Spring ’16 CIS 212 Assignment 2 – 100/100 points possible – Due Wednesday, 4-13, 11:59 PM

The goal of this assignment is to become comfortable using the basic Array and ArrayList data structures and to gain some exposure to sparse arrays. A sparse array is a data structure that does not require any memory to represent an entry with the value 0, which is useful for low-density data. We’ll run a simple algorithm on the two array types as a quantitative experiment to compare the execution times of the two implementations.

For example, a dense array might look like:

[1, 10, 15, 0, 0, 2, 0, 7, 43, 0, 12, 0, 0]

The corresponding sparse array, containing all data necessary to expand back to the sparse representation, night look like:

[[0, 1], [1, 10], [2, 15], [5, 2], [7, 7], [8, 43], [10, 12]]

1. [10] Write a Java program that first prompts the user for an integer array length and a double-precision array density. You will use these values to create the arrays for the experiment (see parts 2 and 3 below). Catch any exceptions due to the user entering unparsable input and prompt them to reenter the value(s). Also prompt the user to reenter values(s) if they are outside of the expected range (see below).

2. [10] Write a method which takes an integer length and an array density of type double as arguments and returns a new array of type int representing a dense array. For each entry in the array, compare the density with a random number on the range [0.0, 1.0) (i.e., 0.0 up to, but not including, 1.0) to determine whether or not the entry should be 0 (hint: see the java.util.Random class). If the entry should be 0, simply populate the entry as such. If the entry should be non-zero, populate it with a random integer on the range [1, 1000000] (i.e., 1 through 1 million). In this way, specifying an array length of 100 and a density of 0.25 would result in an array of length 100 with on average 75% of its values equal to 0.

3. [10] Write a method which takes an integer length and an array density of type double as arguments and returns a new ArrayList representing a sparse array. As above, use the density to determine whether or not each entry should be 0. If the entry should be 0, simply do nothing (i.e., don’t add it to the ArrayList). If the entry should be non-zero, store its index and value (also on the range [1, 1000000]) in the ArrayList (hint: you may want to use an ArrayList of type int[] – note that this is not the most elegant solution but we have not yet covered objects in Java). In this way, specifying an array length of 100 and a density of 0.25 would result in an ArrayList with on average 25 elements, each specifying the index and value of a non-zero integer.

4. [10] Write a method which takes a dense array as an argument and returns a new equivalent sparse array.

5. [10] Write a method which takes a sparse array as an argument and returns a new equivalent dense array. The dense array only needs to be large enough to fit all of the values. For example, the resulting dense array only needs to hold 90 values if the last element in the sparse array is at index 89.

6. [10] Write a method which takes a dense array as an argument and prints the max value in the array, along with the index of that value in the array.

7. [10] Write a method which takes a sparse array as an argument and prints the max value in the array, along with the original index of that value in the array (i.e., the index that you stored in part 3).

8. [10] Use the System.nanoTime() method to record the amount of time taken to run each of the above methods (i.e., steps 2-5 above). Print your timing results in fractional milliseconds (e.g., 0.5 milliseconds). Spend some time trying different combinations of inputs and write your findings in the comments of your code. Which implementation is faster for various cases?

9. [20] Write code that is clear and efficient. Specifically, your code should be indented with respect to code blocks, avoid code that is commented out or otherwise unused, use descriptive and consistent class/method/variable names, etc.

Your output should look something like:

```
Please array length:

10000000

Enter density:

0.001

create dense length: 10000000 time: 285.08398

convert to sparse length: 9926 time: 11.68

create sparse length: 9928 time: 263.208

convert to dense length: 9999961 time: 39.051

find max (dense): 999827 at: 7330114

dense find time: 10.227
```

```
find max (sparse): 999627 at: 6534581

sparse find time: 0.608
```

Zip the Assignment2 folder in your Eclipse workspace directory, name the .zip file <Your Full Name>Assignment2.zip (e.g., EricWillsAssignment2.zip), and upload the .zip file to Canvas (see Assignments section for submission link).