Jack Dubil

2015-10-11

SE/CS 146 Section 01

Project 1 Part 1: Matrix Multiplication Report

**Basic Algorithm**

Very simple to implement compared to the other algorithm, just requiring three for loops to iterate through the matrices, along with just multiplying the appropriate elements of the matrices together and putting them in the correct position in the result matrix.

**Strassen’s Algorithm**

This algorithm is much harder to implement, along with requiring a lot more code. It also requires a lot more temporary matrices throughout the algorithm. It requires eight more matrices to hold the eight quarters of the two matrices being multiplied together, seven more matrices to hold the results of the various additions and multiplications, and then four more to form the four quarters of the resulting matrix. Over all of the recursive calls to the method, these start to add up very, very quickly, and probably contribute to the results that I got in testing the algorithms.

**Running Times**

Time (in ms) to multiply two 4x4 matrices with the basic algorithm: 0

Time (in ms) to multiply two 16x16 matrices with the basic algorithm: 0

Time (in ms) to multiply two 512x512 matrices with the basic algorithm: 606

Time (in ms) to multiply two 1024x1024 matrices with the basic algorithm: 16714

Time (in ms) to multiply two 4x4 matrices with the Strassen algorithm: 0

Time (in ms) to multiply two 16x16 matrices with the Strassen algorithm: 2

Time (in ms) to multiply two 512x512 matrices with the Strassen algorithm: 35479

Time (in ms) to multiply two 1024x1024 matrices with the Strassen algorithm: 241952

**Evaluations**

Although the basic matrix multiplication algorithm is O() and Strassen’s multiplication algorithm is approximately O(), I think it makes sense that Strassen’s algorithm took so much longer in these tests. Because of the extra, constant operations that Strassen’s algorithm involves, in creating the large number of extra matrices throughout the recursion, for small to even fairly large-sized matrices like these, the constant factor will definitely overwhelm the slightly faster growth rate between the two algorithms. Of course, at the same time, when I tried using Strassen’s algorithm with 4096x4096 matrices, I actually hit a stack overflow with the amount of recursive calls on the stack using Strassen’s algorithm, so I cannot properly test that, with my computer at least.

**Conclusions**

Based on how much longer the 1024x1024 matrix took with Strassen’s algorithm compared to the basic, iterative one, the point at which the growth rate of the basic algorithm causes it to take longer than the slightly slower growth rate of Strassen’s algorithm is probably with only VERY large matrices. However, in looking up information about the algorithms while doing the project, I saw some people suggest that having a point at which the ends of the recursion are simply solved through the basic algorithm, and that seems like a good way to deal with the downsides of a pure version of Strassen’s algorithm as I implemented here. Especially since according to the tests above, the basic algorithm is actually still very fast even for fairly large matrices.