CS278 **Design and Analysis of Algorithms**
Assignment # 1 Due Date: September 13, 2001

Note: Create a subdirectory in your home directory and call it `your_last_name.1`, where `your_last_name` is your real last name (for example, `bhatia` for me). Do all the programs for this assignment in that directory. After you are done, submit the code by typing the following command:

```
~sanjiv/bin/handin your_last_name.1 cs278 1
```

Again, do not forget to substitute for `your_last_name`. The command should be executed from your home directory.

Do remove the object files and executables from your directory before submission. I'll encourage you to use RCS, and will like to see the use of `Makefile`.

1. [10 pt] Show the contents of the `id` array after each *union* operation when you use the weighted union algorithm to solve the connectivity problem for the sequence 0-2, 1-4, 2-5, 3-6, 0-4, 6-0, and 1-3. Also give the number of times the program accesses the `id` array for each input pair.

2. [45 pt] Write a function corresponding to each of quick find, quick union, and weighted union algorithms such that you can determine the amount of time required to solve the problem. Run each of the programs with random inputs and neatly tabulate the results. Make sure that you call the functions on the same input for one row of the table. You have to decide the input data constraints but use at least 10,000 pairs.

   Hint: Use the Unix system call `times(2)` to determine the run-time. Break the run time into overall time, user time, and system time. You may have to use the library function `gethrtime(3C)` to get the overall time.

3. [10 pt] Compute the *average* distance from a node to the root in a worst-case tree of 2^n nodes built by the weighted quick union algorithm.

4. [25 pt] Consider the following loop:

   ```
   int i, j, k, count = 0;

   for ( i = 0; i < N; i++ )
       for ( j = 0; j < N; j++ )
           for ( k = 0; k < N; k++ )
               count++;
   ```

   Develop a polynomial expression to accurately describe the running time of this code. Implement this and compute the actual running time for N as 10, 100, and 1000. Does the run time change if you use compiler optimization options? Check the compiler optimization options from the man page of `gcc` or `g++` and show the effect (or lack thereof) of at least five combinations of options.

Note: Problem 1 and 3 are to be done on paper while the other two problems require the use of machine. I do want hardcopy of all your code. The assignment is due at the beginning of class.