * N = 10
     isFull(10, 3) returns wrong value [after 65 total calls to o
pen()]
     - student = true
     - reference = false
  * N = 20
    isFull(20, 1) returns wrong value [after 335 total calls to
open()]
    - student = true
     - reference = false
  * N = 50
    isFull(50, 1) returns wrong value [after 2275 total calls to
 open()]
     - student = true
     - reference = false
==> FAILED
Test 13: Call all methods in random order until all sites are ope
n
  * N = 3
  * N = 5
    isFull(5, 1) returns wrong value [after 19 total calls to op
en()]
     - student = true
     - reference = false
  * N = 7
  * N = 10
    isFull(9, 8) returns wrong value [after 47 total calls to op
en()]
     - student = true
     - reference = false
  * N = 20
    isFull(8, 20) returns wrong value [after 216 total calls to
open()]
     - student = true
    - reference = false
  * N = 50
     isFull(39, 6) returns wrong value [after 1358 total calls to
 open()]
```

```
- student = true
    - reference = false
==> FAILED
Total: 9/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
-----
PercolationStats reports:
      mean(): 0.607 (passed)
      stddev(): 0.020 (passed)
      Overall result: passed
Creating new PercolationStats(200, 10)
PercolationStats reports:
      mean(): 0.653 (passed)
      stddev(): 0.069 (FAILED, outside of range)
      Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
PercolationStats reports:
      confidenceLo(): 0.590 (passed)
      confidenceHi(): 0.610 (passed)
==> passed
```

```
Creating new PercolationStats(200, 10)
-----
 * confidenceLo() = 0.5845893476937963
 * confidenceHi() = 0.6887509088948529
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
==> passed
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * order = stddev(), mean()
==> passed
Total: 4/7 tests passed!
______
* memory usage
Computing memory of Percolation
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
                    bytes
```

```
-----
=> passed 64 41864
=> passed 256 609032
=> passed 512 2397448
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (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
allowed: 8 T + 128 bytes)
             Т
                    bytes
                   97400 (380.5x)
=> FAILED 16
=> FAILED 32 97464 (253.8x)
=> FAILED 64 97592 (152.5x)
=> FAILED 128
                    97848 (84.9x)
==> 0/4 tests passed
Estimated student memory = 4.00 \text{ T} + 97336.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
```

* +imina						
* timing ************************************						

Timing Perco	Timing Percolation					
Running 9 to	Running 9 total tests.					
	Tests 1a-1e: Measuring runtime and counting calls to connected(), union() and find() in WeightedQuickUnionUF.					
mly opened until the sy ests, these	For each N, a percolation object is generated and sites are rando mly opened until the system percolates. If you do not pass the correctness t					
				2 * connected()		
	N	seconds	union()	+ find()		
const	ructor					
=> passed	8 1	0.00	86	250		
=> passed	32 1	0.00	828	3092		
=> passed	128 1	0.02	11554	48006		
=> passed	512 1	0.12	186371	785726		
=> passed	1024 1	0.26	730968	3100964		
==> 5/5 test	s passed					
Running time t runs,	e in seco	nds depend	s on the machine	on which the scrip		
and varies each time that you submit. If one of the values in the table						
	e performa	ance limit	s, the factor by	which you failed t		
	parenthese	es. For exa	ample, (9.6x) in	the union() column		

icates tha					
indicates that it uses 9.6x too many calls.					
Tests 2a-2d: This test checks whether you use a constant number o f calls to union(), connected(), and find() per call to open(), isFull(), and d percolates(). The table below shows max(union(), connected(), find()) calls mad e during a single call to open(), isFull(), and percolates().					
			0	1	
passed 1	128	4	0	1	
passed 1	512	4	0	1	
passed 1	1024	4	0	1	
4/4 tests	passed				
al: 9/9 te ======	sts passed	d! ========		=======	
	ts 2a-2d: alls to on(), conn ercolates(table bel uring a gle call t per per passed 1 passed 1 passed 1 passed 1 4/4 tests	ts 2a-2d: This test alls to on(), connected(), a ercolates(). table below shows r uring a gle call to open(), N per percolates() passed 32 1 passed 128 1 passed 512 1 passed 1024 1 4/4 tests passed	ts 2a-2d: This test checks whether alls to on(), connected(), and find() per ercolates(). table below shows max(union(), couring a gle call to open(), isFull(), and N per open() per percolates()	ts 2a-2d: This test checks whether you use a const alls to on(), connected(), and find() per call to open(), ercolates(). table below shows max(union(), connected(), find(uring a gle call to open(), isFull(), and percolates(). N per open() per isOpen() per percolates() passed 32 4 0 1 passed 128 4 0 1 passed 512 4 0 1 passed 512 4 0 1 passed 1024 4 0 1 4/4 tests passed	

Submission	
Submission time	Sat-26-Oct 02:49:15
Raw Score	59.25 / 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: 9/20 tests passed
Memory: 4/8 tests passed
Timing: 9/9 tests passed

Raw score: 59.25% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 2.0K Oct 26 09:49 Percolation.java
-rw-r--r-- 1 2.4K Oct 26 09:49 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 09:49 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 *		
Test 1: Check whether exception is called if (i, j) are out of bo unds * 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		
Tests 2 through 8 create a Percolation object using your code, then repeatedly open sites using open(i, j). After each call to open, we check that isFull(), isOpen(), and percolates() return the correct 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 t
erminated)
  * N = 3
  * 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 (cor
ner case test)
 * filename = input1.txt
 * filename = input1-no.txt
 * filename = input2.txt
  * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
    isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
    - student = true
     - reference = false
```

```
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
    isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
 * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
       (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
         but change the order in which methods are called
  * filename = input8.txt; order = isFull(),
                                                     isOpen(),
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
   isOpen()
  * filename = input8.txt; order =
                                        isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order = isOpen(), percolates(),
```

```
isFull()
  * filename = input8.txt; order = percolates(), isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
    isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
  * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
  * N = 5
  * N = 7
    isFull(7, 1) returns wrong value [after 32 total calls to op
en()]
     - student = true
     - reference = false
  * N = 10
    isFull(10, 2) returns wrong value [after 71 total calls to o
pen()]
     - student = true
    - reference = false
  * N = 20
    isFull(20, 1) returns wrong value [after 228 total calls to
open()]
    - student = true
     - reference = false
  * N = 50
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
  * N = 5
```

```
isFull(3, 4) returns wrong value [after 13 total calls to op
en()]
    - student = true
    - reference = false
 * N = 7
    isFull(7, 6) returns wrong value [after 32 total calls to op
en()]
    - student = true
    - reference = false
 * N = 10
    isFull(7, 9) returns wrong value [after 61 total calls to op
en()]
    - student = true
    - reference = false
 * N = 20
    isFull(19, 20) returns wrong value [after 293 total calls to
open()]
    - student = true
    - reference = false
 * N = 50
    isFull(47, 12) returns wrong value [after 1529 total calls t
o open()]
    - student = true
    - reference = false
==> FAILED
Total: 9/13 tests passed!
______
Testing methods in PercolationStats
*-----
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 6066.918 (FAILED, outside of range)
       stddev(): 195.207 (FAILED, outside of range)
```

```
Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 26161.627 (FAILED, outside of range)
       stddev(): 2796.918 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
 * confidenceLo() = 5844.820924506238
 * confidenceHi() = 6076.954737239822
==> FAILED
Creating new PercolationStats(200, 10)
-----
  * confidenceLo() = 22073.472162918228
 * confidenceHi() = 27965.370968061077
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
 (to make sure you didn't store data in static variables)
  * 1mean = 59.848425
 * 2mean = 58.61
  * 1mean = 237.565419
  * 2mean = 232.76
```

```
* 1mean = 59.814062875
      * 2mean = 58.31
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
      * order = mean(), stddev()
      * mean = 531.854204875; stddev = 32.638551940072425
      * order = stddev(), mean()
       * mean = 536.8542042500001; stddev = 33.50322749954518
==> FAILED
Total: 0/7 tests passed!
______
 ******************
     memory usage
 ******************
Computing memory of Percolation
*_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
                                                 N bytes
 -----
=> passed 64 41864
=> passed 256 609032
=> passed
                                          512
                                                                      2397448
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \, \text{N}^2 + 74.00 \, \text{N} + 264.00 \, (\text{R}^2 = 1.00 \, \text{N} + 1.00 \, 
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
allowed: 8 T + 128 bytes)
            Т
                    bytes
-----
=> FAILED 16
                   97392 (380.4x)
=> FAILED 32
=> FAILED 64
=> FAILED 128
                   97456 (253.8x)
                  97584 (152.5x)
                   97840 (84.9x)
==> 0/4 tests passed
Estimated student memory = 4.00 \text{ T} + 97328.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
*****
  timing
*******************
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 rando
```

	stem per	colates. If	you do not pa	ss the correctness t
ests, these	ha maani	mmlaaa		
results may	be illeans	ingress.		
const	N Tructor	seconds	union()	2 * connected() + find()
=> passed	8	0.00	86	250
	1			
=> passed	32	0.00	828	3092
	1			
=> passed	128 1	0.02	11554	48006
=> passed	1 512	0 11	186371	785726
passea	1	0.11	100371	103120
=> passed	_	0.25	730968	3100964
-	1			
==> 5/5 test	s passed	I		
Running time	e in seco	onds depends	on the machin	e on which the scrip
t runs, and varies	each tim	ne that you	submit. If one	of the values in th
e table				
violates the	e perform	nance limits	, the factor b	y which you failed t
				n the union() column
appears in p			mple, (9.6x) i many calls.	n the union() column
appears in p	nat it us	ses 9.6x too	many calls.	• •
appears in pindicates the Tests 2a-2d:	nat it us This te	es 9.6x too	many calls.	a constant number o
appears in pindicates the Tests 2a-2d: f calls to union(), cor	This te	es 9.6x too	many calls.	a constant number o
appears in pindicates the Tests 2a-2d: f calls to union(), cord percolates	This tennected()	est checks w	many calls. Thether you use	a constant number o open(), isFull(), an
appears in pindicates the Tests 2a-2d: f calls to union(), cord percolates The table be	This tennected()	est checks w	many calls. Thether you use	a constant number o open(), isFull(), an
appears in pindicates the Tests 2a-2d: f calls to union(), cord percolates The table beed during a	This tennected() S(). elow show	est checks w , and find(us max(union	many calls. Thether you use	a constant number o open(), isFull(), an), find()) calls mad

Submission					
	=> passed	32	4	0	1
	1				
	=> passed	128	4	0	1
	1 => passed	512	4	0	1
	-> passeu	512	4	U	1
		1024	4	0	1
	1				
	==> 4/4 test	s passed			
	Total: 9/9 t	ests passe	ed!		
	=========	=======		=======================================	

Submission time	Sat-26-Oct 02:47:10				
Raw Score	59.25 / 100.00				
Feedback	See the Assessme	ent Guide for information on how to read this report.			
	Assessment Summary				
	A33C33	inient Summary			
	Compilation:	PASSED			
	Style:	PASSED			
	Findbugs:	No potential bugs found.			
	API:	PASSED			
	Correctness:	9/20 tests passed			
	Memory:	4/8 tests passed			
		9/9 tests passed			
	Timing:	o, o cosco passou			

total 12k	
	- 1 2.0K Oct 26 09:47 Percolation.java
	- 1 2.4K Oct 26 09:47 PercolationStats.java
-rw-rr-	- 1 1.7K Oct 26 09:47 studentSubmission.zip
* * * * * * * * *	**************
******	***
* compil	ing

*****	***
% javac F	ercolation.java
*	
=======	=======================================
% javac F	ercolationStats.java
*	
*	
*	
*	
*	
* ======= % checkst	
* ======== % checkst *	
*	yle *.java
* ======== % checkst *	yle *.java
* ======= % checkst * =========	======================================
* ================================	======================================
* ======== % checkst * ===============================	yle *.java s * *.class
* % checkst * % findbug *	yle *.java s
* *	yle *.java s *.class he APIs of your programs.
* ===============================	yle *.java s *.class he APIs of your programs. on:

```
executing
******************
Testing methods in Percolation
Running 13 total tests.
Test 1: Check whether exception is called if (i, j) are out of bo
unds
 * 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
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the 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 t
erminated)
  * N = 3
  * 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 (cor
ner case test)
  * filename = input1.txt
 * filename = input1-no.txt
 * filename = input2.txt
 * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
    isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
    - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
    isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
    isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
```

```
- reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
 * filename = snake13.txt
 * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
       (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                       isFull(),
                                                     isOpen(),
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
   isOpen()
  * filename = input8.txt; order = isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order = isOpen(), percolates(),
   isFull()
  * filename = input8.txt; order = percolates(), isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
   isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
```

```
* N = 20
 * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
    isFull(20, 2) returns wrong value [after 286 total calls to
open()]
    - student = true
    - reference = false
  * N = 50
    isFull(50, 2) returns wrong value [after 1884 total calls to
open()]
    - student = true
    - reference = false
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
  * N = 5
  * N = 7
  * N = 10
    isFull(10, 7) returns wrong value [after 70 total calls to o
pen()]
    - student = true
    - reference = false
  * N = 20
    isFull(17, 7) returns wrong value [after 233 total calls to
open()]
    - student = true
    - reference = false
  * N = 50
    isFull(35, 7) returns wrong value [after 1500 total calls to
open()]
    - student = true
    - reference = false
==> FAILED
```

```
Total: 9/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
-----
PercolationStats reports:
      mean(): 6095.245 (FAILED, outside of range)
      stddev(): 184.363 (FAILED, outside of range)
      Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
      mean(): 26144.961 (FAILED, outside of range)
      stddev(): 2789.449 (FAILED, outside of range)
      Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
 * confidenceLo() = 5898.468913860101
 * confidenceHi() = 6131.285749001362
==> FAILED
Creating new PercolationStats(200, 10)
 * confidenceLo() = 22104.255033290985
 * confidenceHi() = 28004.246657702053
```

80 of 296 2013年10月28日 21:20

```
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
 * 1mean = 60.656504999999996
   2mean = 57.93
 * 1mean = 239.61592199999998
 * 2mean = 237.53
 * 1mean = 59.76883675
 * 2mean = 59.06
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * mean = 537.08536; stddev = 32.922804747780226
 * order = stddev(), mean()
 * mean = 534.9848577500001; stddev = 36.337070794219606
==> FAILED
Total: 0/7 tests passed!
______
*****
  memory usage
             ************
Computing memory of Percolation
Running 4 total tests.
```

81 of 296 2013年10月28日 21:20

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

	N	bytes	
=> passed	64	41864	
=> passed	256	609032	
=> passed	512	2397448	
=> passed	1024	9513224	
==> 4/4 tests	s passed		

Estimated student memory = $9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (R}^2 = 1.)$ 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 allowed: 8 T + 128bytes)

	Т	bytes
=> FAILED	16	97392 (380.4x)
=> FAILED	32	97456 (253.8x)
=> FAILED	64	97584 (152.5x)
=> FAILED	128	97840 (84.9x)
==> 0/4 tests	s passed	

Estimated student memory = $4.00 \text{ T} + 97328.00 \text{ (R}^2 = 1.000)$

Total: 0/4 tests passed!

* * * * * * * * * * * *		* * * * * * * * * *	* * * * * * * * * * * * * * * *	* * * * * * * * * * * * * * * * * * * *
* timing **************		******	******	*******
Timing Perco	olation			
Running 9 to	otal test	S.		
Tests 1a-1e: union() and	t		and counting ca	lls to connected(),
mly opened	-	_	-	and sites are rando s the correctness
	be meani	ngless.		
ests, these results may const		ngless. seconds	union()	2 * connected() + find()
const	N Tructor	seconds		+ find()
results may	N Tructor		union() 	* *
const	N cructor 8 1 32	seconds		+ find()
const	N Tructor 8 1	seconds 0.00	86	+ find()
const => passed => passed	N cructor 8 1 32 1 128	seconds 0.00 0.00	86 828	+ find() 250 3092
const => passed => passed	N cructor 8 1 32 1 128 1	seconds 0.00 0.00 0.00	86 828 11554	+ find() 250 3092 48006
const => passed => passed => passed => passed	N cructor 8 1 32 1 128 1 512 1 1024	seconds 0.00 0.00 0.02 0.12 0.29	86 828 11554 186371	+ find() 250 3092 48006 785726

Submission				
	e table violates the perfo he test appears in parenth indicates that it	eses. For example	e, (9.6x) in the ur	
	Tests 2a-2d: This f calls to union(), connected d percolates(). The table below she during a single call to ope	(), and find() pe	er call to open(), connected(), find(isFull(), an
	() per percolat	es() 4	per isOpen() 	1
	=> passed 128 1 => passed 512		0	1
	=> passed 1024 1 ==> 4/4 tests pass		0	1
	Total: 9/9 tests p			:=======

Submission	
Submission time	Sat-26-Oct 02:38:58
Raw Score	56.00 / 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: 8/20 tests passed
Memory: 4/8 tests passed
Timing: 9/9 tests passed

Raw score: 56.00% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

% ch	eckstyle *.java
*	
====	=======================================
o/ f ;	ndhuga * alaga
*	ndbugs *.class
====	
Test	ing the APIs of your programs.
* Perc	olation:
Perc	olationStats:

	xecuting

***	*****
Test	ing methods in Percolation
Runn	ing 13 total tests.
Test unds	1: Check whether exception is called if (i, j) are out of bo
*	N = 10, (i, j) = (0, 6)
	- IndexOutOfBoundsException NOT thrown for isOpen()
*	N = 10, (i, j) = (12, 6)
*	N = 10, (i, j) = (11, 6)
_	- IndexOutOfBoundsException NOT thrown for isOpen()
*	N = 10, (i, j) = (6, 0) $IndevOutOfBoundoEvention NOT thrown for icOnon()$
*	<pre>- IndexOutOfBoundsException NOT thrown for isOpen() N = 10, (i, j) = (6, 12)</pre>
*	N = 10, (1, j) = (0, 12) N = 10, (i, j) = (6, 11)
	- IndexOutOfBoundsException NOT thrown for isOpen()

```
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the 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 t
erminated)
  * N = 3
  * 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 (cor
ner case test)
 * filename = input1.txt
  * filename = input1-no.txt
 * filename = input2.txt
  * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * filename = input20.txt
    isFull(18, 1) returns wrong value [after 231 total calls to
open()]
    - student = true
     - reference = false
```

87 of 296 2013年10月28日 21:20

```
* filename = input10.txt
     isFull(9, 1) returns wrong value [after 56 total calls to op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
     isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
     - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
     isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
     isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
  * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
==> passed
```

88 of 296 2013年10月28日 21:20

```
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                       isFull(),
                                                     isOpen(),
percolates()
 * filename = input8.txt; order =
                                       isFull(), percolates(),
   isOpen()
  * filename = input8.txt; order =
                                       isOpen(),
                                                     isFull(),
percolates()
  * filename = input8.txt; order = isOpen(), percolates(),
   isFull()
 * filename = input8.txt; order = percolates(),
                                                     isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
   isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
  * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
 * N = 3
  * N = 5
    isFull(5, 2) returns wrong value [after 17 total calls to op
en()]
     - student = true
    - reference = false
  * N = 7
  * N = 10
    isFull(10, 1) returns wrong value [after 84 total calls to o
pen()]
    - student = true
    - reference = false
  * N = 20
    isFull(20, 1) returns wrong value [after 341 total calls to
```

```
open()]
    - student = true
    - reference = false
  * N = 50
    isFull(50, 2) returns wrong value [after 2221 total calls to
 open()]
    - student = true
    - reference = false
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
  * N = 5
    isFull(5, 5) returns wrong value [after 16 total calls to op
en()]
    - student = true
    - reference = false
  * N = 7
    isFull(5, 2) returns wrong value [after 29 total calls to op
en()]
     - student = true
    - reference = false
  * N = 10
    isFull(5, 1) returns wrong value [after 69 total calls to op
en()]
    - student = true
    - reference = false
  * N = 20
    isFull(17, 13) returns wrong value [after 226 total calls to
 open()]
    - student = true
    - reference = false
  * N = 50
    isFull(40, 4) returns wrong value [after 1453 total calls to
 open()]
    - student = true
     - reference = false
==> FAILED
Total: 8/13 tests passed!
______
```

```
Testing methods in PercolationStats
*_____
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 6064.755 (FAILED, outside of range)
       stddev(): 199.935 (FAILED, outside of range)
       Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 26217.849 (FAILED, outside of range)
       stddev(): 2792.823 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
  * confidenceLo() = 5876.748227225469
  * confidenceHi() = 6109.321681103487
==> FAILED
Creating new PercolationStats(200, 10)
-----
  * confidenceLo() = 22069.74205610429
  * confidenceHi() = 27939.56406647413
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
```

```
* N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
  ^{*} N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
 (to make sure you didn't store data in static variables)
 * 1mean = 59.75751599999995
 * 2mean = 61.58
    * 2mean = 235.79
 * 1mean = 59.140696125
 * 2mean = 58.91
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * mean = 536.115510875; stddev = 33.37725989977465
 * order = stddev(), mean()
 * mean = 532.788878125; stddev = 33.41039198776254
==> FAILED
Total: 0/7 tests passed!
______
* memory usage
Computing memory of Percolation
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
                      bytes
```

```
-----
=> passed 64 41864
=> passed 256 609032
=> passed 512 2397448
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (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
allowed: 8 T + 128 bytes)
             Т
                    bytes
=> FAILED 16
                   97392 (380.4x)
=> FAILED 32
=> FAILED 64
           32
                   97456 (253.8x)
                  97584 (152.5x)
=> FAILED 128
                    97840 (84.9x)
==> 0/4 tests passed
Estimated student memory = 4.00 \text{ T} + 97328.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
```

Timing Perc				
Running 9 to				
Tests 1a-1e union() and		ng runtime	and counting cal	ls to connected()
	find()	in Weighted	QuickUnionUF.	
mly opened		_	-	nd sites are rand
until the sy ests, these results may			you do not pass	the correctness
,		3		2 * connected()
	N	seconds	union()	2 * connected() + find()
cons	tructor 			
=> passed	8 1	0.00	86	250
=> passed	32 1	0.00	828	3092
=> passed	128 1	0.02	11554	48006
	512 1	0.12	186371	785726
=> passed	_	0.07	730968	3100964
=> passed => passed		0.27	10000	
	1		.00000	
=> passed ==> 5/5 tes Running time	1 ts passed		on the machine (on which the scri
=> passed ==> 5/5 tes Running time t runs,	1 ts passed e in seco	nds depends		

Submission					
	indicates th	at it uses	s 9.6x too man	y calls.	
	f calls to union(), con d percolates The table be e during a	nected(), (). low shows	<pre>and find() pe max(union(),</pre>	er you use a const r call to open(), connected(), find d percolates().	isFull(), an
	() per pe)	per isOpen()	
	=> passed	32		0	1
	=> passed	128	4	0	1
	=> passed	512	4	0	1
	=> passed	1024	4	0	1
	==> 4/4 test	s passed			
	Total: 9/9 t	•	ed! =======		

Submission	
Submission time	Sat-26-Oct 02:32:46
Raw Score	56.00 / 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: 8/20 tests passed
Memory: 4/8 tests passed
Timing: 9/9 tests passed

Raw score: 56.00% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 2.0K Oct 26 09:32 Percolation.java
-rw-r--r-- 1 2.4K Oct 26 09:32 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 09:32 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 *
<pre>Test 1: Check whether exception is called if (i, j) are out of bo unds * N = 10, (i, j) = (0, 6)</pre>
Tests 2 through 8 create a Percolation object using your code, then repeatedly open sites using open(i, j). After each call to open, we check that isFull(),

```
isOpen(), and percolates() return the 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 t
erminated)
  * N = 3
  * 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 (cor
ner case test)
 * filename = input1.txt
 * filename = input1-no.txt
 * filename = input2.txt
  * filename = input2-no.txt
==> passed
Test 5: Check for backwash with predetermined sites
  * 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 op
en()]
     - student = true
     - reference = false
  * filename = input50.txt
```

```
isFull(22, 28) returns wrong value [after 1412 total calls t
o open()]
    - student = true
     - reference = false
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     isFull(3, 1) returns wrong value [after 4 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input4.txt
     isFull(4, 4) returns wrong value [after 7 total calls to ope
n()]
     - student = true
     - reference = false
  * filename = input7.txt
    isFull(6, 1) returns wrong value [after 12 total calls to op
en()]
     - student = true
     - reference = false
==> FAILED
Test 7: Predetermined sites with very long percolating path
 * filename = snake13.txt
 * filename = snake101.txt
==> passed
Test 8: Opens every site
 * filename = input5.txt
==> passed
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
==> passed
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order = isFull(),
                                                      isOpen(),
percolates()
  * filename = input8.txt; order = isFull(), percolates(),
```

99 of 296 2013年10月28日 21:20

```
isOpen()
  * filename = input8.txt; order = isOpen(), isFull(),
percolates()
 * filename = input8.txt; order = isOpen(), percolates(),
   isFull()
  * filename = input8.txt; order = percolates(),
                                                     isOpen(),
   isFull()
  * filename = input8.txt; order = percolates(), isFull(),
   isOpen()
==> passed
Test 11: Call all methods in random order until just before syste
m percolates
  * N = 3
  * N = 5
  * N = 7
  * N = 10
  * N = 20
  * N = 50
==> passed
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
  * N = 5
  * N = 7
    isFull(7, 1) returns wrong value [after 28 total calls to op
en()]
    - student = true
     - reference = false
  * N = 10
  * N = 20
    isFull(20, 1) returns wrong value [after 339 total calls to
open()]
    - student = true
     - reference = false
    isFull(50, 1) returns wrong value [after 1667 total calls to
 open()]
     - student = true
     - reference = false
==> FAILED
```

```
Test 13: Call all methods in random order until all sites are ope
n
 * N = 3
 * N = 5
    isFull(5, 4) returns wrong value [after 12 total calls to op
en()]
    - student = true
    - reference = false
 * N = 7
    isFull(6, 1) returns wrong value [after 23 total calls to op
en()]
    - student = true
    - reference = false
 * N = 10
    isFull(10, 3) returns wrong value [after 50 total calls to o
pen()]
    - student = true
    - reference = false
 * N = 20
    isFull(13, 9) returns wrong value [after 246 total calls to
open()]
    - student = true
    - reference = false
 * N = 50
    isFull(41, 13) returns wrong value [after 1519 total calls t
o open()]
    - student = true
    - reference = false
==> FAILED
Total: 8/13 tests passed!
______
Testing methods in PercolationStats
*_____
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
```

```
PercolationStats reports:
       mean(): 6047.122 (FAILED, outside of range)
       stddev(): 202.725 (FAILED, outside of range)
       Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 26356.848 (FAILED, outside of range)
       stddev(): 2836.426 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
  * confidenceLo() = 5869.679304881756
  * confidenceHi() = 6092.424796492741
==> FAILED
Creating new PercolationStats(200, 10)
-----
  * confidenceLo() = 22601.24232979259
  * confidenceHi() = 28616.15430406328
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
  * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
 (to make sure you didn't store data in static variables)
```

```
* 1mean = 59.545395
        * 2mean = 59.48
                 1mean = 239.99976000000004
        * 2mean = 237.13
                 1mean = 58.24119875
        * 2mean = 59.565
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
        * order = mean(), stddev()
       * mean = 537.301440375; stddev = 32.80296125782147
       * order = stddev(), mean()
        * mean = 534.7336015; stddev = 31.747648718685976
==> FAILED
Total: 0/7 tests passed!
 *******************
        memory usage
 ********************
Computing memory of Percolation
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
                                                         N bytes
 -----
                                             64
=> passed
                                                                                         41864
=> passed 256 609032
=> passed 512 2397448
=> passed
                                             1024
                                                                                 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \, \text{N}^2 + 74.00 \, \text{N} + 264.00 \, (\text{R}^2 = 1.00 \, \text{N}^2 + 74.00 \, \text{N} + 264.00 \, \text{N}^2 + 1.00 \, \text{N}^2 
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
allowed: 8 T + 128 bytes)
          Т
                 bytes
-----
=> FAILED
        16
                97392 (380.4x)
=> FAILED 32
              97456 (253.8x)
                97584 (152.5x)
=> FAILED
         64
=> FAILED 128
                97840 (84.9x)
==> 0/4 tests passed
Estimated student memory = 4.00 \text{ T} + 97328.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
 timing
     *****
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 rando mly opened

until the system percolates. If you do not pass the correctness \boldsymbol{t} ests, these

results may be meaningless.

				2 * connected()
	N	seconds	union()	+ find()
const	ructor			
=> passed	8	0.00	86	250
	1			
=> passed	32	0.00	828	3092
	1			
=> passed	128	0.02	11554	48006
	1			
=> passed	512	0.11	186371	785726
	1			
=> passed	1024	0.22	730968	3100964
	1			

==> 5/5 tests passed

Running time in seconds depends on the machine on which the scrip t 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 \boldsymbol{t} he 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 o f calls to

union(), connected(), and find() per call to open(), isFull(), an
d percolates().

The table below shows $\max(\text{union}(), \text{ connected}(), \text{ find}())$ calls mad e during a

single call to open(), isFull(), and percolates().

Submission	ı				
	() per pe			per isOpen()	per isFull
	=> passed	32	4	0	1
	=> passed	128	4	0	1
	=> passed	512	4	0	1
	=> passed	1024	4	0	1
	==> 4/4 test	s passed			
	Total: 9/9 1	ests passe	ed! 		

Submission					
Submission time	Sat-26-Oct 02:21:38				
Raw Score	18.89 / 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: 0/20 tests passed Memory: 4/8 tests passed Timing: 5/9 tests passed Raw score: 18.89% [Correctness: 65%, Memory: 10%, Timing: 25%, St				

yle: 0%]

Assessment Details

The fol	lowing files were submitted:
total 1	2K
-rw-r	r 1 2.0K Oct 26 09:21 Percolation.java
-rw-r	r 1 2.4K Oct 26 09:21 PercolationStats.java
-rw-r	r 1 1.7K Oct 26 09:21 studentSubmission.zip
*****	***************
*****	* * * * *
* comp	iling
*****	**************
*****	* * * * *
% javac	Percolation.java
*	
======	=======================================
% javac	PercolationStats.java
*	
======	=======================================
% check	style *.java
*	
======	
% findh	ugs *.class
*	
=====	=======================================
Testina	the APIs of your programs.
*	
Percola	tion:

PercolationStats:

* executing ************************************
Testing methods in Percolation
Running 13 total tests.
<pre>Test 1: Check whether exception is called if (i, j) are out of bounds * N = 10, (i, j) = (0, 6) - IndexOutOfBoundsException NOT thrown for open() - IndexOutOfBoundsException NOT thrown for isOpen() - IndexOutOfBoundsException NOT thrown for isFull() * N = 10, (i, j) = (12, 6) * N = 10, (i, j) = (11, 6) * N = 10, (i, j) = (6, 0) - IndexOutOfBoundsException NOT thrown for open() - IndexOutOfBoundsException NOT thrown for isOpen() - IndexOutOfBoundsException NOT thrown for isFull() * N = 10, (i, j) = (6, 12) * N = 10, (i, j) = (6, 11) ==> FAILED</pre>
Tests 2 through 8 create a Percolation object using your code, the en repeatedly open sites using open(i, j). After each call to open, we check that isFull(),
<pre>isOpen(), and percolates() return the corrrect results. Test 2: Open predetermined list of sites using files * filename = input6.txt java.lang.ArrayIndexOutOfBoundsException Percolation.isFull(Percolation.java:49) TestPercolation.checkIsFull(TestPercolation.java:22)</pre>

```
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:196)
TestPercolation.main(TestPercolation.java:540)
```

- * filename = input8.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:197)
 TestPercolation.main(TestPercolation.java:540)
- * filename = input8-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:198)
 TestPercolation.main(TestPercolation.java:540)
- * filename = input10-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:199)
 TestPercolation.main(TestPercolation.java:540)
- * filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)
 TestPercolation.main(TestPercolation.java:540)
- * filename = heart25.txt
 java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:201)
    TestPercolation.main(TestPercolation.java:540)
==> FAILED
Test 3: Open random sites until system percolates (then test is t
erminated)
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:235)
    TestPercolation.main(TestPercolation.java:541)
   N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:236)
    TestPercolation.main(TestPercolation.java:541)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:237)
    TestPercolation.main(TestPercolation.java:541)
   N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
```

```
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:238)
TestPercolation.main(TestPercolation.java:541)
```

* N = 20

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:239)
TestPercolation.main(TestPercolation.java:541)

* N = 20
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)

TestPercolation.main(TestPercolation.java:541)

* N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:241)
TestPercolation.main(TestPercolation.java:541)

* N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:242)
TestPercolation.main(TestPercolation.java:541)

==> FAILED

Test 4: Opens predetermined sites, but where N = 1 and N = 2 (cor

111 of 296

```
ner case test)
  * filename = input1.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test4(TestPercolation.java:251)
    TestPercolation.main(TestPercolation.java:542)
  * filename = input1-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:252)
     TestPercolation.main(TestPercolation.java:542)
  * filename = input2.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test4(TestPercolation.java:253)
     TestPercolation.main(TestPercolation.java:542)
   filename = input2-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:254)
     TestPercolation.main(TestPercolation.java:542)
==> FAILED
Test 5: Check for backwash with predetermined sites
  * filename = input20.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
```

```
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test5(TestPercolation.java:263)
TestPercolation.main(TestPercolation.java:543)
```

- * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:265)
 TestPercolation.main(TestPercolation.java:543)

==> FAILED

Test 6: Check for backwash with predetermined sites that havemult iple percolating paths

- * filename = input3.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:275)
 TestPercolation.main(TestPercolation.java:544)
- * filename = input4.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)

```
TestPercolation.test6(TestPercolation.java:276)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input7.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:277)
     TestPercolation.main(TestPercolation.java:544)
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:287)
    TestPercolation.main(TestPercolation.java:545)
  * filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:288)
     TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
```

114 of 296 2013年10月28日 21:20

TestPercolation.test8(TestPercolation.java:296)

```
TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:336)
    TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:337)
    TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
         but change the order in which methods are called
  * filename = input8.txt; order = isFull(),
                                                       isOpen(),
percolates()
     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(),
percolates()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = isOpen(), percolates(),
    isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:179)
    TestPercolation.test10(TestPercolation.java:386)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                       isOpen(),
    isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:180)
    TestPercolation.test10(TestPercolation.java:387)
    TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
 * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
```

```
TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:360)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation. random Calls Until Percolation (TestPercolation.\\
java:363)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
    N = 50
     java.lang.ArrayIndexOutOfBoundsException
```

```
Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
     TestPercolation.test12(TestPercolation.java:458)
     TestPercolation.main(TestPercolation.java:551)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
     TestPercolation.test12(TestPercolation.java:459)
     TestPercolation.main(TestPercolation.java:551)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:460)
     TestPercolation.main(TestPercolation.java:551)
    N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
```

```
TestPercolation.test12(TestPercolation.java:461)
    TestPercolation.main(TestPercolation.java:551)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:462)
    TestPercolation.main(TestPercolation.java:551)
   N = 50
     java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:463)
     TestPercolation.main(TestPercolation.java:551)
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:517)
    TestPercolation.main(TestPercolation.java:552)
   N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:518)
    TestPercolation.main(TestPercolation.java:552)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
```

```
Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:519)
    TestPercolation.main(TestPercolation.java:552)
    N = 10
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:520)
    TestPercolation.main(TestPercolation.java:552)
   N = 20
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:521)
    TestPercolation.main(TestPercolation.java:552)
  * N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
```

120 of 296 2013年10月28日 21:20

hold

Test 1a-1b: Test mean and standard deviation of percolation thres

```
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 6063.408 (FAILED, outside of range)
       stddev(): 198.290 (FAILED, outside of range)
       Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 26144.294 (FAILED, outside of range)
       stddev(): 2770.098 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
  * confidenceLo() = 5860.494716862262
 * confidenceHi() = 6092.318657323769
==> FAILED
Creating new PercolationStats(200, 10)
* confidenceLo() = 22287.866684051463
 * confidenceHi() = 28207.28489808384
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
```

```
Test 3: Create multiple PercolationStats objects at the same time
 (to make sure you didn't store data in static variables)
   1mean = 60.474686999999996
 * 2mean = 58.36
 * 1mean = 238.50481200000002
 * 2mean = 237.59
 * 1mean = 59.457279
 * 2mean = 58.87
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * mean = 534.366767375; stddev = 32.54823511493913
 * order = stddev(), mean()
 * mean = 536.1758123750001; stddev = 31.958938425449247
==> FAILED
Total: 0/7 tests passed!
______
******************
*****
* memory usage
*****************
Computing memory of Percolation
*_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
            N
                  bytes
-----
=> passed 64
                 41864
=> passed
          256
                 609032
=> passed 512 2397448
=> passed 1024 9513224
==> 4/4 tests passed
```

Submission Estimated student memory = $9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (R}^2 = 1.)$ 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 allowed: 8 T + 128bytes) Т bytes => FAILED 16 97392 (380.4x) => **FAILED** 32 => **FAILED** 64 97456 (253.8x) 97584 (152.5x) 97840 (84.9x) => FAILED 128 ==> 0/4 tests passed Estimated student memory = $4.00 \text{ T} + 97328.00 \text{ (R}^2 = 1.000)$ Total: 0/4 tests passed! ***** timing 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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                                  2 * connected()
                                   union()
                     seconds
                                                         + find()
                 N
       constructor
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                 8 Infinity
                                      21
                                                             30
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                      68
                                           (0.5x)
                                                             30
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
                                           (0.1x)
               128 Infinity
                                     263
                                                            230
(0.1x)
```

```
java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
              512 Infinity
                                   1029 (0.0x)
                                                        1554
(0.0x)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
             1024 Infinity 2054 (0.0x)
=> FAILED
                                                        3276
(0.0x)
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
and varies each time that you submit. If one of the values in th
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                      per open()
                                     per isOpen()
                                                     per isFull
     per percolates()
  java.lang.ArrayIndexOutOfBoundsException
```

```
Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                32
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
               128
                          0
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
               512
                          0
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                                                          0
              1024
==> 4/4 tests passed
Total: 5/9 tests passed!
```

Submission	

Submission	
Submission time	Sat-26-Oct 02:06:56
Raw Score	18.89 / 100.00
Feedback	See the Assessment Guide for information on how to read this report.
	Assessment Summary

Compilation: PASSED

Findbugs: No potential bugs found.

API: PASSED

Style: PASSED

Correctness: 0/20 tests passed Memory: 4/8 tests passed Timing: 5/9 tests passed

Raw score: 18.89% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 2.0K Oct 26 09:07 Percolation.java
-rw-r--r-- 1 2.4K Oct 26 09:07 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 09:07 studentSubmission.zip
 compiling
```

_						
Su	ın	m		cı	^	n
Ju	ı		13	31	u	

% iavac	Percolation.java
======	.======================================
0/ ====================================	Daniel Intigential Control
	PercolationStats.java
======	
% checks	tyle *.java
*	
======	.======================================
% findbu	gs *.class
	90 101400
*	
*	
* ======= Testing	the APIs of your programs.
* ======= Testing *	the APIs of your programs.
* ======= Testing *	the APIs of your programs.
* ======= Testing * Percolat	the APIs of your programs.
* ======= Testing * Percolat	the APIs of your programs. ion:
* ======= Testing * Percolat	the APIs of your programs. ion:
* ======= Testing * Percolat	the APIs of your programs. ion:
* Testing * Percolat Percolat ======	the APIs of your programs. ion: ionStats:
* Testing * Percolat Percolat ======	the APIs of your programs. ion: ionStats:
* Testing * Percolat Percolat ====== ********	the APIs of your programs. ion: ionStats:
* Testing * Percolat Percolat ====== ********	the APIs of your programs. ion: ionStats: ***********************************
* Testing * Percolat Percolat ====== ******** * execu	the APIs of your programs. ion: ionStats: ***********************************
* Testing * Percolat Percolat ====== ******** * execu	the APIs of your programs. ion: ionStats: ***********************************

```
unds
  * N = 10, (i, j) = (0, 6)
     - IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (12, 6)
  * N = 10, (i, j) = (11, 6)
  * N = 10, (i, j) = (6, 0)
     IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (6, 12)
  * N = 10, (i, j) = (6, 11)
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the corrrect results.
Test 2: Open predetermined list of sites using files
  * filename = input6.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:196)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:197)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
```

```
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:198)
TestPercolation.main(TestPercolation.java:540)
```

- * filename = input10-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:199)
 TestPercolation.main(TestPercolation.java:540)
- * filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)
 TestPercolation.main(TestPercolation.java:540)
- * filename = heart25.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:201)
 TestPercolation.main(TestPercolation.java:540)

==> FAILED

Test 3: Open random sites until system percolates (then test is t erminated)

* N = 3
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)

TestPercolation.test3(TestPercolation.java:235)
TestPercolation.main(TestPercolation.java:541)

* N = 5

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:236)
TestPercolation.main(TestPercolation.java:541)

* N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:237)
TestPercolation.main(TestPercolation.java:541)

* N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:238)
TestPercolation.main(TestPercolation.java:541)

* N = 20

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:239)
TestPercolation.main(TestPercolation.java:541)

* N = 20
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)

TestPercolation.checkIsFull(TestPercolation.java:22)

131 of 296

```
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)
TestPercolation.main(TestPercolation.java:541)
```

* N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:241)
TestPercolation.main(TestPercolation.java:541)

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:242)
TestPercolation.main(TestPercolation.java:541)

==> FAILED

Test 4: Opens predetermined sites, but where N=1 and N=2 (cor ner case test)

* filename = input1.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:251)
 TestPercolation.main(TestPercolation.java:542)

* filename = input1-no.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test4(TestPercolation.java:252)

TestPercolation.main(TestPercolation.java:542)

- * filename = input2.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:253)
 TestPercolation.main(TestPercolation.java:542)
- * filename = input2-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:254)
 TestPercolation.main(TestPercolation.java:542)

==> FAILED

Test 5: Check for backwash with predetermined sites

- * filename = input20.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:263)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test5(TestPercolation.java:265)
    TestPercolation.main(TestPercolation.java:543)
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test6(TestPercolation.java:275)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input4.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:276)
    TestPercolation.main(TestPercolation.java:544)
  * filename = input7.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:277)
    TestPercolation.main(TestPercolation.java:544)
```

==> FAILED

Test 7: Predetermined sites with very long percolating path

* filename = snake13.txt java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:287)
     TestPercolation.main(TestPercolation.java:545)
  * filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test7(TestPercolation.java:288)
    TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test8(TestPercolation.java:296)
     TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:336)
     TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
```

135 of 296 2013年10月28日 21:20

TestPercolation.checkIsFull(TestPercolation.java:22)

```
TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:337)
    TestPercolation.main(TestPercolation.java:548)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
    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(),
percolates()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = isOpen(), percolates(),
    isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
```

```
TestPercolation.file(TestPercolation.java:179)
     TestPercolation.test10(TestPercolation.java:386)
     TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                        isOpen(),
    isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.file(TestPercolation.java:180)
     TestPercolation.test10(TestPercolation.java:387)
     TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation. random Calls Until Percolation (TestPercolation.\\
java:364)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
   N = 7
     java.lang.ArrayIndexOutOfBoundsException
```

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:364)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation. random Calls Until Percolation (TestPercolation.\\
java:363)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
   N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
```

```
TestPercolation.test12(TestPercolation.java:458)
     TestPercolation.main(TestPercolation.java:551)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:459)
     TestPercolation.main(TestPercolation.java:551)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
    TestPercolation.test12(TestPercolation.java:460)
    TestPercolation.main(TestPercolation.java:551)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:461)
    TestPercolation.main(TestPercolation.java:551)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:462)
    TestPercolation.main(TestPercolation.java:551)
   N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
```

```
37)
    TestPercolation.test12(TestPercolation.java:463)
    TestPercolation.main(TestPercolation.java:551)
==> FAILED
Test 13: Call all methods in random order until all sites are ope
    N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
     TestPercolation.test13(TestPercolation.java:517)
    TestPercolation.main(TestPercolation.java:552)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
     TestPercolation.test13(TestPercolation.java:518)
    TestPercolation.main(TestPercolation.java:552)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
     TestPercolation.test13(TestPercolation.java:519)
    TestPercolation.main(TestPercolation.java:552)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
     TestPercolation.test13(TestPercolation.java:520)
    TestPercolation.main(TestPercolation.java:552)
    N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
```

```
TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:521)
    TestPercolation.main(TestPercolation.java:552)
  * N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
    java.lang.ArrayIndexOutOfBoundsException: 50
    PercolationStats.<init>(PercolationStats.java:24)
    TestPercolationStats.test1(TestPercolationStats.java:17)
    TestPercolationStats.main(TestPercolationStats.java:223)
Creating new PercolationStats(200, 10)
_____
    java.lang.ArrayIndexOutOfBoundsException: 10
    PercolationStats.<init>(PercolationStats.java:24)
    TestPercolationStats.test1(TestPercolationStats.java:17)
    TestPercolationStats.main(TestPercolationStats.java:224)
```

```
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
     java.lang.ArrayIndexOutOfBoundsException: 50
     PercolationStats.<init>(PercolationStats.java:24)
    TestPercolationStats.test1c(TestPercolationStats.java:69)
     TestPercolationStats.main(TestPercolationStats.java:232)
==> FAILED
Creating new PercolationStats(200, 10)
     java.lang.ArrayIndexOutOfBoundsException: 10
    PercolationStats.<init>(PercolationStats.java:24)
    TestPercolationStats.test1c(TestPercolationStats.java:69)
    TestPercolationStats.main(TestPercolationStats.java:233)
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
  * N = -23, T = 42
     - IllegalArgumentException NOT thrown for PercolationStats()
  * N = 23, T = 0
     - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
     - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException: 100
     PercolationStats.<init>(PercolationStats.java:24)
    Test Percolation Stats. two Percolation Stats (Test Percolation Stats) \\
s.java:132)
    TestPercolationStats.test3(TestPercolationStats.java:169)
     TestPercolationStats.main(TestPercolationStats.java:236)
     java.lang.ArrayIndexOutOfBoundsException: 100
     PercolationStats.<init>(PercolationStats.java:24)
     Test Percolation Stats. two Percolation Stats (Test Percolation Stats) \\
s.java:132)
```

```
TestPercolationStats.test3(TestPercolationStats.java:170)
    TestPercolationStats.main(TestPercolationStats.java:236)
    java.lang.ArrayIndexOutOfBoundsException: 200
    PercolationStats.<init>(PercolationStats.java:24)
    {\tt TestPercolationStats.twoPercolationStats} ({\tt TestPercolationStat}
s.java:132)
    TestPercolationStats.test3(TestPercolationStats.java:171)
    TestPercolationStats.main(TestPercolationStats.java:236)
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
    java.lang.ArrayIndexOutOfBoundsException: 200
    PercolationStats.<init>(PercolationStats.java:24)
    TestPercolationStats.test4(TestPercolationStats.java:182)
    TestPercolationStats.main(TestPercolationStats.java:237)
==> FAILED
Total: 0/7 tests passed!
______
memory usage
Computing memory of Percolation
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
              N
                      bytes
=> passed
           64
                     41864
=> passed
                    609032
           256
=> passed
           512 2397448
=> passed
          1024
                   9513224
```

==> 4/4 tests passed
Estimated student memory = $9.00 \text{N}^2 + 74.00 \text{N} + 264.00 \text{(R}^2 = 1.000)$
Total: 4/4 tests passed!
Computing memory of PercolationStats
* Exception in thread "main" java.lang.ArrayIndexOutOfBoundsExcepti on: 16
at PercolationStats. <init>(PercolationStats.java:24) at MemoryOfPercolationStats.main(MemoryOfPercolationStats .java:81)</init>
Running 4 total tests. Test 1a-1d: Measuring total memory usage as a function of T (max allowed: 8 T + 128 bytes)
T bytes
Total: 0/4 tests passed:Test aborted. Ran out of time or crashed before completion.

******* * timing

Timing Percolation
*

```
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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                                  2 * connected()
                                   union()
                     seconds
                                                        + find()
                 N
       constructor
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                 8 Infinity
                                      21
                                                             30
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                      68
                                           (0.5x)
                                                             30
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
                                           (0.1x)
=> FAILED
               128 Infinity
                                     263
                                                            230
(0.1x)
```

```
java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
              512 Infinity
                                   1029 (0.0x)
                                                        1554
(0.0x)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
             1024 Infinity 2054 (0.0x)
=> FAILED
                                                        3276
(0.0x)
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
and varies each time that you submit. If one of the values in th
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                      per open()
                                    per isOpen()
                                                     per isFull
     per percolates()
  java.lang.ArrayIndexOutOfBoundsException
```

```
Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                32
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
               128
                          0
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
               512
                          0
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                                                          0
              1024
==> 4/4 tests passed
Total: 5/9 tests passed!
```

Submission	

Submission	
Submission time	Sat-26-Oct 01:56:10
Raw Score	18.89 / 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.

PASSED API:

Correctness: 0/20 tests passed Memory: 4/8 tests passed Timing: 5/9 tests passed

Raw score: 18.89% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 2.0K Oct 26 08:56 Percolation.java
-rw-r--r-- 1 2.5K Oct 26 08:56 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 08:56 studentSubmission.zip
 compiling
```

_				
9	ınn	าiรร	SIA	n
Ju	IDII.	1133	31 U	41

javac Periodercolation	ercolationStats.java
javac Pe	ercolationStats.java vle *.java s *.class
javac Pe	ercolationStats.java vle *.java s *.class
checksty checksty findbugs esting the crcolation ercolation ercolation excellention	/le *.java
checksty checksty checksty findbugs esting the crcolation ercolation ercolation excellent excell	/le *.java
checksty findbugs esting th ercolation ercolation ercolation ercolation	/le *.java
findbugs esting the colation ercolation ercolation ercolation excellent the colation	s *.class
findbugs esting the colation ercolation ercolation ercolation excellent the colation	s *.class
findbugs esting the colation ercolation ercolation ercolation excellent the colation	s *.class
findbugs esting the ercolation ercolation ercolation ercolation excellent the ercolation excellent the ercolation	s *.class
esting the ercolation	
esting the ercolation	
esting the ercolation	
esting the control of	
ercolation ercolation ercolation ercolation ercolation ercolation excelling	:======================================
ercolation ercolation ercolation ercolation ercolation ercolation excelling	
ercolation ercolation ercolation ercolation ercolation ercolation excelling	ne APIs of your programs.
ercolatio	
******** **********	on:
******** **********	Object -
******** ********* executi	instats:
executi	
executi	
executi	**************
*****	***
	.ng
******	**********
esting mo	***
ınning 13	ethods in Percolation
est 1: Ch	ethods in Percolation

```
unds
  * N = 10, (i, j) = (0, 6)
     - IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (12, 6)
  * N = 10, (i, j) = (11, 6)
  * N = 10, (i, j) = (6, 0)
     IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (6, 12)
  * N = 10, (i, j) = (6, 11)
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the corrrect results.
Test 2: Open predetermined list of sites using files
  * filename = input6.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:196)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:197)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
```

```
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:198)
TestPercolation.main(TestPercolation.java:540)
```

- * filename = input10-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:199)
 TestPercolation.main(TestPercolation.java:540)
- * filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)
 TestPercolation.main(TestPercolation.java:540)
- * filename = heart25.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:201)
 TestPercolation.main(TestPercolation.java:540)

==> FAILED

Test 3: Open random sites until system percolates (then test is t erminated)

* N = 3
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)

2013年10月28日 21:20

Submission

TestPercolation.test3(TestPercolation.java:235)
TestPercolation.main(TestPercolation.java:541)

* N = 5

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:236)
TestPercolation.main(TestPercolation.java:541)

* N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:237)
TestPercolation.main(TestPercolation.java:541)

* N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:238)
TestPercolation.main(TestPercolation.java:541)

* N = 20

152 of 296

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:239)
TestPercolation.main(TestPercolation.java:541)

* N = 20
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)

```
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)
TestPercolation.main(TestPercolation.java:541)
```

* N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:241)
TestPercolation.main(TestPercolation.java:541)

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:242)
TestPercolation.main(TestPercolation.java:541)

==> FAILED

Test 4: Opens predetermined sites, but where N=1 and N=2 (cor ner case test)

* filename = input1.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:251)
 TestPercolation.main(TestPercolation.java:542)

* filename = input1-no.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test4(TestPercolation.java:252)

TestPercolation.main(TestPercolation.java:542)

- * filename = input2.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:253)
 TestPercolation.main(TestPercolation.java:542)
- * filename = input2-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:254)
 TestPercolation.main(TestPercolation.java:542)

==> FAILED

Test 5: Check for backwash with predetermined sites

- * filename = input20.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:263)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test5(TestPercolation.java:265)
    TestPercolation.main(TestPercolation.java:543)
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test6(TestPercolation.java:275)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input4.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:276)
    TestPercolation.main(TestPercolation.java:544)
  * filename = input7.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:277)
    TestPercolation.main(TestPercolation.java:544)
==> FAILED
```

Test 7: Predetermined sites with very long percolating path

* filename = snake13.txt java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:287)
     TestPercolation.main(TestPercolation.java:545)
  * filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test7(TestPercolation.java:288)
    TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test8(TestPercolation.java:296)
     TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:336)
     TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
```

156 of 296 2013年10月28日 21:20

TestPercolation.checkIsFull(TestPercolation.java:22)

```
TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:337)
    TestPercolation.main(TestPercolation.java:548)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
         but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
    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(),
percolates()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = isOpen(), percolates(),
    isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
```

```
TestPercolation.file(TestPercolation.java:179)
     TestPercolation.test10(TestPercolation.java:386)
     TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                        isOpen(),
    isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.file(TestPercolation.java:180)
     TestPercolation.test10(TestPercolation.java:387)
     TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation. random Calls Until Percolation (TestPercolation.\\
java:363)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
   N = 7
     java.lang.ArrayIndexOutOfBoundsException
```

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:364)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation. random Calls Until Percolation (TestPercolation.\\
java:364)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     isFull(1, 2) returns wrong value [after 1 total call to open
()]
     - student = false
     - reference = true
```

```
N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
    TestPercolation.test12(TestPercolation.java:459)
    TestPercolation.main(TestPercolation.java:551)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
     TestPercolation.test12(TestPercolation.java:460)
    TestPercolation.main(TestPercolation.java:551)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:461)
     TestPercolation.main(TestPercolation.java:551)
   N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:462)
    TestPercolation.main(TestPercolation.java:551)
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:463)
     TestPercolation.main(TestPercolation.java:551)
```

```
==> FAILED
Test 13: Call all methods in random order until all sites are ope
    N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:517)
    TestPercolation.main(TestPercolation.java:552)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:518)
    TestPercolation.main(TestPercolation.java:552)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
     TestPercolation.test13(TestPercolation.java:519)
    TestPercolation.main(TestPercolation.java:552)
   N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
    TestPercolation.randomCalls(TestPercolation.java:484)
    TestPercolation.test13(TestPercolation.java:520)
    TestPercolation.main(TestPercolation.java:552)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:521)
     TestPercolation.main(TestPercolation.java:552)
```

```
N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
java.lang.ArrayIndexOutOfBoundsException: 50
    PercolationStats.<init>(PercolationStats.java:20)
    TestPercolationStats.test1(TestPercolationStats.java:17)
    TestPercolationStats.main(TestPercolationStats.java:223)
Creating new PercolationStats(200, 10)
    java.lang.ArrayIndexOutOfBoundsException: 10
    PercolationStats.<init>(PercolationStats.java:20)
    TestPercolationStats.test1(TestPercolationStats.java:17)
    TestPercolationStats.main(TestPercolationStats.java:224)
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
    java.lang.ArrayIndexOutOfBoundsException: 50
```

```
PercolationStats.<init>(PercolationStats.java:20)
    TestPercolationStats.test1c(TestPercolationStats.java:69)
     TestPercolationStats.main(TestPercolationStats.java:232)
==> FAILED
Creating new PercolationStats(200, 10)
     java.lang.ArrayIndexOutOfBoundsException: 10
     PercolationStats.<init>(PercolationStats.java:20)
    TestPercolationStats.test1c(TestPercolationStats.java:69)
     TestPercolationStats.main(TestPercolationStats.java:233)
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
  * N = -23, T = 42
     - IllegalArgumentException NOT thrown for PercolationStats()
  * N = 23, T = 0
     - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
     - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
 (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException: 100
     PercolationStats.<init>(PercolationStats.java:20)
     {\tt TestPercolationStats.twoPercolationStats} ({\tt TestPercolationStat}
s.java:132)
     TestPercolationStats.test3(TestPercolationStats.java:169)
    TestPercolationStats.main(TestPercolationStats.java:236)
     java.lang.ArrayIndexOutOfBoundsException: 100
     PercolationStats.<init>(PercolationStats.java:20)
    {\tt TestPercolationStats.twoPercolationStats} ({\tt TestPercolationStat}
s.java:132)
     TestPercolationStats.test3(TestPercolationStats.java:170)
     TestPercolationStats.main(TestPercolationStats.java:236)
     java.lang.ArrayIndexOutOfBoundsException: 200
     PercolationStats.<init>(PercolationStats.java:20)
```

```
{\tt TestPercolationStats.twoPercolationStats} ({\tt TestPercolationStat}
s.java:132)
    TestPercolationStats.test3(TestPercolationStats.java:171)
    TestPercolationStats.main(TestPercolationStats.java:236)
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
    java.lang.ArrayIndexOutOfBoundsException: 200
    PercolationStats.<init>(PercolationStats.java:20)
    TestPercolationStats.test4(TestPercolationStats.java:182)
    TestPercolationStats.main(TestPercolationStats.java:237)
==> FAILED
Total: 0/7 tests passed!
memory usage
Computing memory of Percolation
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
              N
                       bytes
                           _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
            64
                      41864
=> passed
=> passed
           256
                    609032
=> passed
                     2397448
           512
=> passed
           1024
                     9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (R}^2 = 1.)
000)
```

======	=======================================
-	g memory of PercolationStats
	n in thread "main" java.lang.ArrayIndexOutOfBoundsExcept
	at PercolationStats. <init>(PercolationStats.java:20) at MemoryOfPercolationStats.main(MemoryOfPercolationStat</init>
.java:81 Running) 4 total tests.
	1d: Measuring total memory usage as a function of T (max 8 T + 128 bytes)
	T bytes
	T bytes
	/4 tests passed:Test aborted. Ran out of time or crashed ompletion.
before c	/4 tests passed:Test aborted. Ran out of time or crashed
before c	/4 tests passed:Test aborted. Ran out of time or crashed
before c ======= ******	/4 tests passed: Test aborted. Ran out of time or crashed ompletion. ***********************************
before c ======= *******	/4 tests passed:Test aborted. Ran out of time or crashed ompletion.
<pre>before c ====== ******* * timin</pre>	/4 tests passed:Test aborted. Ran out of time or crashed ompletion. ***********************************
******* * timin ****** Timing P	/4 tests passed:Test aborted. Ran out of time or crashed ompletion. ***********************************
******* * timin ****** Timing P	/4 tests passed:Test aborted. Ran out of time or crashed ompletion.
****** * timin ****** Timing P * Running	/4 tests passed:Test aborted. Ran out of time or crashed ompletion.

```
For each N, a percolation object is generated and sites are rando
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                                 2 * connected()
                     seconds
                                   union()
                                                        + find()
       constructor
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                 8 Infinity
                                      21
                                                            30
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                           (0.5x)
                                      68
                                                            30
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               128 Infinity
                                           (0.1x)
                                     263
                                                            230
(0.1x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
```

```
TimePercolation.main(TimePercolation.java:420)
=> FAILED
             512 Infinity
                               1029 (0.0x)
                                                     1554
(0.0x)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
            1024 Infinity 2054 (0.0x)
                                                     3276
(0.0x)
              1
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
and varies each time that you submit. If one of the values in th
e table
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                     per percolates()
()
   java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
```

Submission TimePercolation.main(TimePercolation.java:420) => passed 32 0 0 java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:45) TimePercolation.countMaxOperations(TimePercolation.java:50) TimePercolation.maxOperationCountTest(TimePercolation.java:3 86) TimePercolation.testLite(TimePercolation.java:405) TimePercolation.main(TimePercolation.java:420) => passed 128 0 0 0 java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:45) TimePercolation.countMaxOperations(TimePercolation.java:50) TimePercolation.maxOperationCountTest(TimePercolation.java:3 86) TimePercolation.testLite(TimePercolation.java:405) TimePercolation.main(TimePercolation.java:420) => passed 512 0 0 java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:45) TimePercolation.countMaxOperations(TimePercolation.java:50) TimePercolation.maxOperationCountTest(TimePercolation.java:3 86) TimePercolation.testLite(TimePercolation.java:405) TimePercolation.main(TimePercolation.java:420) => passed 1024 0 ==> 4/4 tests passed Total: 5/9 tests passed! ______

Submission

Raw Score

Feedback

time

Sat-26-Oct 01:51:48

Assess	ment Summary
Compilation:	PASSED
Style:	FAILED
Findbugs:	No potential bugs found.
API:	PASSED
Correctness:	0/20 tests passed
Memory:	4/8 tests passed
Timing:	5/9 tests passed
Raw score: 18	8.89% [Correctness: 65%, Memory: 10%, Timing: 25%,
yle: 0%]	

Assessment Details

% javad	PercolationStats.java
*	
======	:======================================
% check	kstyle *.java
*	
Percola	ationStats.java:18:17: Assignment of parameter 'T' is not
llowed.	
======	
% findh	ougs *.class
	Jugs .Class
======	.======================================
Testing	g the APIs of your programs.
Percola	ation:
Percola	ationStats:
======	
*****	************
*****	****
* exec	cuting
*****	*****************
*****	*****
Tootina	wathada in Danaslation
*	g methods in Percolation
Runnino	g 13 total tests.
	,
Test 1:	: Check whether exception is called if (i, j) are out of b
unds	
	= 10, (i, j) = (0, 6)
	<pre>IndexOutOfBoundsException NOT thrown for open()</pre>
	IndexOutOfBoundsException NOT thrown for isOpen()
-	<pre>IndexOutOfBoundsException NOT thrown for isFull()</pre>

```
N = 10, (i, j) = (12, 6)
  * N = 10, (i, j) = (11, 6)
  * N = 10, (i, j) = (6, 0)
     - IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (6, 12)
 * N = 10, (i, j) = (6, 11)
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the corrrect results.
Test 2: Open predetermined list of sites using files
  * filename = input6.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:196)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:197)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:198)
     TestPercolation.main(TestPercolation.java:540)
```

```
* filename = input10-no.txt
    java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:199)
TestPercolation.main(TestPercolation.java:540)
```

- * filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)
 TestPercolation.main(TestPercolation.java:540)
- * filename = heart25.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:201)
 TestPercolation.main(TestPercolation.java:540)

==> FAILED

Test 3: Open random sites until system percolates (then test is t erminated)

* N = 3
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:235)
TestPercolation.main(TestPercolation.java:541)

N = 5
java.lang.ArrayIndexOutOfBoundsException

Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:236)
TestPercolation.main(TestPercolation.java:541)

* N = 10 java.lang.ArrayIndexOutOfBoundsException Percolation.isFull(Percolation.java:49) TestPercolation.checkIsFull(TestPercolation.java:22) TestPercolation.check(TestPercolation.java:75)

TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:237)
TestPercolation.main(TestPercolation.java:541)

* N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)

TestPercolation.checkIsFull(TestPercolation.java:22)

TestPercolation.check(TestPercolation.java:75)

TestPercolation.random(TestPercolation.java:214)

TestPercolation.test3(TestPercolation.java:238)

TestPercolation.main(TestPercolation.java:541)

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:239)
TestPercolation.main(TestPercolation.java:541)

N = 20
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)
TestPercolation.main(TestPercolation.java:541)

```
N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:241)
     TestPercolation.main(TestPercolation.java:541)
    N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:242)
    TestPercolation.main(TestPercolation.java:541)
==> FAILED
Test 4: Opens predetermined sites, but where N = 1 and N = 2 (cor
ner case test)
  * filename = input1.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:251)
    TestPercolation.main(TestPercolation.java:542)
  * filename = input1-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test4(TestPercolation.java:252)
```

filename = input2.txt
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)

TestPercolation.main(TestPercolation.java:542)

TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test4(TestPercolation.java:253)
TestPercolation.main(TestPercolation.java:542)

* filename = input2-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:254)
 TestPercolation.main(TestPercolation.java:542)

==> FAILED

Test 5: Check for backwash with predetermined sites

- * filename = input20.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:263)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:265)

```
TestPercolation.main(TestPercolation.java:543)
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:275)
    TestPercolation.main(TestPercolation.java:544)
  * filename = input4.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test6(TestPercolation.java:276)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input7.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:277)
     TestPercolation.main(TestPercolation.java:544)
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
```

176 of 296 2013年10月28日 21:20

TestPercolation.file(TestPercolation.java:138)
TestPercolation.test7(TestPercolation.java:287)

```
TestPercolation.main(TestPercolation.java:545)
   filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:288)
     TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test8(TestPercolation.java:296)
    TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:336)
     TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:337)
```

177 of 296 2013年10月28日 21:20

TestPercolation.main(TestPercolation.java:548)

```
java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
    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(),
percolates()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order =
                                        isOpen(), percolates(),
    isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:179)
    TestPercolation.test10(TestPercolation.java:386)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                      isOpen(),
```

```
isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.file(TestPercolation.java:180)
     TestPercolation.test10(TestPercolation.java:387)
     TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:360)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation. random Calls Until Percolation (TestPercolation.\\
java:363)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
```

```
N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     Test Percolation. random {\tt CallsUntil Percolation} (Test Percolation. \\
java:360)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:364)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:458)
     TestPercolation.main(TestPercolation.java:551)
     N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
```

```
TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
     TestPercolation.test12(TestPercolation.java:459)
    TestPercolation.main(TestPercolation.java:551)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsNoBackwash(TestPercolation.java: 4)}\\
31)
    TestPercolation.test12(TestPercolation.java:460)
    TestPercolation.main(TestPercolation.java:551)
    N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:461)
    TestPercolation.main(TestPercolation.java:551)
    N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:462)
    TestPercolation.main(TestPercolation.java:551)
   N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:463)
    TestPercolation.main(TestPercolation.java:551)
==> FAILED
```

```
Test 13: Call all methods in random order until all sites are ope
n
    N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:517)
     TestPercolation.main(TestPercolation.java:552)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:518)
     TestPercolation.main(TestPercolation.java:552)
   N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:519)
     TestPercolation.main(TestPercolation.java:552)
    N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:520)
    TestPercolation.main(TestPercolation.java:552)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:521)
     TestPercolation.main(TestPercolation.java:552)
```

```
* N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
*-----
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
      mean(): 6093.143 (FAILED, outside of range)
      stddev(): 216.861 (FAILED, outside of range)
      Overall result: FAILED
Creating new PercolationStats(200, 10)
_____
PercolationStats reports:
      mean(): 26236.960 (FAILED, outside of range)
      stddev(): 2778.814 (FAILED, outside of range)
      Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
```

```
* confidenceLo() = 5899.30794059368
 * confidenceHi() = 6135.888900788782
==> FAILED
Creating new PercolationStats(200, 10)
  * confidenceLo() = 22203.12292610215
  * confidenceHi() = 28129.78038256141
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
  * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
 (to make sure you didn't store data in static variables)
  * 1mean = 60.30297
    2mean = 58.68
  * 1mean = 236.817945
  * 2mean = 233.92
  * 1mean = 59.6381835
  * 2mean = 59.765
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
  * order = mean(), stddev()
  * mean = 532.1858631250001; stddev = 34.210549958530024
  * order = stddev(), mean()
 * mean = 532.939631875; stddev = 32.724425864637176
==> FAILED
Total: 0/7 tests passed!
 *****************
```

```
* memory usage
****************
Computing memory of Percolation
*-----
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
             N bytes
-----
=> passed 64 41864
=> passed 256 609032
                  2397448
=> passed
          512
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (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
allowed: 8 T + 128 bytes)
             T bytes
=> FAILED 16 1556384 (6e+03x)
=> FAILED 32 3112672 (8e+03x)
=> FAILED 64 6225248 (1e+04x)
=> FAILED 128 12450400 (1e+04x)
```

```
==> 0/4 tests passed
Estimated student memory = 97268.00 \text{ T} + 96.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
 ****************
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 rando
mly opened
until the system percolates. If you do not pass the correctness \boldsymbol{t}
ests, these
results may be meaningless.
                                          2 * connected()
                seconds union()
                                                + find()
      constructor
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
```

```
=> passed
                 8 Infinity
                                       21
                                                             30
                1
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                      68
                                            (0.5x)
                                                             30
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               128 Infinity
                                     263
                                            (0.1x)
                                                            230
(0.1x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               512 Infinity
                                    1029
                                            (0.0x)
                                                           1554
(0.0x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
              1024 Infinity
                                    2054
                                                           3276
                                            (0.0x)
(0.0x)
                1
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
```

```
and varies each time that you submit. If one of the values in th
e table
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                 N
                       per open()
                                       per isOpen()
                                                       per isFull
()
      per percolates()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                32
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
               128
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
```

Submission					
	TimePer	colation.n	naxOperationC	CountTest(TimePerco	olation.java:3
		colation.t	estLite(Time	Percolation.java:	405)
	TimePer	colation.m	nain(TimePerc	colation.java:420)	
	=> passed 0	512	0	0	0
	java.la	ng.ArrayIr	ndexOutOfBour	dsException	
	Percola	tion.isOpe	en(Percolatio	n.java:45)	
			· ·	tions(TimePercola	
		colation.m	naxOperationC	CountTest(TimePerco	olation.java:3
	86)				
			•	Percolation.java:	405)
	TimePer	cotation.	iain(TimePerd	colation.java:420)	
	=> passed	1024	0	0	0
	0				
	==> 4/4 test	s passeu			
	Total: 5/9 t	ests nasse	r d I		
	========	=========	.u. :========	:=========	========

Submission time Raw Score 18.89 / 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: 0/20 tests passed	Submission			
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		Sat-26-Oct 01:05:08		
Assessment Summary Compilation: PASSED Style: PASSED Findbugs: No potential bugs found. API: PASSED	Raw Score	18.89 / 100.00		
	Feedback	Assess compilation: Style: Findbugs: API:	ment Summary PASSED PASSED No potential bugs found. PASSED	

```
Memory: 4/8 tests passed
Timing: 5/9 tests passed

Raw score: 18.89% [Correctness: 65%, Memory: 10%, Timing: 25%, St yle: 0%]
```

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 2.0K Oct 26 08:05 Percolation.java
-rw-r--r-- 1 2.7K Oct 26 08:05 PercolationStats.java
-rw-r--r-- 1 1.8K Oct 26 08:05 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 13 total tests.	
<pre>Test 1: Check whether exception is called if (i, j) are unds * N = 10, (i, j) = (0, 6) - IndexOutOfBoundsException NOT thrown for open() - IndexOutOfBoundsException NOT thrown for isOpen() - IndexOutOfBoundsException NOT thrown for isFull() * N = 10, (i, j) = (12, 6) * N = 10, (i, j) = (11, 6) * N = 10, (i, j) = (6, 0) - IndexOutOfBoundsException NOT thrown for open() - IndexOutOfBoundsException NOT thrown for isOpen() * N = 10, (i, j) = (6, 12) * N = 10, (i, j) = (6, 11) ==> FAILED</pre>))
Tests 2 through 8 create a Percolation object using your en repeatedly open sites using open(i, j). After each call to open, we at isFull(), isOpen(), and percolates() return the corrrect results.	
Test 2: Open predetermined list of sites using files	

* filename = input6.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:196)
 TestPercolation.main(TestPercolation.java:540)

* filename = input8.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:197)
 TestPercolation.main(TestPercolation.java:540)

* filename = input8-no.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:198)
TestPercolation.main(TestPercolation.java:540)

* filename = input10-no.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:199)
TestPercolation.main(TestPercolation.java:540)

* filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)

```
TestPercolation.main(TestPercolation.java:540)
   filename = heart25.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:201)
     TestPercolation.main(TestPercolation.java:540)
==> FAILED
Test 3: Open random sites until system percolates (then test is t
erminated)
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:235)
    TestPercolation.main(TestPercolation.java:541)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:236)
    TestPercolation.main(TestPercolation.java:541)
   N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:237)
    TestPercolation.main(TestPercolation.java:541)
```

193 of 296 2013年10月28日 21:20

N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:238)
TestPercolation.main(TestPercolation.java:541)

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)

N = 20

TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:239)
TestPercolation.main(TestPercolation.java:541)

* N = 20
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)
TestPercolation.main(TestPercolation.java:541)

N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:241)
TestPercolation.main(TestPercolation.java:541)

* N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:242)
TestPercolation.main(TestPercolation.java:541)

194 of 296

```
==> FAILED
Test 4: Opens predetermined sites, but where N = 1 and N = 2 (cor
ner case test)
  * filename = input1.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test4(TestPercolation.java:251)
    TestPercolation.main(TestPercolation.java:542)
  * filename = input1-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:252)
    TestPercolation.main(TestPercolation.java:542)
  * filename = input2.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:253)
    TestPercolation.main(TestPercolation.java:542)
  * filename = input2-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:254)
    TestPercolation.main(TestPercolation.java:542)
```

195 of 296 2013年10月28日 21:20

==> FAILED

```
Test 5: Check for backwash with predetermined sites
  * filename = input20.txt
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test5(TestPercolation.java:263)
    TestPercolation.main(TestPercolation.java:543)
```

- * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:265)
 TestPercolation.main(TestPercolation.java:543)

==> FAILED

Test 6: Check for backwash with predetermined sites that havemult iple percolating paths

- * filename = input3.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:275)
 TestPercolation.main(TestPercolation.java:544)
- * filename = input4.txt
 java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test6(TestPercolation.java:276)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input7.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test6(TestPercolation.java:277)
     TestPercolation.main(TestPercolation.java:544)
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test7(TestPercolation.java:287)
     TestPercolation.main(TestPercolation.java:545)
   filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test7(TestPercolation.java:288)
     TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
```

197 of 296 2013年10月28日 21:20

Percolation.isFull(Percolation.java:49)

```
TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test8(TestPercolation.java:296)
    TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:336)
     TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:337)
    TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
         but change the order in which methods are called
  * filename = input8.txt; order =
                                         isFull(),
                                                       isOpen(),
percolates()
    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(),
percolates()
     java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = isOpen(), percolates(),
    isFull()
    java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:179)
    TestPercolation.test10(TestPercolation.java:386)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                      isOpen(),
    isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:180)
    TestPercolation.test10(TestPercolation.java:387)
    TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
```

```
m percolates
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     {\tt TestPercolation.randomCallsUntilPercolation}. \\ {\tt TestPercolation.randomCallsUntilPercolation}.
java:360)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:360)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:364)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
```

```
N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:364)
    TestPercolation.test11(TestPercolation.java:403)
    TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
     TestPercolation.test12(TestPercolation.java:458)
    TestPercolation.main(TestPercolation.java:551)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:459)
    TestPercolation.main(TestPercolation.java:551)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:460)
    TestPercolation.main(TestPercolation.java:551)
    N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
```

```
TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:461)
    TestPercolation.main(TestPercolation.java:551)
    N = 20
     java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:462)
     TestPercolation.main(TestPercolation.java:551)
   N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:463)
     TestPercolation.main(TestPercolation.java:551)
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:517)
    TestPercolation.main(TestPercolation.java:552)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:518)
     TestPercolation.main(TestPercolation.java:552)
```

```
N = 7
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:519)
    TestPercolation.main(TestPercolation.java:552)
   N = 10
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:520)
    TestPercolation.main(TestPercolation.java:552)
  * N = 20
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:521)
    TestPercolation.main(TestPercolation.java:552)
  * N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
```

```
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
      mean(): 0.000 (FAILED, outside of range)
      stddev(): 0.000 (FAILED, outside of range)
      Overall result: FAILED
Creating new PercolationStats(200, 10)
-----
PercolationStats reports:
      mean(): 0.000 (FAILED, outside of range)
      stddev(): 0.000 (FAILED, outside of range)
      Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
 * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Creating new PercolationStats(200, 10)
-----
  * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0

    IllegalArgumentException NOT thrown for PercolationStats()

 * N = -42, T = 0
```

```
- IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
   1mean = 0.0
 * 2mean = 0.0
 * 1mean = 0.0
 * 2mean = 0.0
 * 1mean = 0.0
 * 2mean = 0.0
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * mean = 0.0; stddev = 0.0
 * order = stddev(), mean()
 * mean = 0.0; stddev = 0.0
==> FAILED
Total: 0/7 tests passed!
______
* * * * * * * * * * * *
 memory usage
******************
Computing memory of Percolation
*------
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
            N
                  bytes
=> passed 64
                  41864
=> passed
          256
                609032
=> passed 512 2397448
=> passed 1024
                9513224
```

```
==> 4/4 tests passed
 Estimated student memory = 9.00 \, \text{N}^2 + 74.00 \, \text{N} + 264.00 \, (\text{R}^2 = 1.00 \, \text{N}^2 + 1.00 \, \text{N}^2 +
 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
 allowed: 8 T + 128 bytes)
                                                                 Т
                                                                                                    bytes
 -----
=> FAILED 16 1556384 (6e+03x)
=> FAILED 32 3112672 (8e+03x)
=> FAILED 64 6225248 (1e+04x)
 => FAILED 128 12450400 (1e+04x)
 ==> 0/4 tests passed
 Estimated student memory = 97268.00 \text{ T} + 96.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
 ______
             timing
                                                      ***********
 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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                               2 * connected()
                    seconds
                                 union()
                                                      + find()
      constructor
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> passed
                8 Infinity
                                                          30
                                    21
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
               32 Infinity
                                    68
                                        (0.5x)
                                                          30
(0.2x)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
```

```
=> FAILED
               128 Infinity
                                     263
                                           (0.1x)
                                                           230
(0.1x)
                1
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
                                                          1554
               512 Infinity
                                    1029
                                           (0.0x)
(0.0x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
              1024 Infinity
                                    2054
                                         (0.0x)
                                                          3276
(0.0x)
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
and varies each time that you submit. If one of the values in th
e table
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                                                       per isFull
                       per open()
                                       per isOpen()
      per percolates()
()
```

```
java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
    TimePercolation.main(TimePercolation.java:420)
                32
=> passed
                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
               128
                          0
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
         0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
              1024
         0
==> 4/4 tests passed
```

Total: 5/9 tests passed!

Submission					
Submission time	Sat-26-Oct 01:04:21				
Raw Score	18.89 / 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: 0/20 tests passed Memory: 4/8 tests passed Timing: 5/9 tests passed Raw score: 18.89% [Correctness: 65%, Memory: 10%, Timing: 25%, St yle: 0%]				

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 2.0K Oct 26 08:04 Percolation.java
-rw-r--r-- 1 2.7K Oct 26 08:04 PercolationStats.java
-rw-r--r-- 1 1.8K Oct 26 08:04 studentSubmission.zip
```

Submission	
	******* * compiling ***********************************
	% javac Percolation.java *
	% javac PercolationStats.java *
	% checkstyle *.java *
	% findbugs *.class *
	Testing the APIs of your programs. * Percolation:
	Percolation: ===================================

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

```
Running 13 total tests.
Test 1: Check whether exception is called if (i, j) are out of bo
unds
  * N = 10, (i, j) = (0, 6)
     - IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (12, 6)
  * N = 10, (i, j) = (11, 6)
  * N = 10, (i, j) = (6, 0)
     - IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()

    IndexOutOfBoundsException NOT thrown for isFull()

  * N = 10, (i, j) = (6, 12)
  * N = 10, (i, j) = (6, 11)
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the corrrect results.
Test 2: Open predetermined list of sites using files
  * filename = input6.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:196)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:197)
     TestPercolation.main(TestPercolation.java:540)
```

```
* filename = input8-no.txt
    java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:198)
TestPercolation.main(TestPercolation.java:540)
```

- * filename = input10-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:199)
 TestPercolation.main(TestPercolation.java:540)
- * filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)
 TestPercolation.main(TestPercolation.java:540)
- * filename = heart25.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:201)
 TestPercolation.main(TestPercolation.java:540)

==> FAILED

Test 3: Open random sites until system percolates (then test is t erminated)

* N = 3
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)

TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:235)
TestPercolation.main(TestPercolation.java:541)

N = 5
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:236)
TestPercolation.main(TestPercolation.java:541)

* N = 10
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.random(TestPercolation.java:214)
 TestPercolation.test3(TestPercolation.java:237)
 TestPercolation.main(TestPercolation.java:541)

* N = 10
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.random(TestPercolation.java:214)
 TestPercolation.test3(TestPercolation.java:238)
 TestPercolation.main(TestPercolation.java:541)

N = 20
java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:239)
TestPercolation.main(TestPercolation.java:541)

* N = 20

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)
TestPercolation.main(TestPercolation.java:541)

* N = 50

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:241)
TestPercolation.main(TestPercolation.java:541)

* N = 50

java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:242)
TestPercolation.main(TestPercolation.java:541)

==> FAILED

Test 4: Opens predetermined sites, but where N=1 and N=2 (cor ner case test)

* filename = input1.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test4(TestPercolation.java:251)
TestPercolation.main(TestPercolation.java:542)

* filename = input1-no.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)

215 of 296

```
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test4(TestPercolation.java:252)
TestPercolation.main(TestPercolation.java:542)
```

- * filename = input2.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:253)
 TestPercolation.main(TestPercolation.java:542)
- * filename = input2-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:254)
 TestPercolation.main(TestPercolation.java:542)

==> FAILED

Test 5: Check for backwash with predetermined sites

- * filename = input20.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:263)
 TestPercolation.main(TestPercolation.java:543)
 - * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)

* filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test5(TestPercolation.java:265)
TestPercolation.main(TestPercolation.java:543)

==> FAILED

Test 6: Check for backwash with predetermined sites that havemult iple percolating paths

* filename = input3.txt
 java.lang.ArrayIndexOutOfBoundsException
Percolation.isFull(Percolation.java:49)
TestPercolation.checkIsFull(TestPercolation.java:22)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test6(TestPercolation.java:275)
TestPercolation.main(TestPercolation.java:544)

* filename = input4.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:276)
 TestPercolation.main(TestPercolation.java:544)

* filename = input7.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isFull(Percolation.java:49)
 TestPercolation.checkIsFull(TestPercolation.java:22)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:277)
 TestPercolation.main(TestPercolation.java:544)

==> FAILED

```
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:287)
    TestPercolation.main(TestPercolation.java:545)
   filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:288)
     TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test8(TestPercolation.java:296)
     TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:336)
     TestPercolation.main(TestPercolation.java:548)
```

```
java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:337)
    TestPercolation.main(TestPercolation.java:548)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
    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(),
percolates()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = isOpen(), percolates(),
    isFull()
```

```
java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.file(TestPercolation.java:179)
     TestPercolation.test10(TestPercolation.java:386)
     TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                        isOpen(),
    isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.file(TestPercolation.java:180)
     TestPercolation.test10(TestPercolation.java:387)
     TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:360)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
```

```
N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. \\ {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
    N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     Test Percolation. random {\tt CallsUntilPercolation} (Test Percolation. \\
java:364)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     Test Percolation. random Calls Until Percolation (Test Percolation.\\
java:364)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:21)
```

```
TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
    TestPercolation.test12(TestPercolation.java:458)
     TestPercolation.main(TestPercolation.java:551)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:459)
    TestPercolation.main(TestPercolation.java:551)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:460)
    TestPercolation.main(TestPercolation.java:551)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:461)
    TestPercolation.main(TestPercolation.java:551)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:462)
     TestPercolation.main(TestPercolation.java:551)
    N = 50
     java.lang.ArrayIndexOutOfBoundsException
```

222 of 296

```
Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
37)
    TestPercolation.test12(TestPercolation.java:463)
    TestPercolation.main(TestPercolation.java:551)
==> FAILED
Test 13: Call all methods in random order until all sites are ope
    N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:517)
     TestPercolation.main(TestPercolation.java:552)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:518)
     TestPercolation.main(TestPercolation.java:552)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TestPercolation.checkIsFull(TestPercolation.java:22)
     TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:519)
    TestPercolation.main(TestPercolation.java:552)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:45)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:520)
     TestPercolation.main(TestPercolation.java:552)
```

```
N = 20
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isFull(Percolation.java:49)
    TestPercolation.checkIsFull(TestPercolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:496)
    TestPercolation.test13(TestPercolation.java:521)
    TestPercolation.main(TestPercolation.java:552)
  * N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 0.000 (FAILED, outside of range)
       stddev(): 0.000 (FAILED, outside of range)
       Overall result: FAILED
Creating new PercolationStats(200, 10)
_____
PercolationStats reports:
       mean(): 0.000 (FAILED, outside of range)
```

```
stddev(): 0.000 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
 * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Creating new PercolationStats(200, 10)
  * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
 (to make sure you didn't store data in static variables)
  * 1mean = 0.0
  * 2mean = 0.0
  * 1mean = 0.0
  * 2mean = 0.0
  * 1mean = 0.0
  * 2mean = 0.0
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
  * mean = 0.0; stddev = 0.0
  * order = stddev(), mean()
```

```
* mean = 0.0; stddev = 0.0
==> FAILED
Total: 0/7 tests passed!
______
* memory usage
Computing memory of Percolation
*_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
          N bytes
-----
=> passed 64 41864
=> passed 256 609032
=> passed
        512
              2397448
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (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
```

```
allowed: 8 T + 128 bytes)
             Т
                    bytes
=> FAILED 16 1556384 (6e+03x)
                  3112672 (8e+03x)
=> FAILED
            32
=> FAILED 64
=> FAILED 128
            64
                  6225248 (1e+04x)
                  12450400 (1e+04x)
==> 0/4 tests passed
Estimated student memory = 97268.00 \text{ T} + 96.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
  timing
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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                         2 * connected()
                 seconds
                             union()
                                               + find()
     constructor
```

```
java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                 8 Infinity
                                      21
                                                             30
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                      68
                                                             30
                                            (0.5x)
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               128 Infinity
                                           (0.1x)
                                     263
                                                            230
(0.1x)
                1
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               512 Infinity
                                            (0.0x)
                                    1029
                                                           1554
(0.0x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isFull(Percolation.java:49)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
```

```
TimePercolation.main(TimePercolation.java:420)
=> FAILED
             1024 Infinity
                                 2054 (0.0x)
                                                        3276
(0.0x)
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
and varies each time that you submit. If one of the values in th
e table
violates the performance limits, the factor by which you failed t
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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                      per isFull
()
     per percolates()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
    TimePercolation.main(TimePercolation.java:420)
=> passed
               32
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:45)
    TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
```

Submission					
	TimePer	colation.t	estLite(Time	ePercolation.java:	405)
	TimePer	colation.m	ain(TimePerd	colation.java:420)	
	=> passed	128	0	0	0
	0				
				ndsException	
		-	n(Percolatio	-	
			•	ations(TimePercola	
		colation.ma	axOperation(CountTest(TimePerco	olation.java:3
	86)				
			-	ePercolation.java:	405)
	TimePer	colation.ma	ain(TimePerd	colation.java:420)	
	_				
	=> passed	512	0	0	0
	0	_			
	_	-		ndsException	
		•	n(Percolatio	•	
			•	ations(TimePercola	• .
		colation.ma	axuperation	CountTest(TimePerco	olation.java:3
	86)	1			405)
			-	ePercolation.java:	405)
	TimePer	JULALIUII.III	ain(TimePerc	colation.java:420)	
	=> passed	1024	0	0	0
	0				
	==> 4/4 tests	s passed			
	Total: 5/9 to	ests passe	d !		
		•			========

Submission		
Submission time	Sat-26-Oct 00:31:23	
Raw Score	18.89 / 100.00	
Feedback	See the Assessment Guide for information on how to read this report.	
	Assessment Summary	

Compilation: PASSED Style: FAILED

Findbugs: No potential bugs found.

API: PASSED

Correctness: 0/20 tests passed
Memory: 4/8 tests passed
Timing: 5/9 tests passed

Raw score: 18.89% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 1.6K Oct 26 07:31 Percolation.java
-rw-r--r-- 1 2.2K Oct 26 07:31 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 07:31 studentSubmission.zip
  ****************
 compiling
      *************
% javac Percolation.java
______
% javac PercolationStats.java
% checkstyle *.java
*_____
```

	plation.java:2:1: File contains tab characters (this is the f
	instance).
	olationStats.java:5:1: File contains tab characters (this is
tne	first instance).
====	
% fir	ndbugs *.class
*	
Test:	ing the APIs of your programs.
Perc	olation:
Perco	olationStats:
====	
de de de de .	***********

	xecuting ************************************
***	*****
Test:	ing methods in Percolation
*	
Runn	ing 13 total tests.
	1: Check whether exception is called if (i, j) are out of bo
unds *	N = 10, (i, j) = (0, 6)
	- IndexOutOfBoundsException NOT thrown for open()
	- IndexOutOfBoundsException NOT thrown for isOpen()
	- IndexOutOfBoundsException NOT thrown for isFull()
*	N = 10, (i, j) = (12, 6)
*	N = 10, (1, j) = (11, 6) N = 10, (i, j) = (11, 6)
*	N = 10, (1, j) = (11, 0) N = 10, (1, j) = (6, 0)
	- IndexOutOfBoundsException NOT thrown for open()
	- IndexOutOfBoundsException NOT thrown for isOpen()
	- IndexOutOfBoundsException NOT thrown for isFull()

```
* N = 10, (i, j) = (6, 12)
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (6, 11)
     - IndexOutOfBoundsException NOT thrown for isFull()
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the corrrect results.
Test 2: Open predetermined list of sites using files
  * filename = input6.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test2(TestPercolation.java:196)
    TestPercolation.main(TestPercolation.java:540)
   filename = input8.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:197)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test2(TestPercolation.java:198)
     TestPercolation.main(TestPercolation.java:540)
  * filename = input10-no.txt
     java.lang.ArrayIndexOutOfBoundsException
```

233 of 296 2013年10月28日 21:20

Percolation.isOpen(Percolation.java:46)

```
TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test2(TestPercolation.java:199)
     TestPercolation.main(TestPercolation.java:540)
   filename = greeting57.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:200)
     TestPercolation.main(TestPercolation.java:540)
  * filename = heart25.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test2(TestPercolation.java:201)
     TestPercolation.main(TestPercolation.java:540)
==> FAILED
Test 3: Open random sites until system percolates (then test is t
erminated)
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:235)
    TestPercolation.main(TestPercolation.java:541)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
```

234 of 296 2013年10月28日 21:20

TestPercolation.random(TestPercolation.java:214)

TestPercolation.test3(TestPercolation.java:236)
TestPercolation.main(TestPercolation.java:541)

* N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:237)
TestPercolation.main(TestPercolation.java:541)

* N = 10

java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:238)
TestPercolation.main(TestPercolation.java:541)

java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)

TestPercolation.test3(TestPercolation.java:239)

TestPercolation.main(TestPercolation.java:541)

* N = 20

N = 20

java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)
TestPercolation.main(TestPercolation.java:541)

N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)

```
TestPercolation.check(TestPercolation.java:75)
     TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:241)
     TestPercolation.main(TestPercolation.java:541)
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:242)
    TestPercolation.main(TestPercolation.java:541)
==> FAILED
Test 4: Opens predetermined sites, but where N = 1 and N = 2 (cor
ner case test)
  * filename = input1.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:251)
     TestPercolation.main(TestPercolation.java:542)
  * filename = input1-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:252)
     TestPercolation.main(TestPercolation.java:542)
  * filename = input2.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
```

236 of 296 2013年10月28日 21:20

TestPercolation.test4(TestPercolation.java:253)

filename = input2-no.txt
java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test4(TestPercolation.java:254)

TestPercolation.main(TestPercolation.java:542)

TestPercolation.main(TestPercolation.java:542)

==> FAILED

Test 5: Check for backwash with predetermined sites

- * filename = input20.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:263)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)
- * filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:265)
 TestPercolation.main(TestPercolation.java:543)

==> FAILED

237 of 296

```
Test 6: Check for backwash with predetermined sites that havemult iple percolating paths
```

- * filename = input3.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:275)
 TestPercolation.main(TestPercolation.java:544)
- * filename = input4.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:276)
 TestPercolation.main(TestPercolation.java:544)
- * filename = input7.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:277)
 TestPercolation.main(TestPercolation.java:544)

==> FAILED

Test 7: Predetermined sites with very long percolating path

* filename = snake13.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test7(TestPercolation.java:287)
 TestPercolation.main(TestPercolation.java:545)

* filename = snake101.txt
 java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:288)
    TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test8(TestPercolation.java:296)
     TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:336)
     TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:337)
     TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
```

239 of 296 2013年10月28日 21:20

```
TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
    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(),
percolates()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order =
                                        isOpen(), percolates(),
   isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:179)
    TestPercolation.test10(TestPercolation.java:386)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                      isOpen(),
   isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
```

```
TestPercolation.file(TestPercolation.java:180)
     TestPercolation.test10(TestPercolation.java:387)
     TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:360)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:360)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
```

```
TestPercolation. random Calls Until Percolation (TestPercolation.\\
java:363)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
   N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
   N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:458)
     TestPercolation.main(TestPercolation.java:551)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
     TestPercolation.test12(TestPercolation.java:459)
     TestPercolation.main(TestPercolation.java:551)
```

```
N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:460)
    TestPercolation.main(TestPercolation.java:551)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:461)
     TestPercolation.main(TestPercolation.java:551)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:462)
     TestPercolation.main(TestPercolation.java:551)
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:463)
    TestPercolation.main(TestPercolation.java:551)
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
```

Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.randomCalls(TestPercolation.java:490)
TestPercolation.test13(TestPercolation.java:517)
TestPercolation.main(TestPercolation.java:552)

N = 5
java.lang.ArrayIndexOutOfBoundsException
Percolation.open(Percolation.java:22)
TestPercolation.randomCalls(TestPercolation.java:484)

TestPercolation.test13(TestPercolation.java:518)

TestPercolation.main(TestPercolation.java:552)

* N = 7

java.lang.ArrayIndexOutOfBoundsException
Percolation.open(Percolation.java:22)
TestPercolation.randomCalls(TestPercolation.java:484)
TestPercolation.test13(TestPercolation.java:519)
TestPercolation.main(TestPercolation.java:552)

* N = 10
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.randomCalls(TestPercolation.java:490)
 TestPercolation.test13(TestPercolation.java:520)

TestPercolation.main(TestPercolation.java:552)

* N = 20
java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.randomCalls(TestPercolation.java:490)
TestPercolation.test13(TestPercolation.java:521)
TestPercolation.main(TestPercolation.java:552)

* N = 50
java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.randomCalls(TestPercolation.java:490)
TestPercolation.test13(TestPercolation.java:522)
TestPercolation.main(TestPercolation.java:552)

```
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
*_____
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
_____
PercolationStats reports:
      mean(): 0.000 (FAILED, outside of range)
      stddev(): 0.000 (FAILED, outside of range)
      Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
      mean(): 0.000 (FAILED, outside of range)
      stddev(): 0.000 (FAILED, outside of range)
      Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
 * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Creating new PercolationStats(200, 10)
```

```
* confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
 * N = -23, T = 42
   - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = -42, T = 0

    IllegalArgumentException NOT thrown for PercolationStats()

==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
 * 1mean = 0.0
 * 2mean = 0.0
 * 1mean = 0.0
 * 2mean = 0.0
 * 1mean = 0.0
 * 2mean = 0.0
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
 * order = mean(), stddev()
 * mean = 0.0; stddev = 0.0
 * order = stddev(), mean()
 * mean = 0.0; stddev = 0.0
==> FAILED
Total: 0/7 tests passed!
______
memory usage
Computing memory of Percolation
```

```
Running 4 total tests.
```

Test 1a-1d: Measuring total memory usage as a function of grid si ze (max allowed: $17 \text{ N}^2 + 128 \text{ N} + 1024 \text{ bytes}$)

	N	bytes	
=> passed	64	41864	
=> passed	256	609032	
=> passed	512	2397448	
=> passed	1024	9513224	
==> 4/4 tests	passed		

Estimated student memory = $9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (R}^2 = 1.000)$

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 allowed: 8 T + 128 bytes)

```
T bytes

=> FAILED 16 1556384 (6e+03x)

=> FAILED 32 3112672 (8e+03x)

=> FAILED 64 6225248 (1e+04x)

=> FAILED 128 12450400 (1e+04x)

==> 0/4 tests passed
```

Estimated student memory = $97268.00 \text{ T} + 96.00 \text{ (R}^2 = 1.000)$

Total: 0/4 tests passed!

```
______
  timing
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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                           2 * connected()
                  seconds
                              union()
                                                 + find()
               N
      constructor
 _____
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> passed
              8 Infinity
                                 21
                                                     32
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
```

```
TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                      68
                                           (0.5x)
                                                             32
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               128 Infinity
                                     263
                                            (0.1x)
                                                            232
(0.1x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               512 Infinity
                                    1029
                                            (0.0x)
                                                           1556
(0.0x)
                1
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
              1024 Infinity
                                    2054
                                           (0.0x)
                                                           3278
(0.0x)
                1
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
and varies each time that you submit. If one of the values in th
e table
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                       per open()
                                       per isOpen()
                                                       per isFull
()
     per percolates()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                32
                                          0
                                                          0
                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
=> passed
               128
                          0
                                          0
                                                          0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
    TimePercolation.main(TimePercolation.java:420)
=> passed
               512
                                          0
                                                          0
         0
```

Submission				
	<pre>java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TimePercolation.countMaxOperations(TimePercolation.java:50) TimePercolation.maxOperationCountTest(TimePercolation.java:3 86) TimePercolation.testLite(TimePercolation.java:405) TimePercolation.main(TimePercolation.java:420)</pre>			
	=> passed 1024 0 0 0 0 0 ==> 4/4 tests passed Total: 5/9 tests passed!			

Sat-26-Oct 00:09:18		
18.89 / 100.00		
See the Assessment Guide for information on how to read this report. Assessment Summary		
Style:	PASSED FAILED No potential bugs found. PASSED	
Memory:	0/20 tests passed 4/8 tests passed 5/9 tests passed	
	18.89 / 100.00 See the Assessme ASSESSI Compilation: Style: Findbugs: API: Correctness: Memory:	

```
The following files were submitted:
______
total 12K
-rw-r--r-- 1 1.6K Oct 26 07:09 Percolation.java
-rw-r--r-- 1 2.3K Oct 26 07:09 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 07:09 studentSubmission.zip
*************
 compiling
% javac Percolation.java
______
% javac PercolationStats.java
______
% checkstyle *.java
*-----
Percolation.java:2:1: File contains tab characters (this is the f
irst instance).
Percolation.java:3:38: Name 'perco_uf' must match pattern '^[a-z]
[a-zA-Z0-9]*$|^[A-Z][A-Z_0-9]*$'.
Percolation.java:4:29: Name 'open_status' must match pattern '^[a
-z][a-zA-Z0-9]*$|^[A-Z][A-Z_0-9]*$'.
Percolation.java:12:21: Name 'bottom_line' must match pattern '^[
a-z][a-zA-Z0-9]*$|^[A-Z][A-Z_0-9]*$'.
Percolation.java:13:20: 'for' is not followed by whitespace.
Percolation.java:21:19: 'if' is not followed by whitespace.
Percolation.java:25:19: 'if' is not followed by whitespace.
Percolation.java:25:38: Expression can be simplified.
Percolation.java:29:19: 'if' is not followed by whitespace.
Percolation.java:32:19: 'if' is not followed by whitespace.
Percolation.java:35:19: 'if' is not followed by whitespace.
```

Percolation.java:38:19: 'if' is not followed by whitespace.
Percolation.java:43:19: 'if' is not followed by whitespace.
Percolation.java:52:19: 'if' is not followed by whitespace.
PercolationStats.java:6:1: File contains tab characters (this is
the first instance).
PercolationStats.java:13:20: 'for' is not followed by whitespace.
PercolationStats.java:20:20: 'for' is not followed by whitespace.
PercolationStats.java:27:20: 'for' is not followed by whitespace.
PercolationStats.java:41:21: Name 'count_percolate' must match pa
ttern '^[a-z][a-zA-Z0-9]*\$ ^[A-Z][A-Z_0-9]*\$'.
PercolationStats.java:42:19: 'if' is not followed by whitespace.
PercolationStats.java:42:37: '{' is not preceded with whitespace.
PercolationStats.java:56:14: 'while' is not followed by whitespac
e.
PercolationStats.java:73: Use x*x instead of Math.pow(x, 2)
PercolationStats.java:74: Use x*x instead of Math.pow(x, 2)
PercolationStats.java:75: Use x*x instead of Math.pow(x, 2)
% findbugs *.class
*
*
Testing the APIs of your programs.
Testing the APIs of your programs. * Percolation:
Testing the APIs of your programs. * Percolation:
Testing the APIs of your programs. * Percolation:
Testing the APIs of your programs. * Percolation:
Testing the APIs of your programs. * Percolation:
Testing the APIs of your programs. * Percolation: PercolationStats:
Testing the APIs of your programs. * Percolation: PercolationStats:
Testing the APIs of your programs. Percolation: PercolationStats: ***********************************
Testing the APIs of your programs. * Percolation: PercolationStats: ***********************************
Testing the APIs of your programs. * Percolation: PercolationStats: ***********************************
Testing the APIs of your programs. * Percolation: ****************************
Testing the APIs of your programs. * Percolation: PercolationStats: ***********************************
Testing the APIs of your programs. * Percolation: ****************************

```
Test 1: Check whether exception is called if (i, j) are out of bo
unds
  * N = 10, (i, j) = (0, 6)

    IndexOutOfBoundsException NOT thrown for open()

     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (12, 6)
  * N = 10, (i, j) = (11, 6)
  * N = 10, (i, j) = (6, 0)
    - IndexOutOfBoundsException NOT thrown for open()
     - IndexOutOfBoundsException NOT thrown for isOpen()
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (6, 12)
     - IndexOutOfBoundsException NOT thrown for isFull()
  * N = 10, (i, j) = (6, 11)
     - IndexOutOfBoundsException NOT thrown for isFull()
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the corrrect results.
Test 2: Open predetermined list of sites using files
  * filename = input6.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:196)
    TestPercolation.main(TestPercolation.java:540)
  * filename = input8.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:197)
     TestPercolation.main(TestPercolation.java:540)
```

```
* filename = input8-no.txt
    java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:198)
TestPercolation.main(TestPercolation.java:540)
```

- * filename = input10-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:199)
 TestPercolation.main(TestPercolation.java:540)
- * filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)
 TestPercolation.main(TestPercolation.java:540)
- * filename = heart25.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:201)
 TestPercolation.main(TestPercolation.java:540)

==> FAILED

Test 3: Open random sites until system percolates (then test is t erminated)

* N = 3 java.lang.ArrayIndexOutOfBoundsException

Percolation.isOpen(Percolation.java:46) TestPercolation.checkIsOpen(TestPercolation.java:43) TestPercolation.check(TestPercolation.java:75) TestPercolation.random(TestPercolation.java:214) TestPercolation.test3(TestPercolation.java:235) TestPercolation.main(TestPercolation.java:541)

* N = 5java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TestPercolation.checkIsOpen(TestPercolation.java:43) TestPercolation.check(TestPercolation.java:75) TestPercolation.random(TestPercolation.java:214)

TestPercolation.test3(TestPercolation.java:236)

TestPercolation.main(TestPercolation.java:541)

N = 10

java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TestPercolation.checkIsOpen(TestPercolation.java:43) TestPercolation.check(TestPercolation.java:75) TestPercolation.random(TestPercolation.java:214) TestPercolation.test3(TestPercolation.java:237) TestPercolation.main(TestPercolation.java:541)

N = 10java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TestPercolation.checkIsOpen(TestPercolation.java:43) TestPercolation.check(TestPercolation.java:75) TestPercolation.random(TestPercolation.java:214) TestPercolation.test3(TestPercolation.java:238)

TestPercolation.main(TestPercolation.java:541)

N = 20

java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TestPercolation.checkIsOpen(TestPercolation.java:43) TestPercolation.check(TestPercolation.java:75) TestPercolation.random(TestPercolation.java:214) TestPercolation.test3(TestPercolation.java:239) TestPercolation.main(TestPercolation.java:541)

```
N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:240)
     TestPercolation.main(TestPercolation.java:541)
    N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:241)
    TestPercolation.main(TestPercolation.java:541)
    N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:242)
     TestPercolation.main(TestPercolation.java:541)
==> FAILED
Test 4: Opens predetermined sites, but where N = 1 and N = 2 (cor
ner case test)
  * filename = input1.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:251)
     TestPercolation.main(TestPercolation.java:542)
   filename = input1-no.txt
```

257 of 296 2013年10月28日 21:20

java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)

```
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test4(TestPercolation.java:252)
TestPercolation.main(TestPercolation.java:542)
```

- * filename = input2.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:253)
 TestPercolation.main(TestPercolation.java:542)
- * filename = input2-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test4(TestPercolation.java:254)
 TestPercolation.main(TestPercolation.java:542)

==> FAILED

Test 5: Check for backwash with predetermined sites

- * filename = input20.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:263)
 TestPercolation.main(TestPercolation.java:543)
 - * filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)

```
TestPercolation.main(TestPercolation.java:543)
   filename = input50.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test5(TestPercolation.java:265)
     TestPercolation.main(TestPercolation.java:543)
==> FAILED
Test 6: Check for backwash with predetermined sites that havemult
iple percolating paths
  * filename = input3.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test6(TestPercolation.java:275)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input4.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:276)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input7.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:277)
    TestPercolation.main(TestPercolation.java:544)
```

259 of 296 2013年10月28日 21:20

==> FAILED

```
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:287)
     TestPercolation.main(TestPercolation.java:545)
  * filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test7(TestPercolation.java:288)
     TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test8(TestPercolation.java:296)
    TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:336)
     TestPercolation.main(TestPercolation.java:548)
```

260 of 296 2013年10月28日 21:20

```
java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:337)
    TestPercolation.main(TestPercolation.java:548)
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
        but change the order in which methods are called
  * filename = input8.txt; order =
                                        isFull(),
                                                      isOpen(),
percolates()
    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(),
percolates()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = isOpen(), percolates(),
```

```
isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:179)
    TestPercolation.test10(TestPercolation.java:386)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                        isOpen(),
    isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.file(TestPercolation.java:180)
    TestPercolation.test10(TestPercolation.java:387)
    TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    Test Percolation. random Calls Until Percolation (Test Percolation.\\
java:363)
    TestPercolation.test11(TestPercolation.java:398)
    TestPercolation.main(TestPercolation.java:550)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
    TestPercolation.test11(TestPercolation.java:399)
```

262 of 296 2013年10月28日 21:20

```
TestPercolation.main(TestPercolation.java:550)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
  * N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
```

```
java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
    TestPercolation.test12(TestPercolation.java:458)
    TestPercolation.main(TestPercolation.java:551)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:459)
    TestPercolation.main(TestPercolation.java:551)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:460)
     TestPercolation.main(TestPercolation.java:551)
   N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
    TestPercolation.test12(TestPercolation.java:461)
    TestPercolation.main(TestPercolation.java:551)
   N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:462)
    TestPercolation.main(TestPercolation.java:551)
    N = 50
```

```
java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:463)
    TestPercolation.main(TestPercolation.java:551)
==> FAILED
Test 13: Call all methods in random order until all sites are ope
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:517)
    TestPercolation.main(TestPercolation.java:552)
   N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:484)
    TestPercolation.test13(TestPercolation.java:518)
     TestPercolation.main(TestPercolation.java:552)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:484)
    TestPercolation.test13(TestPercolation.java:519)
    TestPercolation.main(TestPercolation.java:552)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:484)
     TestPercolation.test13(TestPercolation.java:520)
    TestPercolation.main(TestPercolation.java:552)
    N = 20
     java.lang.ArrayIndexOutOfBoundsException
```

265 of 296

```
Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:521)
    TestPercolation.main(TestPercolation.java:552)
  * N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 0.000 (FAILED, outside of range)
       stddev(): 0.000 (FAILED, outside of range)
       Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 0.000 (FAILED, outside of range)
       stddev(): 0.000 (FAILED, outside of range)
```

```
Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
  * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Creating new PercolationStats(200, 10)
  * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
     - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
  * 1mean = 0.0
  * 2mean = 0.0
  * 1mean = 0.0
  * 2mean = 0.0
  * 1mean = 0.0
  * 2mean = 0.0
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
  * order = mean(), stddev()
  * mean = 0.0; stddev = 0.0
  * order = stddev(), mean()
  * mean = 0.0; stddev = 0.0
==> FAILED
```

```
Total: 0/7 tests passed!
______
******************
*****
 memory usage
*****************
Computing memory of Percolation
*_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
            N
                 bytes
-----
=> passed 64
                 41864
=> passed 256 609032
=> passed 512 2397448
=> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \text{ N}^2 + 74.00 \text{ N} + 264.00 \text{ (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
allowed: 8 T + 128 bytes)
```

```
Т
                     bytes
=> FAILED
            16
                  1556384 (6e+03x)
=> FAILED
            32
                  3112672 (8e+03x)
=> FAILED
            64
                  6225248 (1e+04x)
=> FAILED
           128
                   12450400 (1e+04x)
==> 0/4 tests passed
Estimated student memory = 97268.00 \text{ T} + 96.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
  ******
  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 rando
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                        2 * connected()
                seconds
                           union()
                                              + find()
     constructor
```

```
java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                 8 Infinity
                                      21
                                                             32
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                      68
                                           (0.5x)
                                                             32
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               128 Infinity
                                     263
                                           (0.1x)
                                                            232
(0.1x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               512 Infinity
                                    1029
                                           (0.0x)
                                                           1556
(0.0x)
                1
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
    TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
```

```
=> FAILED
             1024 Infinity
                                  2054 (0.0x)
                                                          3278
(0.0x)
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
and varies each time that you submit. If one of the values in th
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                       per open()
                                     per isOpen()
                                                      per isFull
     per percolates()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
    TimePercolation.main(TimePercolation.java:420)
=> passed
                32
                         0
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TimePercolation.countMaxOperations(TimePercolation.java:50)
    TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
    TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
```

Submission					
	=> passed	128	0	0	0
	Percola TimePer TimePer 86)	tion.isOpe colation.c colation.m	en(Percolati countMaxOper naxOperation	ndsException on.java:46) ations(TimePercola CountTest(TimePerc ePercolation.java: colation.java:420)	colation.java:3
	Percola TimePer TimePer 86) TimePer	tion.isOpe colation.c colation.m	en(Percolati countMaxOper naxOperation	0 ndsException on.java:46) ations(TimePercola CountTest(TimePercolation.java: ePercolation.java:420)	colation.java:3
	=> passed 0 ==> 4/4 test Total: 5/9 t =======	·	0 ∙d! 	0	0

Submission			
Submission time	Sat-26-Oct 00:08:20		
Raw Score	18.89 / 100.00		
Feedback	See the Assessment Guide for information on how to read this report.		
	Assessment Summary		
	Compilation: PASSED		

Style: FAILED

Findbugs: No potential bugs found.

API: PASSED

Correctness: 0/20 tests passed
Memory: 4/8 tests passed
Timing: 5/9 tests passed

Raw score: 18.89% [Correctness: 65%, Memory: 10%, Timing: 25%, St

yle: 0%]

Assessment Details

```
The following files were submitted:
-----
total 12K
-rw-r--r-- 1 1.6K Oct 26 07:08 Percolation.java
-rw-r--r-- 1 2.3K Oct 26 07:08 PercolationStats.java
-rw-r--r-- 1 1.7K Oct 26 07:08 studentSubmission.zip
 *****************
*****
 compiling
% javac Percolation.java
% javac PercolationStats.java
*_____
______
% checkstyle *.java
*_____
Percolation.java:2:1: File contains tab characters (this is the f
irst instance).
```

```
Percolation.java:3:38: Name 'perco_uf' must match pattern '^[a-z]
[a-zA-Z0-9]*$|^[A-Z][A-Z_0-9]*$'.
Percolation.java:4:29: Name 'open_status' must match pattern '^[a
-z][a-zA-Z0-9]*$|^[A-Z][A-Z_0-9]*$'.
Percolation.java:12:21: Name 'bottom_line' must match pattern '^[
a-z][a-zA-Z0-9]*$|^[A-Z][A-Z_0-9]*$'.
Percolation.java:13:20: 'for' is not followed by whitespace.
Percolation.java:21:19: 'if' is not followed by whitespace.
Percolation.java:25:19: 'if' is not followed by whitespace.
Percolation.java:25:38: Expression can be simplified.
Percolation.java:29:19: 'if' is not followed by whitespace.
Percolation.java:32:19: 'if' is not followed by whitespace.
Percolation.java:35:19: 'if' is not followed by whitespace.
Percolation.java:38:19: 'if' is not followed by whitespace.
Percolation.java:43:19: 'if' is not followed by whitespace.
Percolation.java:52:19: 'if' is not followed by whitespace.
PercolationStats.java:6:1: File contains tab characters (this is
the first instance).
PercolationStats.java:13:20: 'for' is not followed by whitespace.
PercolationStats.java:20:20: 'for' is not followed by whitespace.
PercolationStats.java:27:20: 'for' is not followed by whitespace.
PercolationStats.java:41:21: Name 'count_percolate' must match pa
ttern '^[a-z][a-zA-Z0-9]*$|^[A-Z][A-Z_0-9]*$'.
PercolationStats.java:42:19: 'if' is not followed by whitespace.
PercolationStats.java:42:37: '{' is not preceded with whitespace.
PercolationStats.java:56:14: 'while' is not followed by whitespac
PercolationStats.java:73: Use x*x instead of Math.pow(x, 2)
PercolationStats.java:74: Use x*x instead of Math.pow(x, 2)
PercolationStats.java:75: Use x*x instead of Math.pow(x, 2)
% findbugs *.class
______
Testing the APIs of your programs.
Percolation:
PercolationStats:
```

```
______
  **********************
  executing
Testing methods in Percolation
*_____
Running 13 total tests.
Test 1: Check whether exception is called if (i, j) are out of bo
unds
 * N = 10, (i, j) = (0, 6)
    - IndexOutOfBoundsException NOT thrown for open()
    - IndexOutOfBoundsException NOT thrown for isOpen()
    - IndexOutOfBoundsException NOT thrown for isFull()
 * N = 10, (i, j) = (12, 6)
 * N = 10, (i, j) = (11, 6)
 * N = 10, (i, j) = (6, 0)
    - IndexOutOfBoundsException NOT thrown for open()
    - IndexOutOfBoundsException NOT thrown for isOpen()
    - IndexOutOfBoundsException NOT thrown for isFull()
 * N = 10, (i, j) = (6, 12)
    - IndexOutOfBoundsException NOT thrown for isFull()
 * N = 10, (i, j) = (6, 11)
    - IndexOutOfBoundsException NOT thrown for isFull()
==> FAILED
Tests 2 through 8 create a Percolation object using your code, th
en repeatedly
open sites using open(i, j). After each call to open, we check th
at isFull(),
isOpen(), and percolates() return the corrrect results.
Test 2: Open predetermined list of sites using files
 * filename = input6.txt
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
```

```
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test2(TestPercolation.java:196)
TestPercolation.main(TestPercolation.java:540)
```

- * filename = input8.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:197)
 TestPercolation.main(TestPercolation.java:540)
- * filename = input8-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:198)
 TestPercolation.main(TestPercolation.java:540)
- * filename = input10-no.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:199)
 TestPercolation.main(TestPercolation.java:540)
- * filename = greeting57.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test2(TestPercolation.java:200)
 TestPercolation.main(TestPercolation.java:540)
- * filename = heart25.txt
 java.lang.ArrayIndexOutOfBoundsException

```
Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test2(TestPercolation.java:201)
    TestPercolation.main(TestPercolation.java:540)
==> FAILED
Test 3: Open random sites until system percolates (then test is t
erminated)
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:235)
    TestPercolation.main(TestPercolation.java:541)
   N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
     TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:236)
    TestPercolation.main(TestPercolation.java:541)
  * N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.random(TestPercolation.java:214)
    TestPercolation.test3(TestPercolation.java:237)
    TestPercolation.main(TestPercolation.java:541)
   N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
```

```
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:238)
TestPercolation.main(TestPercolation.java:541)
```

* N = 20

java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:239)
TestPercolation.main(TestPercolation.java:541)

* N = 20
java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:240)
TestPercolation.main(TestPercolation.java:541)

* N = 50

java.lang.ArrayIndexOutOfBoundsException

Percolation.isOpen(Percolation.java:46)

TestPercolation.checkIsOpen(TestPercolation.java:43)

TestPercolation.check(TestPercolation.java:75)

TestPercolation.random(TestPercolation.java:214)

TestPercolation.test3(TestPercolation.java:241)

TestPercolation.main(TestPercolation.java:541)

* N = 50

java.lang.ArrayIndexOutOfBoundsException
Percolation.isOpen(Percolation.java:46)
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.random(TestPercolation.java:214)
TestPercolation.test3(TestPercolation.java:242)
TestPercolation.main(TestPercolation.java:541)

==> FAILED

Test 4: Opens predetermined sites, but where N = 1 and N = 2 (cor

```
ner case test)
  * filename = input1.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test4(TestPercolation.java:251)
    TestPercolation.main(TestPercolation.java:542)
   filename = input1-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
     TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:252)
     TestPercolation.main(TestPercolation.java:542)
  * filename = input2.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test4(TestPercolation.java:253)
     TestPercolation.main(TestPercolation.java:542)
   filename = input2-no.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test4(TestPercolation.java:254)
     TestPercolation.main(TestPercolation.java:542)
==> FAILED
Test 5: Check for backwash with predetermined sites
  * filename = input20.txt
     java.lang.ArrayIndexOutOfBoundsException
```

279 of 296 2013年10月28日 21:20

Percolation.isOpen(Percolation.java:46)

```
TestPercolation.checkIsOpen(TestPercolation.java:43)
TestPercolation.check(TestPercolation.java:75)
TestPercolation.file(TestPercolation.java:138)
TestPercolation.test5(TestPercolation.java:263)
TestPercolation.main(TestPercolation.java:543)
```

* filename = input10.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:264)
 TestPercolation.main(TestPercolation.java:543)

* filename = input50.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test5(TestPercolation.java:265)
 TestPercolation.main(TestPercolation.java:543)

==> FAILED

Test 6: Check for backwash with predetermined sites that havemult iple percolating paths

* filename = input3.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)
 TestPercolation.test6(TestPercolation.java:275)
 TestPercolation.main(TestPercolation.java:544)

* filename = input4.txt
 java.lang.ArrayIndexOutOfBoundsException
 Percolation.isOpen(Percolation.java:46)
 TestPercolation.checkIsOpen(TestPercolation.java:43)
 TestPercolation.check(TestPercolation.java:75)
 TestPercolation.file(TestPercolation.java:138)

```
TestPercolation.test6(TestPercolation.java:276)
     TestPercolation.main(TestPercolation.java:544)
  * filename = input7.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test6(TestPercolation.java:277)
     TestPercolation.main(TestPercolation.java:544)
==> FAILED
Test 7: Predetermined sites with very long percolating path
  * filename = snake13.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test7(TestPercolation.java:287)
    TestPercolation.main(TestPercolation.java:545)
  * filename = snake101.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
    TestPercolation.test7(TestPercolation.java:288)
     TestPercolation.main(TestPercolation.java:545)
==> FAILED
Test 8: Opens every site
  * filename = input5.txt
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.file(TestPercolation.java:138)
     TestPercolation.test8(TestPercolation.java:296)
```

```
TestPercolation.main(TestPercolation.java:546)
==> FAILED
Test 9: Create multiple Percolation objects at the same time
        (to make sure you didn't store data in static variables)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:336)
    TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
     TestPercolation.test9(TestPercolation.java:337)
    TestPercolation.main(TestPercolation.java:548)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.check(TestPercolation.java:75)
    TestPercolation.twoPercolations(TestPercolation.java:310)
    TestPercolation.test9(TestPercolation.java:338)
    TestPercolation.main(TestPercolation.java:548)
==> FAILED
Test 10: Open predetermined list of sites using file
         but change the order in which methods are called
  * filename = input8.txt; order = isFull(),
                                                       isOpen(),
percolates()
     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(),
percolates()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:178)
    TestPercolation.test10(TestPercolation.java:385)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = isOpen(), percolates(),
    isFull()
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:179)
    TestPercolation.test10(TestPercolation.java:386)
    TestPercolation.main(TestPercolation.java:549)
  * filename = input8.txt; order = percolates(),
                                                       isOpen(),
    isFull()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.file(TestPercolation.java:180)
    TestPercolation.test10(TestPercolation.java:387)
    TestPercolation.main(TestPercolation.java:549)
  * 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 11: Call all methods in random order until just before syste
m percolates
 * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
```

```
TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:398)
     TestPercolation.main(TestPercolation.java:550)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation} ({\tt TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:399)
     TestPercolation.main(TestPercolation.java:550)
     N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:400)
     TestPercolation.main(TestPercolation.java:550)
     N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     Test Percolation. random Calls Until Percolation (Test Percolation.\\
java:363)
     TestPercolation.test11(TestPercolation.java:401)
     TestPercolation.main(TestPercolation.java:550)
    N = 20
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation}. \\ {\tt TestPercolation.randomCallsUntilPercolation}.
java:363)
     TestPercolation.test11(TestPercolation.java:402)
     TestPercolation.main(TestPercolation.java:550)
     N = 50
```

```
java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     {\tt TestPercolation.randomCallsUntilPercolation(TestPercolation.}
java:363)
     TestPercolation.test11(TestPercolation.java:403)
     TestPercolation.main(TestPercolation.java:550)
==> FAILED
Test 12: Call all methods in random order with inputs not prone t
o backwash
  * N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:458)
     TestPercolation.main(TestPercolation.java:551)
    N = 5
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
     TestPercolation.test12(TestPercolation.java:459)
     TestPercolation.main(TestPercolation.java:551)
  * N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
     TestPercolation.test12(TestPercolation.java:460)
     TestPercolation.main(TestPercolation.java:551)
   N = 10
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.open(Percolation.java:22)
     TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
25)
```

```
TestPercolation.test12(TestPercolation.java:461)
    TestPercolation.main(TestPercolation.java:551)
  * N = 20
     isFull(1, 2) returns wrong value [after 3 total calls to ope
n()]
     - student = false
     - reference = true
  * N = 50
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCallsNoBackwash(TestPercolation.java:4
31)
    TestPercolation.test12(TestPercolation.java:463)
     TestPercolation.main(TestPercolation.java:551)
==> FAILED
Test 13: Call all methods in random order until all sites are ope
    N = 3
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
     TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:517)
    TestPercolation.main(TestPercolation.java:552)
  * N = 5
     java.lang.ArrayIndexOutOfBoundsException
    Percolation.open(Percolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:484)
    TestPercolation.test13(TestPercolation.java:518)
    TestPercolation.main(TestPercolation.java:552)
    N = 7
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:519)
     TestPercolation.main(TestPercolation.java:552)
```

```
N = 10
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.open(Percolation.java:22)
    TestPercolation.randomCalls(TestPercolation.java:484)
    TestPercolation.test13(TestPercolation.java:520)
    TestPercolation.main(TestPercolation.java:552)
  * N = 20
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:521)
    TestPercolation.main(TestPercolation.java:552)
  * N = 50
    java.lang.ArrayIndexOutOfBoundsException
    Percolation.isOpen(Percolation.java:46)
    TestPercolation.checkIsOpen(TestPercolation.java:43)
    TestPercolation.randomCalls(TestPercolation.java:490)
    TestPercolation.test13(TestPercolation.java:522)
    TestPercolation.main(TestPercolation.java:552)
==> FAILED
Total: 0/13 tests passed!
______
Testing methods in PercolationStats
*-----
Running 7 total tests.
Test 1a-1b: Test mean and standard deviation of percolation thres
hold
Creating new PercolationStats(100, 50)
PercolationStats reports:
       mean(): 0.000 (FAILED, outside of range)
       stddev(): 0.000 (FAILED, outside of range)
```

```
Overall result: FAILED
Creating new PercolationStats(200, 10)
PercolationStats reports:
       mean(): 0.000 (FAILED, outside of range)
       stddev(): 0.000 (FAILED, outside of range)
       Overall result: FAILED
Test 1c-d: Test confidence interval of PercolationStats
Creating new PercolationStats(100, 50)
-----
 * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Creating new PercolationStats(200, 10)
-----
 * confidenceLo() = 0.0
 * confidenceHi() = 0.0
==> FAILED
Test 2: Check whether exception is called if N, T are out of boun
ds
 * N = -23, T = 42
    - IllegalArgumentException NOT thrown for PercolationStats()
 * N = 23, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
  * N = -42, T = 0
    - IllegalArgumentException NOT thrown for PercolationStats()
==> FAILED
Test 3: Create multiple PercolationStats objects at the same time
(to make sure you didn't store data in static variables)
  * 1mean = 0.0
 * 2mean = 0.0
  * 1mean = 0.0
  * 2mean = 0.0
```

```
* 1mean = 0.0
      * 2mean = 0.0
==> FAILED
Test 4: Call the methods of PercolationStats in either order.
      * order = mean(), stddev()
      * mean = 0.0; stddev = 0.0
      * order = stddev(), mean()
      * mean = 0.0; stddev = 0.0
==> FAILED
Total: 0/7 tests passed!
______
 ******************
 *****
     memory usage
 ******************
Computing memory of Percolation
*_____
Running 4 total tests.
Test 1a-1d: Measuring total memory usage as a function of grid si
ze (max allowed: 17 N^2 + 128 N + 1024 bytes)
                                                   N bytes
 -----
=> passed 64 41864
=> passed 256 609032
=> passed
                                           512
                                                                        2397448
-> passed 1024 9513224
==> 4/4 tests passed
Estimated student memory = 9.00 \, \text{N}^2 + 74.00 \, \text{N} + 264.00 \, (\text{R}^2 = 1.00 \, \text{N} + 1.00 \, 
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
allowed: 8 T + 128 bytes)
            Т
                   bytes
-----
=> FAILED 16 1556384 (6e+03x)
=> FAILED
           32
                3112672 (8e+03x)
=> FAILED 64 6225248 (1e+04x)
=> FAILED 128 12450400 (1e+04x)
==> 0/4 tests passed
Estimated student memory = 97268.00 \text{ T} + 96.00 \text{ (R}^2 = 1.000)
Total: 0/4 tests passed!
______
*****
  timing
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 rando
```

```
mly opened
until the system percolates. If you do not pass the correctness t
ests, these
results may be meaningless.
                                                  2 * connected()
                     seconds
                                                        + find()
                                   union()
       constructor
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> passed
                 8 Infinity
                                      21
                                                             32
                1
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
    TimePercolation.main(TimePercolation.java:420)
=> FAILED
                32 Infinity
                                                             32
                                      68
                                           (0.5x)
(0.2x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
    TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
               128 Infinity
                                     263
                                           (0.1x)
                                                            232
(0.1x)
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
    TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
```

```
=> FAILED
               512 Infinity
                                    1029
                                           (0.0x)
                                                          1556
(0.0x)
                1
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.run(TimePercolation.java:16)
     TimePercolation.operationCountTest(TimePercolation.java:326)
     TimePercolation.testLite(TimePercolation.java:404)
     TimePercolation.main(TimePercolation.java:420)
=> FAILED
              1024 Infinity
                                    2054 (0.0x)
                                                          3278
(0.0x)
==> 1/5 tests passed
Running time in seconds depends on the machine on which the scrip
t runs,
and varies each time that you submit. If one of the values in th
e table
violates the performance limits, the factor by which you failed t
he 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 o
f calls to
union(), connected(), and find() per call to open(), isFull(), an
d percolates().
The table below shows max(union(), connected(), find()) calls mad
e during a
single call to open(), isFull(), and percolates().
                 Ν
                       per open()
                                       per isOpen()
                                                       per isFull
      per percolates()
     java.lang.ArrayIndexOutOfBoundsException
     Percolation.isOpen(Percolation.java:46)
     TimePercolation.countMaxOperations(TimePercolation.java:50)
     TimePercolation.maxOperationCountTest(TimePercolation.java:3
86)
     TimePercolation.testLite(TimePercolation.java:405)
     TimePercolation.main(TimePercolation.java:420)
```

Submission => passed 32 0 java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TimePercolation.countMaxOperations(TimePercolation.java:50) TimePercolation.maxOperationCountTest(TimePercolation.java:3 86) TimePercolation.testLite(TimePercolation.java:405) TimePercolation.main(TimePercolation.java:420) => passed 128 0 0 java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TimePercolation.countMaxOperations(TimePercolation.java:50) TimePercolation.maxOperationCountTest(TimePercolation.java:3 86) TimePercolation.testLite(TimePercolation.java:405) TimePercolation.main(TimePercolation.java:420) => passed 0 0 512 0 java.lang.ArrayIndexOutOfBoundsException Percolation.isOpen(Percolation.java:46) TimePercolation.countMaxOperations(TimePercolation.java:50) TimePercolation.maxOperationCountTest(TimePercolation.java:3 86) TimePercolation.testLite(TimePercolation.java:405) TimePercolation.main(TimePercolation.java:420) => passed 1024 0 0 ==> 4/4 tests passed Total: 5/9 tests passed!

Submission

Submission			
Submission time	Fri-25-Oct 23:59:04		
Raw Score	0.00 / 100.00		
Feedback Compilation: PASSED			
	API: FAILED		
PercolationStats:			
	The following fields should be made private: * public Percolation[] per		
	* public int[] counter		

Submission			
Submission time	Fri-25-Oct 23:59:01		
Raw Score	0.00 / 100.00		
Feedback	Compilation: PASSED		
API: FAILED			
	PercolationStats:		
	The following fields should be made private: * public Percolation[] per * public int[] counter		

Submission		
Submission time	Fri-25-Oct 20:29:27	
Raw Score	0.00 / 100.00	

Submission	
Feedback	Error extracting files! Zip file invalid.

https://class.coursera.org/algs4partI-003/assignm...