Week 1 Activities

1. Running and testing Sample Code (10 mins)

Copy the following code and header information from course precept page, compile and run. This code uses algs4 library WeightedQuickUnionUF.

```java
import edu.princeton.cs.algs4.WightedQuickUnionUF;
import edu.princeton.cs.algs4.Stopwatch;
import edu.princeton.cs.algs4.StdRandom;

public class UFExample1 {
    public static void main(String[] args) {
        Stopwatch Clock = new Stopwatch();
        int N = Integer.parseInt(args[0]);
        WeightedQuickUnionUF UFL = new WeightedQuickUnionUF(N);
        while (true) {
            int i = StdRandom.uniform(N);
            int j = StdRandom.uniform(N);
            if (!UFL.connected(i,j)) {
                UFL.union(i,j);
            }
            if (UFL.connected(0,N-1)) {
                // done message
                break;
            }
        }
        System.out.println(N + " " + Clock.elapsedTime());
    }
}
```

2. Analysis of runtime (15 mins)

The runtime of an algorithm can be estimated using experimental values.

(a) Build a table of values, \( N \) versus runtime \( T \) using the UFExample1 (given above). Consider only the run times greater than 1 second (why?). Use the doubling principle to increase the value of \( N \) (That is, \( N = 1000, 2000, 4000 \) etc).

<table>
<thead>
<tr>
<th>( N )</th>
<th>( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Assuming the code runs in polynomial time, we will use the formula \( T = aN^b \) to estimate runtime \( T \) for a data set of size \( N \). Compute the values of \( a \) and \( b \) up to two decimal places (Do not round the exponent).
3. Memory Analysis (10 mins)

Suppose that the Java library java.util.LinkedList is implemented using a doubly-linked list, maintaining a reference to the first and last node in the list, along with its size.

```java
public class LinkedList<Item> {
    private Node first; // the first node in the linked list
    private Node last; // the last node in the linked list
    private int N; // number of items in the linked list
    private class Node {
        private Item item; // the item reference
        private Node next, prev; // the next and previous nodes
    }
    ...
}
```

Using the 64-bit memory cost model from the textbook, how much memory (in bytes) does a Node object use and how much does a LinkedList object use to store N items? Do not include the memory for the items themselves but do include the memory for the references to them.

(a) Memory of a node

(b) Memory of a LinkedList with N nodes.

4. Percolation Assignment (15 mins)

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

(a) What is percolation and how can Union-Find be used to simulate a percolating system?

(b) Learn the methods to be implemented in the Percolation class.

```java
public class Percolation {
    public Percolation(int N);
    public void open(int row, int col);
    public boolean isOpen(int row, int col);
    public boolean isFull(int row, int col);
    public int numberOfOpenSites();
    public boolean percolates();
}
```

(c) WeightedQuickUnionUF is from algs4. What is the runtime complexity of WeightedQuickUnionUF methods, union and find?

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