Coding Assignment 2

CS 124, Spring 2021

Due Friday, March 26, 2021

Thomas Jaklitsch, no collaborators
Number of late days on previous problem sets: 10
Number of late days including this problem set: 12
Task 1
Let us first look at normal matrix multiplication. Say we are multiplying matrix A by matrix B, which are $n \times n$. Then multiplying one row in A by one row in B will take n real number multiplications, and $n-1$ additions. Therefore, since we need to multiply n rows by n columns, we have that this takes $(2n-1)n^2 = 2n^3 - n^2$ operations.
Now for Strassen's algorithm, first let us assume that n is a power of 2 we have a recurrence in which we do 7 matrix multiplications where each sub-matrix is $n/2 \times n/2$. We also have to do a total of 18 additions or subtractions of $n/2 \times n/2$ matrices. Therefore, we have a recurrence of $T(n) = 7T(n/2) + 18(n/2)^2$.
Now we want to find the best place to switch from Strassen's algorithm to regular matrix multiplication. Let us take this value of n to be n_0. Therefore, at n_0, Strassen's algorithm will switch to using $2n^3 - n^2$ for the recursive step. Also, since we want this to be the ideal place to switch over, we want regular matrix multiplication to take the same number of operations here. Therefore, we get $2n_0^3 - n_0^2 = 7T(n_0/2) + 18(n_0/2)^2 = 7(2(n_0/2)^3 - (n_0/2)^2) + 18(n_0/2)^2$.
If we solve for n_0 in the equation $2n_0^3 - n_0^2 = 7(2n_0^3 - n_0^2) + 18(n_0/2)^2$, we get $n_0 = 15$.
This is for when n is a power of 2, however. If n is ever odd during Strassen's algorithm, however, then we have to account for the fact that we are splitting the matrices not into $n/2$ matrices but into $(n+1)/2$ matrices. Therefore, the equation becomes
$2n_0^3 - n_0^2 = 7(2((n_0+1)/2)^3 - ((n_0+1)/2)^2) + 18((n_0+1)/2)^2$.
When we solve for n_0 in this equation, we get that the value is approximately 37.12. Therefore, using this form of analysis for the runtime of the program, we find that the best place to switch between Strassen and regular matrix multiplication is between 15 and 37 inclusive depending on what the dimensions of the matrix are.
Task 2
For this task I wrote my code in C. I calculated the time taken to compute the product of matrices and varied the size of my cutoff point to find the optimal cutoff point to switch between Strassen's algorithm and the standard algorithm for matrix multiplication.
I began by completing the Strassen algorithm for powers of 2. I then extended this procedure to matrices that are any dimension n. The way I completed the algorithm for every dimension was by adding 0's as padding so that my matrices were even dimensions when I ran Strassen's algorithm on them. The way I attempted to optimize this algorithm was by only adding padding to matrices when the dimension was odd. I found that this was a better strategy than simply padding the matrices with 0's up to powers of two. This made my Strassen algorithm run more efficiently and still produced an accurate result, since the 0's were added to the ends of the rows and columns so they did not effect computation when adding or multiplying matrices.

The values I computed for the crossover point were larger than the approximations I calculated for task 1. One thing that they had in common was that I could get a better crossover point when the dimension of the matrices were a power of 2. For powers of 2, the optimal crossover took place at 64. This was lower than what I got for the general case which was about 200.
I calculated a higher crossover point for general dimensions, because I had to add a row and column of zeroes to the matrices when their dimensions were odd. The reason that both my crossover points were higher than what I calculated in task 1 was most likely due to the cost of other operations I did not consider in my analysis. For example, I used a significant amount of memory allocation in my implementation, which was not accounted for in task 1. I believe that my biggest loss in efficiency when using Strassen's algorithm came from these memory allocations.
I also used matrices that were completely populated when calculating my crossover points to ensure that my crossover points would work for the most time consuming matrix multiplications. Fully populated matrices take more time to compute, because the values in the matrix will grow to be large and multiplying large numbers takes more time than small ones, since there are more single number multiplications and additions to account for.

Task 3

For this task I used Strassen's algorithm to compute the triangles in the random graphs for $p = .01, .02, .03, .04$ and $.05$ using the method described in the assignment. I ran the algorithm 5 times for each p value. Below is a chart with these 5 trials for each value compared to the expected result. On the top of the chart are my trials $1 - 5$ and expected. On the left hand side of the chart are the different values for p.

	1	2	3	4	5	expected
.01	181	161	184	163	188	178.433
.02	1,439	1,296	1,368	1,464	1,400	1,427.464
.03	4,684	4,595	5,007	4,838	4,757	4,817.692
.04	11,729	11,293	11,173	11,532	11,373	11,419.714
.05	21,637	21,353	22,409	23,135	22,635	22,304.128

Overall, the trials that I ran produced results that were close to the expected with some variance. Therefore, this was what we should expect to see using this method.