COS 226 Data Structures and Algorithms
Computer Science Department
Princeton University
Fall 2015

Week 1 Activities

1. **Course website and Blackboard (5 mins)**

   Take a tour of the website. Show how to access lectures, readings, study guides, demos assignments, homework, old exams, grades (Blackboard).

2. **Setting up the Programming Environment (15 mins)**

   Set up the course programming environment. Use drjava or eclipse. Install algs4.jar. Do a simple programming assignment with QuickFindUF class and WeightedQuickUnionUF class. A sample program is given below.

```
import edu.princeton.cs.algs4.WeightedQuickUnionUF;
import edu.princeton.cs.algs4.Stopwatch;
import java.util.Random;

public class UFExample1 {
    public static void main(String[] args) {
        Stopwatch Clock = new Stopwatch();
        int n = Integer.parseInt(args[0]);
        int nexttolast = n-1;
        WeightedQuickUnionUF UF1 = new WeightedQuickUnionUF(n);
        Random R = new Random();
        while (true) {
           int i = R.nextInt(n);
           int j = R.nextInt(n);
           if (!UF1.connected(i,j)){
              UF1.union(i,j);
              System.out.println("connecting " + i + " and " + j);
           }
           if (UF1.connected(0,n-1)) {
             System.out.println("\n EXITING ... now  0 and " + nexttolast + " are c
             break;
           }
        }
        System.out.println("The elapsed time is " + Clock.elapsedTime());

    }
}
```

3. **Analysis of runtime (10 mins)**
   The runtime of an algorithm can be estimated by using experimental values.

   (a) Build a table of values, N versus runtime T using the WeightedQuickUnionUF. Consider only the values greater than 1 second. Explain why we would not consider times under 1 second.

   (b) Use the formula $T = aN^b$ to estimate the values of a and b

4. **Percolation Assignment (15 mins)**

   The first programming assigment is to write a program to estimate the value of the percolation threshold via Monte Carlo simulation.

   (a) Explain the notion of percolation and how Union-Find can be used to simulate a percolating system.

   (b) Explain the methods in the class

```
public class Percolation {
    public Percolation(int N)              // create N-by-N grid, with all site
    public void open(int row, int col)     // open the site (row, col) if it is
    public boolean isOpen(int row, int col) // is the site (row, col) open?
    public boolean isFull(int row, int col) // is the site (row, col) full?
    public int numberOfOpenSites()         // number of open sites
    public boolean percolates()            // does the system percolate?
}
```

   (c) Explain why all methods should take constant time. A good time to discuss the WeightedQuickUnionUF

   (d) Explain the backwash problem and ask students to come up with strategies to solve the backwash problem.

   (e) Discuss the deliverables, Percolation.java and PercolationStats.java and readme.txt files. More specifically discuss readme.txt