CS146 Data Structures and Algorithms

Spring 2016, Instructor: K. Potika SJSU CS Department

Programming Project 1

Divide and Conquer

Important: Do individually (each student alone is NOT a group assignment), each student has to turn in a java program.

Part1) Matrix Multiplication

Implement two types of algorithms for multiplying two n × n matrices. Assume n is a power of 2:

1. The straight-forward O(n^3) matrix multiplication algorithm.
2. Strassen's matrix multiplication algorithm.

Evaluate your different algorithms, and write a short report. Create test matrices for different values of n (4, 16, 512 and 1024). Generate matrices using random numbers. Compute the running time of your algorithms. Your report should include the running times and conclusions.

Sample Test class

```
public class MatrixMultiplicatonTest
{
  public static void main(String[] args) {
        // create two double 2-D arrays
        double[][] a…
        double[][] b…
        Matrix m1 = new Matrix(a);
        Matrix m2 = new Matrix(b);
        System.out.println(m1);
        Matrix productStrassen=m1.multiplyStrassen(m2);
        Matrix productRegular=m1.multiply(m2);
        System.out.println("Are matrices the same?
                    "+productStrassen.equals(productRegular));
                    System.out.println(productStrassen);
                    …
  }

}
```

Part2) Quicksort

Implement the simplest version of the quicksort algorithm by choosing as pivot:

1. the last element
2. the median (select algorithm)

Count the time and the number of comparisons for this version when sorting an array of random generated numbers and display the totals. Test it with large random generated arrays (10,000 - 100 million).

Use a very simple test driver that runs all two quicksort algorithms on a large array of randomly chosen integers and then verifies that the result is in order. What other practical improvements can you add? Your output should have something that looks like the following total runtime measurements for arrays of various sizes:

Array size	10,000	100,000	1,000,000	10,000,000	100,000,000
Total time (s) of QS1					
Total time (s) of QS2					
Comparisons in QS1					
Comparisons in QS2					

Sample Test class

```
public class QuickSortTest
{
	public static void main(String[] args)
	{
		int maxSize = SIZE; // array size
		int[] arr=...// create array

		arr.quickSort1(); // quicksort version 1
		arr.quickSort2(); // quicksort version 2
		...
	}
}
```

Note: To count the time use *system.currentTimeMillis()*. Every time you compare a pair of numbers increase an appropriate counter. The first programming assignment is basically to learn JUnit testing.

Deliverables:

- ✓ Your main grade will be based on (a) how well your tests cover your own code, (b) how well your code does on your tests (create for all non-trivial methods), and (c) how well your code does on my tests (which you have to add to your test file). For JUnit tests check canvas.
- ✓ Use sjsu.<lastname>.cs146.project1.<part1/2> as your package, and Test classes should be your main java file, along with your JUnit java tests.
- ✓ Zip up the directory with your entire project (source code and report). Turn the zip file by uploading it to canvas
- ✓ All projects need to compile. If your program does not compile you will receive 0 points on this project.
- ✓ Do not use any fancy libraries. We should be able to compile it under standard installs. Include a readme file on how to you compile the project.