PRINCETON UNIVERSITY

COS 226–Algorithms and Data Structures

Week 1: *Logistics, WeightedUnionFind, Doubling hypothesis and Percolation problem (Algorithms §1.5 & videos 1.D & 1.E)*

Version: September 15, 2017

Exercise 1 – Understanding COS 226 Course Components

A. Lectures meet twice per week, from 11 to 12:20pm on Tuesdays and Thursdays in Thomas Lab 003.

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. Optional Weekly Review sessions. Review sessions meet at 3:30 to 4:20pm on Friday afternoons in a room TBA.

E. Supplemental videos to lectures are available through `http://salon.cs.princeton.edu`.

Exercise 2 – WeightedQuickUnionUF

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

```
1  /*************************************************************************
2  Name:    Andy Guna
3  NetID:   guna
4  Precept: P99
5
6  Description: This program demonstrates the use of various classes in
7  algs4.jar (WeightedQuickUnionUF, StdRandom, Stopwatch, and StdOut).
8
9  The code addresses the following problem. Given a set of n vertices,
10 suppose that in each step you select two vertices at random and
11 connect them with an edge. How many steps will it take until there is
12 a path between two specified vertices?
13
14 *************************************************************************/
15
16 import edu.princeton.cs.algs4.StdOut;
17 import edu.princeton.cs.algs4.StdRandom;
18 import edu.princeton.cs.algs4.Stopwatch;
19 import edu.princeton.cs.algs4.WeightedQuickUnionUF;
20
21 public class UFExample1 {
22 public static void main(String[] args) {
23     Stopwatch timer = new Stopwatch();
24     int n = Integer.parseInt(args[0]);
25
26     WeightedQuickUnionUF uf = new WeightedQuickUnionUF(n);
27
28     for (int steps = 1; true; steps++) {
29
30         // pick two vertices, uniformly at random
31         int v = StdRandom.uniform(n);
32         int w = StdRandom.uniform(n);
33
34         // add edge v-w
35         uf.union(v, w);
36         StdOut.println("adding␣edge␣" + v + "-" + w);
37
38         // stop if vertices 0 and n-1 are connected
39         if (uf.connected(0, n-1)) {
40             StdOut.println("connected␣after␣" + steps + "␣steps");
41             break;
42         }
43     }
44
45     StdOut.println("elapsed␣time␣=␣" + timer.elapsedTime());
46 }
```

Exercise 3 – Analysis of Running Time

Consider the code example in exercise 2. Paste the code to DrJava and run the program to make following observations.

A. Run UFExample1.java for the values of $n = 10^6, 2 \times 10^6, 4 \times 10^6, 8 \times 10^6, 16 \times 10^6, 32 \times 10^6$, and 64×10^6. Complete the table of values n, $T(n)$, and 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 may not be a good idea to consider running times under 0.1 second.

Exercise 4 – Percolation Model

Your first assignment is to write a program to estimate the value of the percolation threshold via Monte Carlo simulation.

A. What is percolation and why is it interesting to study this problem?

B. What is percolation threashold and why is it important to know this value?

C. How can percolation problem be efficiently solved using a UnionFind data structure?

D. The method percolates() in the API can be implemented at a cost of $2N^2$, find() operations. Please explain how? Now design a clever algorithm where it can be solved using a constant number of find() opearations.

E. Discuss key points when working on the percolation assignment.

