CMPSC 250
Analysis of Algorithms
Fall 2012
Bob Roos

Lab 2
Tues., 11 September 2012
Hand In ...

In today's lab you'll examine several programs and try to get an estimate of their running time. You'll also take a look at the "`FastFindUF`" program which we began discussing on Monday (Section 1.5 in your book).

I strongly suggest that you create a new subdirectory called "lab2" or something similar and place all your files from this lab in that directory. At the end of the assignment I'll show you how to create a "zip" file to upload to the Sakai drop box.

1. Create a document for answers to some of the questions below. BE SURE TO PUT YOUR NAME, THE DATE, AND THE LAB NUMBER at the top of the file so I'll be able to identify it as yours when I print it out!

   Go to the Sakai web site and download the program "`ThreeSumLab2.java`". This is a variation of the program we looked at in class on Monday (`ThreeSum.java`, described on pages 173–175 of your textbook). The modified version simply repeats the computation for different values of array size n and plot the running times as a function of n.

   ADD A COMMENT AT THE TOP WITH YOUR NAME so I can identify it when (if) I print it out to return to you!

   Try it; save the resulting graph (which may take a few minutes to plot) in a file named "`3sum.png`" in your lab2 folder.

2. How can we be sure that this graph really looks like cn^3 for some constant c? One way is to compare the running time for different values of n with the value of the function n^3. If the ratio is roughly constant regardless of n then we can feel pretty confident about the time.

   Add the following lines to program "`ThreeSumLab2.java`" (these go at the bottom of the inner loop, right after the `StdDraw.point` call), then compile and run the program:

   ```
   long n3 = n * (long) n * (long) n;
   StdOut.println(n3 / elapse);
   ```

   The printed output is just a series of values; they should appear to be converging to a value c. (If not, ask!) What is this value (to two significant digits, at least)?

3. Add lines to the program to plot the value of n^3/c for the value of c you found. Print it in a different color, e.g., red. (You will have to add `StdDraw.setPenColor` statements to switch between black and red.) (When I did this, the two graphs were almost a perfect match; the red points overwrote the black ones.)

Save the resulting graph as "`3sumfinal.png`".

4. Download the program `ThreeSumFast.java` from the Sakai site. This is identical to the program on page 190 of your textbook. It speeds up the problem from n^3 to $n^2 \log n$. We'll talk about how this is done in class; for now I want you to simply repeat what we did above:

   (a) Modify this file just as I did for the original `ThreeSum` program:

   - Eliminate the need for an input file; instead, use random integer values
   - Repeat the computation for values of n in the range 100, 200, 300, ..., 5000, calculating execution time using a `Stopwatch` object
   - Plot the execution time as a function of n (you may need to modify the y-scale of the plot since it will be faster than before)
   - Determine whether or not there is a constant ratio c between the execution time and the function $n^2 \log n$
   - Plot $n^2 \log n / c$ along side the timing plot

   Save your timing plot in a file named `fast3sumfinal.png`

5. Download the file "`QuickFindUF.java`" from the Sakai site. (This is a modified version of the file on the textbook website.)

   Run the program, supplying data "by hand" rather than from a data file. First, enter an integer representing the number of objects, n. Then enter a sequence of "connect" commands by entering pairs of numbers between 0 and $n-1$; if the pair belong to different components they are joined and the new table of id values is printed. (If they are already connected nothing is printed.) For instance,

```
Welcome to DrJava.  Working directory is /Users/rroos/Dropbox/cs250f2012
> run QuickFindUF
 10
 1 2
  i:    0  1  2  3  4  5  6  7  8  9
       -- -- -- -- -- -- -- -- -- --
id[i]:  0  2  2  3  4  5  6  7  8  9

 4 6
  i:    0  1  2  3  4  5  6  7  8  9
       -- -- -- -- -- -- -- -- -- --
id[i]:  0  2  2  3  6  5  6  7  8  9

 3 1
  i:    0  1  2  3  4  5  6  7  8  9
       -- -- -- -- -- -- -- -- -- --
id[i]:  0  2  2  2  6  5  6  7  8  9

 3 6
  i:    0  1  2  3  4  5  6  7  8  9
```

```
        -- -- -- -- -- -- -- -- -- --
id[i]:  0  6  6  6  6  5  6  7  8  9

 6 5
  i:    0  1  2  3  4  5  6  7  8  9
        -- -- -- -- -- -- -- -- -- --
id[i]:  0  5  5  5  5  5  5  7  8  9

 5 9
  i:    0  1  2  3  4  5  6  7  8  9
        -- -- -- -- -- -- -- -- -- --
id[i]:  0  9  9  9  9  9  9  7  8  9

 9 5

 ^D
4 components
```

Answer the following questions:

(a) For n elements, what is the *minimum* number of join commands needed to reduce the n components down to one component? (In other words, how many pairs of numbers, at a minimum, must you enter before you have just one component?) Express your answer as a function of n (for instance "$n + \log_2 n$" or "n^2" or something like that). Your function doesn't have to give precise results, but should be of the right general form (e.g., quadratic or linear or ....)

(b) Each time two components are joined together, the entire `id` array must be scanned to update the id numbers of one of the two components. What is the minimum number of *array accesses* needed to reduce the n original values to a single component? (An "array access" is any attempt to examine or change a value in an array.) Again, try to express your answer as a function of n that accurately describes the general category—linear or ....

(c) **[Extra credit—not trivial!]** Modify the main method so that it automatically makes the correct calls to `union` in order to reduce the number of components to 1 with the fewest calls. Then add code that demonstrates that it works (either via print statements or a graph).

6. Move up one level from your `lab2` directory (e.g., "`cd ..`") and type "`zip -r lastnamelab2.zip lab2`" (if you named it something else, use the something else). Now upload this to your drop box.