CSC 403 W18 Homework 7 part B

In search of .... The Giant Component

Objectives:

- Practice using the Graph/Digraph graph classes.
- Practice using the connected-component classes and the author's design pattern
- Practice developing code to perform a computational experiment
- Practice analyzing data

In this assignment, you will investigate the emergence of the so-called Giant Component in random graphs. The term Giant Component is somewhat loosely defined – the basic notion is: the component that contains a significant portion of the vertices (or the one that is significantly larger than all the other components).

You will use the Erdos-Renyi (N,p) random graph model. In this model, we choose a graph/digraph size N, and a value p in [0,1]. Here, p represents the probability of the existence of an edge between every pair of vertices v,w in the graph/digraph. Functions are provided to generate the ER graphs/digraphs for you.

You are to write a program(s) to experimentally investigate the relationship between the value p and the size of:

- the largest component in an ER undirected graph
- largest strongly connected component in an ER digraph.

Undirected graphs: suggested procedure

1. The algs41/CC.java program computes the connected components of a graph. Modify this class to include a function which returns the largest component size.
2. Copy/paste the erRandom function into your source file (or copy/paste into the algs41/GraphGenerator class).
3. Modify the main program to create a series of erRandom Graphs of size N=100, with varying values for p; computing the largest component size for each one.
4. Repeat step 3 'M' times to attempt to get 'average' component sizes for each value of p that are relatively stable.
5. Analyze the data. (maybe plot p vs component size) (maybe change N to 200, see if you see a pattern?)

Directed graphs: suggested procedure

1. The algs42/KosarajuSharirSCC.java program computes the strongly connected components of a graph. Modify this class to include a function which returns the largest component size.
2. Copy/paste the erRandom function (Digraph version) into your source file (or copy/paste into the algs42/DigraphGenerator class).
3. Modify the main program to create a series of erRandom Digraphs of size N=100, with varying values for p; computing the largest component size for each one.
4. Repeat step 3 'M' times to attempt to get 'average' component sizes for each value of p that are relatively stable.
5. Analyze the data. (maybe plot p vs component size) (maybe change N to 200, see if you see a pattern?)

Vers 1.0

Hint: The only interesting values of p will turn out to be relatively small. You might investigate this by trial and error. See if you can find an interval (p1, p2) where interesting things happen.

Create a document which contains an overview of your experiment, observation & results.
Include plots if you made any
Be sure you are addressing the experimental question (in red, first page).

1-2 pages should be plenty (including any plots).

Turn in:

1 Document
2 Source files.

Please make sure the code you added is clearly delineated. E.g. add a bunch of blank lines and/or comment blocks

/* this is my code */

/* my code ends here */

```
// Erdos-Renyi  (N,p) random graph
     public static Graph erRandom(int V, double p) {
          if (V < 0 || p < 0) throw new IllegalArgumentException ();
          Graph G = new Graph (V);
          for ( int v = 0; v < V; v++)
               for (int w = v+1; w < V; w++)
               if ( StdRandom.uniform() <= p)
                    G.addEdge(v, w);
          return G;
     }


 // Erdos-Renyi  (N,p) random digraph
     public static Digraph erRandom(int V, double p) {
          if (V < 0 || p < 0) throw new IllegalArgumentException ();
          Digraph D = new Digraph (V);
          for ( int v = 0; v < V; v++)
               for (int w = 0; w < V; w++)
                    if ( v != w )
                         if ( StdRandom.uniform() <= p)
                              D.addEdge(v, w);
          return D;
     }
```