PRINCETON UNIVERSITY

COS 226–Algorithms and Data Structures

Week 1: *Logistics, WeightedUnionFind, Doubling hypothesis and Percolation problem (Algorithms §1.4 and 1.5)*

Version: February 9, 2018

Exercise 1 – Understanding COS 226 Course Components

A. Class meetings meet twice per week, from 11 to 12:20pm on Mondays and Wednesdays in Friend 101.

B. Precepts meet once per week and cover details pertinent to programming assignments, quizzes, and exams.

C. Quizzes are due Friday at 11PM with a 59 minute grace period.

D. Assignments are due Monday 11PM with a one hour grace period. Files submitted at 12:01am are considered a day late.

E. Lectures are available on the lecture web page.

Exercise 2 – WeightedQuickUnionUF. Algorithms Textbook 1.5

Consider the following code that uses WeightedUnionFind objects (`WeightedQuickUnionUF`). Describe the purpose of this program.

```
/*************************************************************************
*  Name:    Kevin Wayne
*  NetID:   wayne
*  Precept: P99
*
*  Description: This program demonstrates the use of various classes in
*  algs4.jar (WeightedQuickUnionUF, StdRandom, Stopwatch, and StdOut).
*************************************************************************/
import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdStats;
import edu.princeton.cs.algs4.Stopwatch;
import edu.princeton.cs.algs4.StdRandom;
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class ErdosRenyi {

  public static int count(int n) {
      int edges = 0;
      WeightedQuickUnionUF uf = new WeightedQuickUnionUF(n);
      while (uf.count() > 1) {
            int i = StdRandom.uniform(n);
            int j = StdRandom.uniform(n);
            uf.union(i, j);
            edges++;
      }
      return edges;
  }

  public static void main(String[] args) {
      int n = Integer.parseInt(args[0]);          // number of vertices
      int trials = Integer.parseInt(args[1]);     // number of trials
      int[] edges = new int[trials];
      Stopwatch timer = new Stopwatch();

      // repeat the experiment trials times
      for (int t = 0; t < trials; t++) {
            edges[t] = count(n);
      }
      double timed = timer.elapsedTime();

      // report statistics
      StdOut.println(``n                       = `` + n);
      StdOut.println(``mean of number of edges = `` + StdStats.mean(edges));
      StdOut.println(``stddev of mean          = `` + StdStats.stddev(edges));
      StdOut.println(``elapsed time            = `` + timed);
 }
```

Exercise 3 – Analysis of Running Time. Algorithms Textbook 1.4

Consider the code example in exercise 2. Paste the code to DrJava (or your preferred editor) and run the program to make following observations.

A. Run ErdosRenyi.java with n = 12500 and T = 100. Double n as appropriate. For each n, calculate the log ratio $log_2(T(2n)/T(n))$ where $T(n)$ is the time required to run the above code on a data set of size of n, using the WeightedQuickUnionUF.

n	T(n)	$log_2(T(2n)/T(n))$

B. If you observe that the ratio $log_2(T(2n)/T(n))$ column is likely converging to a specific value, write it down. Discuss how this value may be related to the model $T(n) = a \times n^b$

C. Explain why it is not a good idea to consider running times under 0.5 second.

Exercise 4 – Memory analysis. Algorithms textbook 1.4

Suppose you have an array `p[]` *as declared and initialized below. How much memory (in bytes) does the array use as a function of N? Include the memory for both the array and the points. Repeat the previous question, but use tilde notation to simplify your answer.*

```
public class Point {
    private final int x;
    private final int y;
}
Point[] p = new Point[N];
for (int i = 0; i < N; i++)
    p[i] = new Point(...);
```