CMPS 2200 Introduction to Algorithms – Fall 15

9/22/15

Programming Project 1

Due **10/28/15** at 11:55pm on Blackboard

Matrix Multiplication (50 points)

Assume n is a power of 2. Implement the following four algorithms for multiplying two $n \times n$ matrices:

1. (10 points) The straight-forward $\Theta(n^3)$ matrix multiplication algorithm.
2. (10 points) The recursive $\Theta(n^3)$ matrix multiplication algorithm.
3. (10 points) Strassen's matrix multiplication algorithm.
4. (10 points) A mixture of the straight-forward algorithm and Strassen's algorithm: Assume some parameter a is given. For all recursive calls of Strassen's algorithm in which $n > a$ use the regular recursion by Strassen. If $n \leq a$, use the straight-forward $\Theta(n^3)$ algorithm (i.e., this is the "base case" of this algorithm).

Evaluate your different algorithms, and write a short report. This evaluation (consisting of report, test cases, and test code) will be worth 10 points. For this, create test matrices for different values of n, and record the runtimes of your four algorithms. For the fourth algorithm also vary the parameter a. The range for n should reach at least $n = 1024$ and a should reach at least $a = 32$. Your report should include the runtimes as well as a conclusion as to which algorithm performs best.

Turnin instructions

- You can use Java, C, C++, or Python for this project. If you want to use a different programming language, check with the instructor or TA first.
- **The name of your project directory should be** `project1_<lastName><firstName>`
- Zip up a directory with your entire project (source code and report). Turn in the zip file on Blackboard.
- All projects need to compile and run. 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 of Java, C, C++, or Python. You may want to include some comments how you compiled the project.

Flip over to back page ⟹

Code for runtime measurements

Below is code that you can use to measure the execution time of a `<code snippet>` in seconds.

- Python:

  ```
  start_time = time.time()
  <code snippet>
  total_time = time.time() - start_time
  ```

- C/C++:

  ```
  #include <time.h>
  ...
  double total_time;
  double start_time = (double)clock();
  <code snippet>
  total_time = ((double)clock() - start_time)/CLOCKS_PER_SEC;
  ```

- Java:

  ```
  double total_time;
  long start_time = System.currentTimeMillis();
  <code snippet>
  total_time = (double)(System.currentTimeMillis() - start_time)/1000;
  ```