CS200 - Programming Assignment 2: Percolation

Motivation
Percolation refers to the movement of a fluid through a porous material, but there are many systems that are mathematically similar, and percolation theory is the study of the general principles behind percolation. The following are examples of percolation: water flowing through ground coffee beans to make coffee, a crack developing in wood or stone, decay in a tooth, disease spreading through a population. We are generally interested in the probability that fluid will easily flow through the system. There is often a variable (a parameter) than controls how flow happens in the system, and it is important to determine the *critical threshold*, which is the value of the parameter at which point it is easy for flow to occur.

Guidelines
Please read and abide by the honor code guidelines in the syllabus.

Please read the rubric so you know how you will be graded. For example, turning in a program that compiles and runs without errors but does nothing will earn you more points than a program that is close to working but does not compile or contains errors on running.

There are two options for this assignment: Standard and Challenge. (You must turn in only one of the two options.) You will be graded using this rubric. Since the rubric is out of 30 points, if you earn X points, then your grade will be $X/30 \times .85$ for a Standard assignment and $X/30 \times .95$ for Challenge. Thus a poorly done Challenge program could give you a worse score than a well executed Standard program.

You must write your own graph search subroutine, but you can use built-in functions or external packages to implement your queue (if you do Breadth-First-Search). You should create your graphs from scratch. (For example, you can represent an adjacency matrix as a list of lists or as an array of arrays.) Otherwise you may use any functions, methods, or packages that you find useful, and you may use code snippets that you find on, for example, Stack Overflow (as long as you properly cite your reference).

Put a multi-line comment at the beginning of your program. It should contain:

- Your name
- "Programming Assignment 1"
- "Challenge" or "Standard"
- The name of anyone you worked with and the nature of your collaboration
- Output from your program
- The amount of time (approximately) that you spent on this assignment

Standard Assignment
Consider the sample space of all possible graphs on 5 vertices (only considering graphs where between any two vertices, there is either no edge or one edge), and assume each possible graph occurs with the same probability. We will say there is flow in a graph whenever there is a path between all pairs vertices in the graph (we say the graph is connected). In other words, there is flow when, starting at vertex 1, you can run a search algorithm, and find all other vertices in the graph. Let E be the event that includes all elements of the sample space that have flow. Write a program in java or python that determines the probability of E. That is, your program should determine the probability that you end up with a graph that is connected.

Challenge Assignment
Consider a graph on n^2 vertices where the vertices are on the intersection points of a 2-D square grid, and there are edges between two vertices that are adjacent on the grid (as in the graph below). Now we consider creating a new graph where, for each edge in the original grid graph, we actually put an edge there with probability p. We say there is flow in this graph if there is a path from the top left vertex to the bottom right vertex. Let E be the event that includes all graphs with flow. The percolation threshold $p^*(n)$ is the value of p where the probability of E is $1/2$. Write a program in python or java that estimates the percolation threshold for values of n from 3 to k. (k should be at least 4, but could be larger if your code runs fast enough.) To do this, you should first write a program that, given n and p as input, determines the probability of E. Next, for fixed n, you should perform a binary search over values of p to get an estimate of $p^*(n)$ to accuracy $\pm.01$. (Why can you do a binary search?) Then repeat for each value of n from 3 to k.

Your program should output a list of values of n and corresponding estimates for $p^*(n)$. In the preamble of your program please also comment on whether your output makes sense.