CMSC 180 Laboratory Research Problem 1

Jarem Thimoty M. Arias — CMSC 180 T-5L

I. Introduction

This Research Problem serves as a review on implementation and analysis of Algorithms before implementing parallel programming. In this paper specifically, we will be dealing with the computation of the Pearson Correlation Coefficient on a 2D Matrix that we will call X, whose columns of data shall be compared to a constant array we will call Y. This process will be done in a serial fashion before the next paper which will have improvements using parallel programming.

February 23, 2024

A. Research Question 1

What do you think is the complexity of solving the Pearson Correlation Coefficient vector of an n x n square matrix X with a n x 1 vector y? (hint: CMSC 142)

From my understanding of complexities, the problem above will most likely have a complexity of $O(n^2)$ as the growth of the n x n square matrix has the most impact when it comes to both the space and time complexity of the program.

II. Objectives

1) Understand the Pearson Correlation Coefficient Algorithm and implement it into C
2) Review the concepts of Time and Space complexity when it comes to Analyzing Algorithms
3) Analyze the results from running the Pearson Correlation Program and determine different conclusions from the results

III. Methodology

In this Laboratory Research Problem, I had decided to use C for its familiarity to the researcher and its fast speed comparatively to other languages. This language will also be used moving forward to program future Lab Research Problems.

The code has a file structure containing the following files and their uses

- main.c - Contains the codebase.
- a.out - The executable file compiled from main.c
- input.txt - The file containing the inputs to run the program
- output.txt - The file that will hold the results of the code
- makefile - A makefile containing simple commands to compile and run the code in various manners

The code also makes use of two structures named pearson_in and pearson_out which serves as the input and output of all the functions relating to the Pearson Correlation, and both make use of a technique called Flexible Array Members (IBM Documentation, n.d.) to allow them to contain dynamically sized lists which makes organizing and freeing the data much easier.

A simple system using the time.h library built into C was used to keep track of the time it took for every run of the Pearson Correlation. The Pearson Correlation Algorithm itself was based off of the teaching material used in the Laboratory Session, however the provided pseudocode was not followed to the tee.

IV. Results

N	Trial 1	Trial 2	Trial 3	Average	Complexity
100	0.00041	0.00043	0.00045	0.00043	0.00043
200	0.00173	0.00166	0.00201	0.0018	0.00172
300	0.00374	0.00402	0.00365	0.003803333	0.00387
400	0.00666	0.00713	0.00657	0.006786667	0.00688
500	0.01085	0.01027	0.01049	0.010536667	0.01075
600	0.01586	0.01559	0.01514	0.01553	0.01548
700	0.02156	0.02078	0.02028	0.020873333	0.02107
800	0.02838	0.02777	0.02751	0.027886667	0.02752
900	0.03548	0.03439	0.03348	0.03445	0.03483
1000	0.04471	0.04321	0.04227	0.043396667	0.043
2000	0.20563	0.20357	0.19711	0.202103333	0.172
4000	0.86607	0.86615	0.86501	0.865743333	0.688
8000	3.53308	3.53851	3.65723	3.576273333	2.752
16000	28.13732	28.30954	28.24609	28.23098333	11.008
20000	29.8669	29.66867	29.75433	29.7633	17.2
32000	135.37549	134.62109	134.57459	134.8570567	44.032

TABLE I
RUNTIME RESULTS OF THE CODE

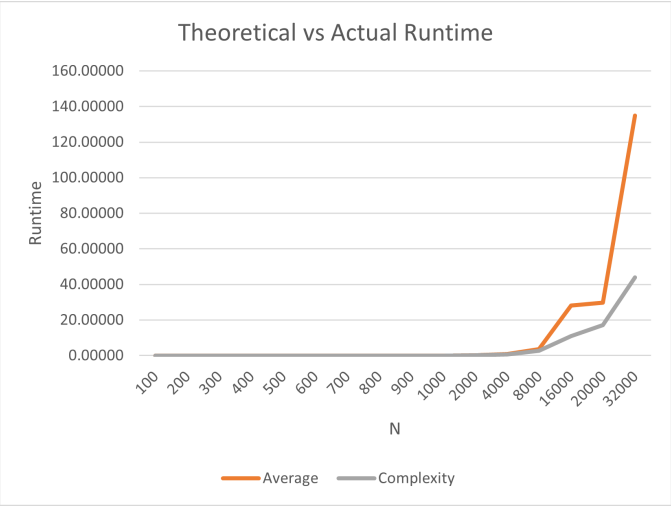

Fig. 1. graph of runtime results

A. Research Question 2

Were you able to run up to n ¿ 10,000,000? If so, can you make it higher to 50,000,000 or even 100,000,000? If not, why do you think so and what do you need to do to make it so?

I was not able to run up to n ¿ 10,000,000. In fact the limit of my code was 45,000. I was not able to run an N larger than that. The reason for this is very simple, there was not enough memory in the machine I was running for it to run the code. Running N=45000 was saturating the 16 gigabytes of memory that the PC I was using for the Laboratory Research Problem.

V. Conclusion

A. Research Question 3

Do the two lines agree, at least in the form? If not, provide an explanation why so?

The theoretical and actual runtime agreed up to a point, specifically up to around N=2000, after that, the two did not agree completely but still had the same form of an upward trend. This is not a surprise as a lot of times, the theoretical computations do not fully match the actual results from experimentation. This could be due to several factors, such as the runtime of N=100 which was used as the entire basis of the complexity accounting for other processes that was not the Pearson Correlation, changing the theoretical runtime.

Another reason could be that 3 runs is not enough of a sample size to determine the proper theoretical runtime. It could also be that the base N=100 is not big enough to be able to rule out negligible processes that is not the Pearson Correlation Algorithm such as allocating of memory which may have a significant effect on its runtime while the N is a small value.

B. Research Question 4

Discuss ways on how we can make it better (lower average runtime) without using any extra processors or cores (notice that the word “ways” is in plural form).

The first and easy way to improve the program is to look for inefficiencies in the base algorithms, not just the Pearson Correlation, there could be changes in what datatypes are used to allow for more efficient memory usage. There could also be changes to other accessory functions such as the populate function which gives values to the matrix.

And the obvious solution to speed up the code would be to implement Parallel Programming to be able to fully harness the specifications of the Computer being used rather than always being ran single threaded as it is currently programmed.

References

[1] IBM documentation. (n.d.). https://www.ibm.com/docs/en/i/7.2?topic=declarations-flexible-array-members