Tyler Paulsen
csci-351
Assignment 1

1. Describe, at a high level but completely, what your simulation program(s)' inputs are, what the program(s) do, and what the outputs are.

The simulation program was build to generate all the data needed for the assignment with one command. The user only needs to choose the degree of accuracy he/she wants.

The three inputs are:
1. Seed: Used in the random number generator.
2. Number of Vertices: The number of vertices to sweep for both tests.
3. Number of Trials: How many times to perform each simulation before the values are accumulated and averaged.

With the given inputs, two Monte Carlo simulations are performed. The first simulation creates the table for Probability Vs. Average Diameter. The simulation takes the number of vertices and splits it up into 5 groups then performs the simulation on each set of vertices. The second simulation uses the probabilities 0.20, 0.40, 0.60, 0.80, and 1.00 for each group in simulation for Vertices Vs. Average Diameter.

The output for each group of the simulation is three columns with the values for:
- Number of Vertices
- Edge Probability
- Average Diameter.

The data was then copied into excel to create the graphs.

The Monte Carlo simulation is done in parallel. Each thread first creates a graph based on the edge probability by taking all the possible edge combinations and flipping a coin with the probability given. Next every path in the graph is calculated, and the longest path is kept. The larges path is then reduced until the number of trials is reach. The simulation then returns the average diameter.

2. State the exact command line(s) for running the program(s), including the Java main program class name and a description of each command line argument.

java pj2 [threads=<K>] AverageDiameterRandomGraphSmp <seed> <V> <T>
- <K> - Number of parallel threads
- <seed> - Random seed
- <V> - Number of vertices
- <T> - Number of trials

Example (Command takes ~2 min to execute)
java pj2 AverageDiameterRandomGraphSmp 243095 100 500

3. Put in the report the complete source code of your simulation program(s).

```
import edu.rit.pj2.LongLoop;
import edu.rit.pj2.Task;
import edu.rit.pj2.vbl.DoubleVbl;
import edu.rit.util.Random;
import java.util.*;

/**
* Class AverageDiameterRandomGraphSmp is a multicore parallel program that measures
* the average diameter distribution of a series of random graph.
* <P>
* Usage: <TT>java pj2 [threads=<I>K</I>] DegreeDistRandomGraphSmp <I>seed</I>
* <I>V</I> <I>T</I> </TT>
* <BR><TT><I>K</I></TT> = Number of parallel threads
* <BR><TT><I>seed</I></TT> = Random seed
* <BR><TT><I>V</I></TT> = Number of vertices
* <BR><TT><I>T</I></TT> = Number of trials
*
* @author Tyler Paulsen
*          -- Templated from Alan Kaminsky java file:
*                  DegreeDistRandomGraphSmp
*                  (https://cs.rit.edu/~ark/351/monte/java2html.php?file=2)
*
*
* @version 20-Feb-2016
*/
public class AverageDiameterRandomGraphSmp extends Task {
   // Command line arguments.
   long seed;
   int V,V_total;
   double P_inc,P;
   long T;
   DoubleVbl avg_diameter;

   // Main program.
   public void main(String[] args) throws Exception {
       // Parse command line arguments, print provenance.
       if (args.length != 3) usage();
       seed = Long.parseLong(args[0]);
       V = Integer.parseInt(args[1]);
       T = Long.parseLong(args[2]);
       V_total = V;
       P_inc = 0.05;
       int V_inc = V_total/5;
       System.out.printf ("$ java pj2 AverageDiameterRandomGraphSmp");
       for (String arg : args) System.out.printf (" %s", arg);
       System.out.println();
```

```
       // set up the first simulation Probability Vs. Average Diameter. Sweeps the
vertices around the main simulation
       for( V = V_inc; V <= V_total; V+= V_inc) {
           System.out.println("# Vert\tProb.\tAvg. Diam.");
           //cheating the precision of floats.
           for (P = P_inc; P <= 1.01; P += P_inc){
               seed += 1;
               monte_carlo_simulation();
           }

       }
       // set up the second simulation Vertices Vs. Average Diameter. Sweeps the
Probability around the main simulation
       P_inc = .2;
       for( P = P_inc; P <= 1.0; P+= P_inc) {
           System.out.println("# Vert\tProb.\tAvg. Diam.");
           for (V = 2; V <= V_total; V += 5) {
               monte_carlo_simulation();
           }
           System.out.println();
       }
   }


   /**
    * monte carlo simulation for the average diameter.
    */
   void monte_carlo_simulation(){

       avg_diameter = new DoubleVbl.Sum(0);
       parallelFor(0, T - 1).exec(new LongLoop() {
           // Per-thread variables.
           Random prng;
           HashMap<Integer, HashSet<Integer>> graph;
           DoubleVbl thr_avg_diameter;

           public void start() {
               prng = new Random(seed + rank());
               graph = new HashMap();
               thr_avg_diameter = threadLocal(avg_diameter);
           }

           public void run(long t) {

               for (int i = 0; i < V; ++i)
                   graph.put(i, new HashSet());

               for (int a = 0; a < V - 1; ++a) {
                   for (int b = a + 1; b < V; ++b) {
```

```
                if (prng.nextDouble() < P) {
                    graph.get(a).add(b);
                    graph.get(b).add(a);
                }
            }
        }
        int hops, max_hops = 0;
        for (int a = 0; a < V; ++a) {
            for (int b = 0; b < V; ++b) {
                if (a != b && (hops = BFS(graph, a, b)) != 0) {
                    if (max_hops < hops)
                        max_hops = hops;
                }
            }
        }
        thr_avg_diameter.item += max_hops;
    }

    /**
     * Breadth first search for a given graph.
     * @param graph - graph to perform the search on.
     * @param start - What node to start from
     * @param dest - What node we are looking for.
     * @return
     */
    int BFS(HashMap<Integer, HashSet<Integer>> graph, int start, int dest) {
        HashSet<Integer> seen = new HashSet();
        LinkedList<Path> queue = new LinkedList();
        Path A = new Path(start);
        queue.addLast(A);
        seen.add(A.id);
        while (!queue.isEmpty()) {
            A = queue.poll();
            if (graph.get(A.id).contains(dest)) {
                return A.dist + 1;
            }
            for (Integer B : graph.get(A.id)) {
                if (!seen.contains(B)) {
                    seen.add(B);
                    queue.addLast(new Path(B, A));
                }
            }
        }
        return 0;
    }


    // Class used in the breadth first search.
    class Path {
        int dist, id;
```

```
                Path parent;

                //constructor, keeps tract of the distance by adding one to the
previous.
                Path(int _id, Path _parent) {
                    id = _id;
                    parent = _parent;
                    dist = parent.dist + 1;
                }
                //initial constructor
                Path(int _id) {
                    id = _id;
                    dist = 0;
                }
            }
        });
        System.out.printf("%d\t%f\t%f\n", V, P, avg_diameter.item/T);
    }
    // Print a usage message and exit.
    private static void usage()
    {
        System.err.println ("Usage: java pj2 [threads=<K>]
AverageDiameterRandomGraphSmp <seed> <V> <T>");
        System.err.println ("<K> = Number of parallel threads");
        System.err.println("<seed> = Random seed");
        System.err.println ("<V> = Number of vertices");
        System.err.println("<T> = Number of trials");
        throw new IllegalArgumentException();
    }
}
```

4. Based on data generated by running your simulation program(s), discuss this question: For a given number of vertices *V*, what happens to the average diameter as the edge probability *p* increases, and why is this happening?

For each of the 5 simulations, they all had the same type of distribution. A low probability for all vertices exhibited a high average diameter. It is interesting to point out that the lowest number of vertices did not have the highest average diameter. The high average diameter is due having a limited number of edges, so if a path was available, there was more of a chance to take a large path to get there. As the probability goes up, the average diameter two until the probability equals 1, then the average diameter is 1. With a small number of vertices, the average diameter takes longer to reach two because there is more of a chance to get a path between two vertices that is long.

Tyler Paulsen
csci-351
Assignment 1

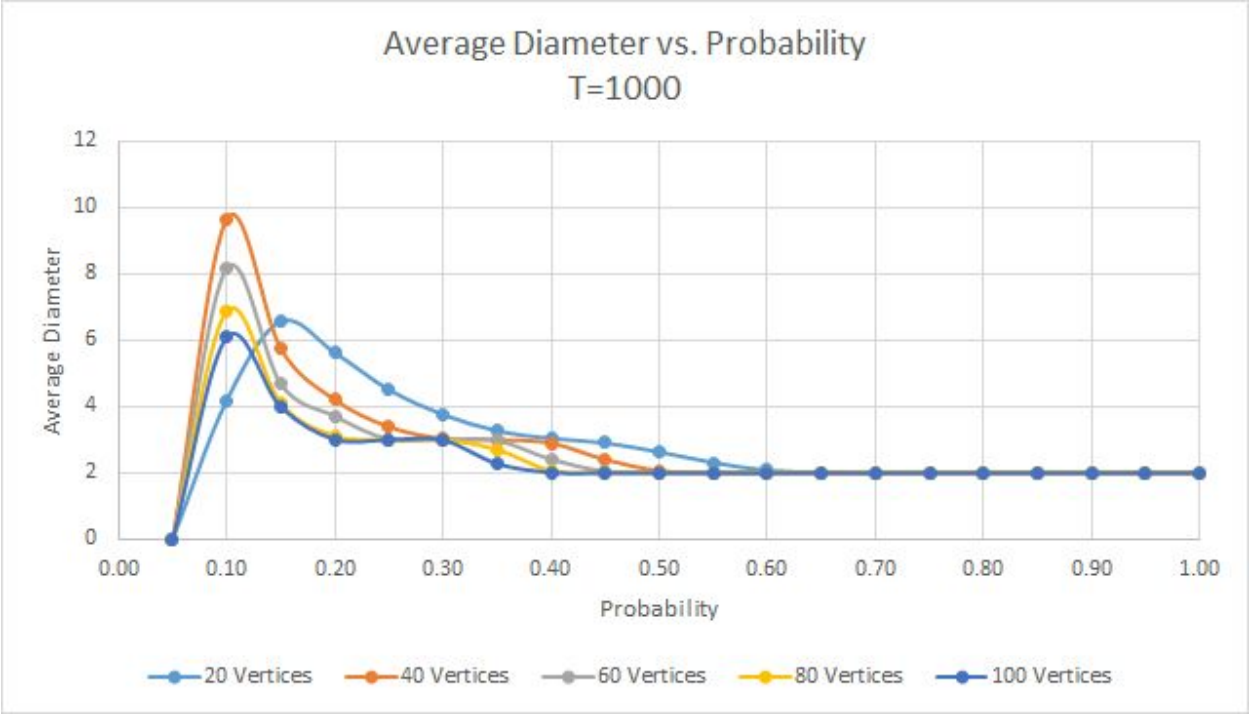
5. Include in your report the data that you analyzed to answer the preceding question. The data must be presented ***both*** in a table or tables ***and*** in a plot or plots. ***Also include*** the exact program command line(s) that generated the data.

Command Executed (The below command takes ~2 min):
java pj2 AverageDiameterRandomGraphSmp 243095 100 500

The above command generated text to fill in the tables below.

# Vert	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Prob.	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
Avg. Diam.	6.08	3.99	3.01	3.00	2.99	2.28	2.01	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00
# Vert	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
Prob.	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
Avg. Diam.	6.88	4.09	3.11	3.00	3.00	2.70	2.07	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00
# Vert	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
Prob.	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
Avg. Diam.	9.64	5.74	4.20	3.39	3.03	3.00	2.90	2.40	2.07	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00
# Vert	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
Prob.	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
Avg. Diam.	4.19	6.57	5.62	4.51	3.76	3.27	3.04	2.90	2.63	2.31	2.08	2.01	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00
# Vert	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
Prob.	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
Avg. Diam.	8.17	4.68	3.71	3.02	3.00	2.98	2.41	2.05	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00

Tyler Paulsen
csci-351
Assignment 1



6. Based on data generated by running your simulation program(s), discuss this question: For a given edge probability *p*, what happens to the average diameter as the number of vertices *V* increases, and why is this happening?

The average diameter eventually converges on two as the probability goes up. The convergence on two happens at different intervals depending on the number of vertices. The bigger the number of vertices, the less it takes to converge onto two, and it takes longer for the smaller number of vertices. With a higher number of vertices the number of nodes with paths between them shrinks, lowering the average diameter. With a smaller number of nodes, it is more likely to have a path between two nodes and increasing the average diameter.

7. Include in your report the data that you analyzed to answer the preceding question. The data must be presented ***both*** in a table or tables ***and*** in a plot or plots. ***Also include*** the exact program command line(s) that generated the data.

Command Executed (The below command takes ~2 min):
java pj2 AverageDiameterRandomGraphSmp 243095 100 500

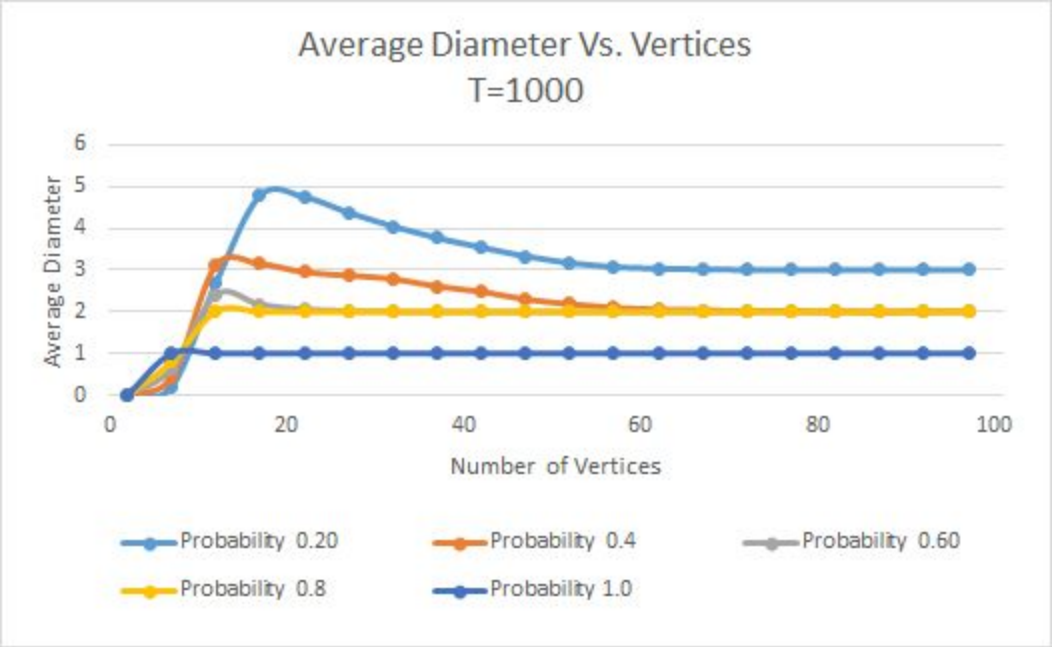
The above command generated text to fill in the tables below.

# Vert	2	7	12	17	22	27	32	37	42	47	52	57	62	67	72	77	82	87	92	97

Tyler Paulsen
csci-351
Assignment 1

Prob.	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Avg. Diam.	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
# Vert	2	7	12	17	22	27	32	37	42	47	52	57	62	67	72	77	82	87	92	97
Prob.	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80
Avg. Diam.	0.79	2.01	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
# Vert	2	7	12	17	22	27	32	37	42	47	52	57	62	67	72	77	82	87	92	97
Prob.	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
Avg. Diam.	0.63	2.41	2.17	2.06	2.02	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
# Vert	2	7	12	17	22	27	32	37	42	47	52	57	62	67	72	77	82	87	92	97
Prob.	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40
Avg. Diam.	0.42	3.11	3.14	2.95	2.87	2.79	2.61	2.48	2.30	2.18	2.10	2.05	2.04	2.01	2.02	2.00	2.00	2.00	2.00	2.00
# Vert	2	7	12	17	22	27	32	37	42	47	52	57	62	67	72	77	82	87	92	97
Prob.	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
Avg. Diam.	0.19	2.67	4.78	4.75	4.37	4.04	3.77	3.54	3.33	3.16	3.07	3.03	3.01	3.00	3.00	3.00	3.00	3.00	3.00	3.00

Tyler Paulsen
csci-351
Assignment 1



8. Write a paragraph describing what you learned from this project.

When first thinking about the project, I thought the Average diameter would reach some number in between one and two, but after the tests were completed, I found out that number was two. It is interesting that if you increase the number of vertices or raise the edge probability, they still have the same trends -- they both eventually hit that average diameter of 2. Knowing this information, someone could create a graph with a particular combination of edge probability and vertices to get the desired outcome. Someone creating a graph can also take into consideration that even if you keep increasing the probability, the diameter still can remain the same, this can help people know when a particular number of vertices should stop adding to the edge probability. The programming part of the assignment helped my understanding on the different data structures in java and what their purpose was.