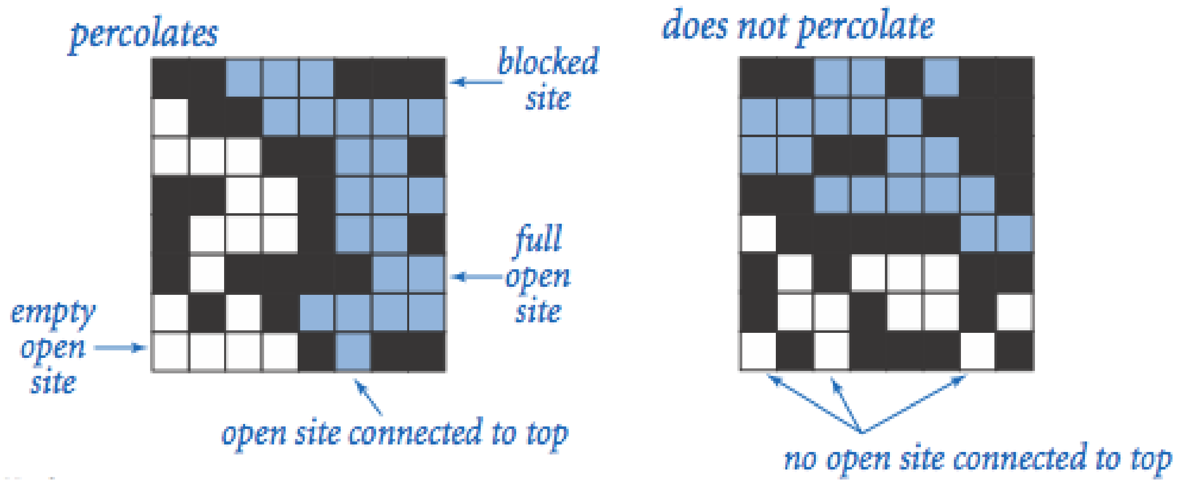


Figure 1:

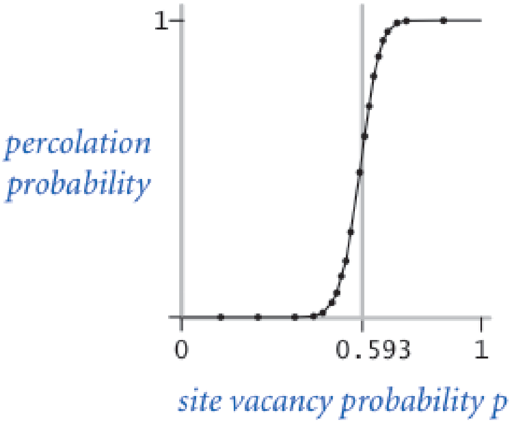
Project-1 Due Date: 12th June, 2019, 11:59 pm

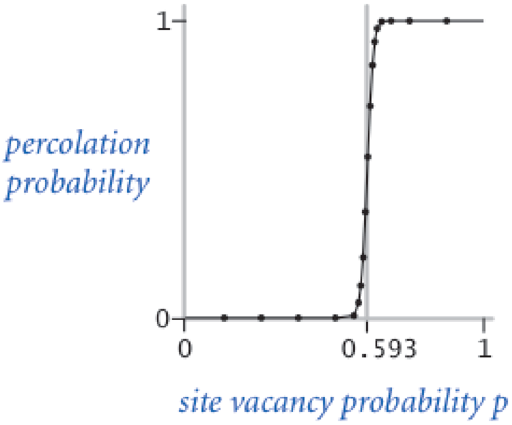
Percolation:

Geologically, percolation is the slow movement of water through the pores in soil or permeable rock.

The Problem:

In a famous scientific problem, researchers are interested in the following question: if sites are independently set to be open with probability p (and therefore blocked with probability $(1-p)$), what is the probability that the system percolates? When p equals 0, the system does not percolate; when p equals 1, the system percolates. The plots below show the site vacancy probability p versus the percolation probability for 20-by-20 random grid (left) and 100-by-100 random grid (right).





When N is sufficiently large, there is a threshold value $p^\star$ such that when $p < p^\star$ a random N-by-N grid almost never percolates, and when $p > p^\star$, a random N-by-N grid almost always percolates. No mathematical solution for determining the percolation threshold $p^\star$ has yet been derived. Your task is to write a computer program to estimate $p^\star$.

We use an N-by-N grid to model such a system. Each of the sites in the grid can be any of the three integers, 0,1,2 or 3. We call an N-by-N system percolating if there is a sequence of the same integer, connecting the top row to the bottom.

Question-1

(*Model a Percolation System*) To model a percolation system, create a data type *New_Percolation* in *New_Percolation.java* with the following API:

method	description
$Percolation(intN)$	create an N-by-N grid, with all sites blocked
void open(int i, int j)	open site (i, j)
boolean isOpen(int i, int j)	is site (i, j) open?
boolean isFull(int i, int j)	is site (i, j) full?
int numberOfOpenSites()	number of open sites
boolean percolates()	does the system percolate?

Corner cases: By convention, the row and column indices i and j are integers between 0 and $N-1$, where $(0,0)$ is the upper-left site. Throw a *java.lang.IndexOutOfBoundsException* if any argument to *open()*, *isOpen()*, or *isFull()* is outside its prescribed range. The constructor should throw a *java.lang.IllegalArgumentException* if $N \leq 0$.

Performance requirements. The constructor should take time proportional to N^2; all methods should take constant time plus a constant number of calls to the union-find methods *union()*, *find()*, *connected()*, and *count()*.

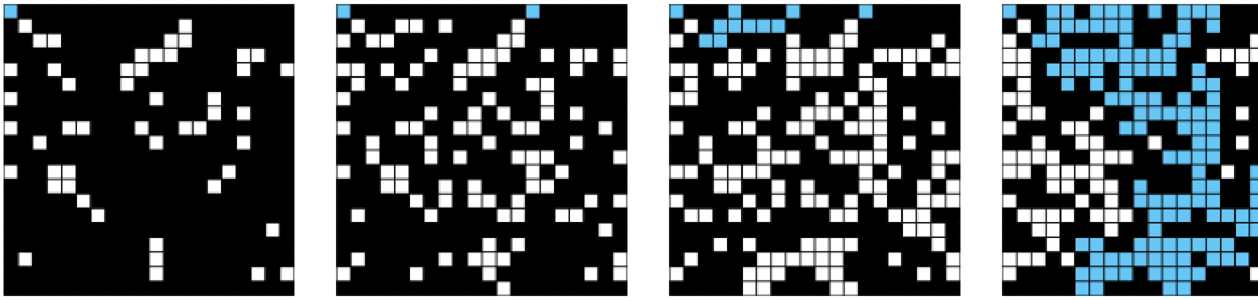
```
$ java Percolation data/input10.txt
56 open sites
percolates
$ java Percolation data/input10-no.txt
55 open sites
does not percolate
```

Question-2

Estimate Percolation Threshold: To estimate the percolation threshold, consider the following computational (Monte Carlo simulation) experiment:

- Initialize all sites to be blocked.
- Repeat the following until the system percolates:
  - Choose a site (row *i*, column *j*) uniformly at random among all blocked sites.
  - Open the site (row *i*, column *j*).
- The fraction of sites that are opened when the system percolates provides an estimate of the percolation threshold.

For example, if sites are opened in a 20-by-20 grid according to the snapshots below, then our estimate of the percolation threshold is 204/400 = 0.51 because the system percolates when the 204th site is opened.



50 open sites *100 open sites* *150 open sites* *204 open sites*

By repeating this computational experiment T times and averaging the results, we obtain a more accurate estimate of the percolation threshold. Let x_t be the fraction of open sites in computational experiment t. The sample mean μ provides an estimate of the percolation threshold, and the sample standard deviation σ measures the sharpness of the threshold:

$$\mu = \frac{x_1 + x_2 + \cdots + x_T}{T}, \quad \sigma^2 = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_T - \mu)^2}{T - 1}.$$

Assuming T is sufficiently large (say, at least 30), the following provides a 95% confidence interval for the percolation threshold:

$$\left[\mu - \frac{1.96\sigma}{\sqrt{T}}, \mu + \frac{1.96\sigma}{\sqrt{T}}\right].$$

To perform a series of computational experiments, create a data type *PercolationStats* in *PercolationStats.java* with the following API:

method	description
$PercolationStats(intN, intT)$	perform T independent experiments on an N-by-N grid
double mean()	sample mean of percolation threshold
double stddev()	sample standard deviation of percolation threshold
double confidenceLow()	low endpoint of 95% confidence interval
double confidenceHigh()	high endpoint of 95% confidence interval

The constructor should take two arguments N and T, and perform T independent computational experiments (discussed above) on an N-by-N grid. Using this experimental data, it should calculate the mean, standard deviation, and the 95% confidence interval for the percolation threshold.

Corner cases. The constructor should throw a *java.lang.IllegalArgumentException* if either $N \leq 0$ or $T \leq 0$.

```
$ java PercolationStats 100 1000
mean           = 0.592804
stddev         = 0.015764
confidenceLow  = 0.591827
confidenceHigh = 0.593781
```

Files to Submit

- New_Percolation.java
- New_PercolationStats.java
- Report.txt
  - Organize your thoughts. Describe your approach to the problem. What difficulties did you face and how did you resolve them.
  - What is the backwash problem? Did you encounter it? What was your work around?
  - Cite all your references.