605.420: Algorithms for Bioinformatics

Jack Zhan

Project 1 Analysis

02/21/2016

Running Time Analysis of Matrix Multiplication

The following code fragment implements Matrix Multiplication:

```
39              for(int i=0; i<order; i++){
40                  for(int j=0; j<order; j++){
41                      sum=0;
42                      for(int k=0; k<order; k++){
43                          sum += matrix1[i][k]*matrix2[k][j];
44                          counter += 1;
45                      }
46                      matrix3[i][j] = sum;
47                  }
48              }
```

The following table shows the cost and running time of each step in the code fragment. Assume that n=order.

Line	Statement	Cost	Time
39	for(int i=0; i<order; i++)	c_1	n
40	for(int j=0; j<order; j++)	c_2	n^2
41	sum=0;	c_3	n^2
42	for(int k=0; k<order; k++)	c_4	n^3
43	sum += matrix1[i][k]*matrix2[k][j];	c_5	n^3
44	counter += 1;	c_6	n^3
46	matrix3[i][j] = sum;	c_7	n^2

Line 39 is a standard for loop that goes from 0 to n-1. Therefore it has a runtime of n.

Line 40 is a standard for loop that goes from 0 to n-1. However it is nested in another for loop that has a run time of n. Therefore it has a runtime of n*n or n^2.

Line 41 and 46 is nested inside the above loop and runs in the same time as the loop itself.

Line 40 is a standard for loop that goes from 0 to n-1. However it is nested in another for loop that has a run time of n^2. Therefore it has a runtime of $n^2 * n$ or n^3.

Line 43 and 44 is nested inside the above loop and runs in the same time as the loop itself.

Adding the cost of all statements gives the following equation:

$$T\ n\ =\ c_1\ n+\ c_2+c_3+c_7\ n^2+\ c_4+c_5+c_6\ n^3$$

$$T\ n\ =\ an+\ bn^2+\ cn^3$$

Where a, b, and c are constants. The running time is therefore proportion to n^3. In addition, it can be bounded on by the constants and the asymptotic running time of this code fragment for Matrix Multiplication is:

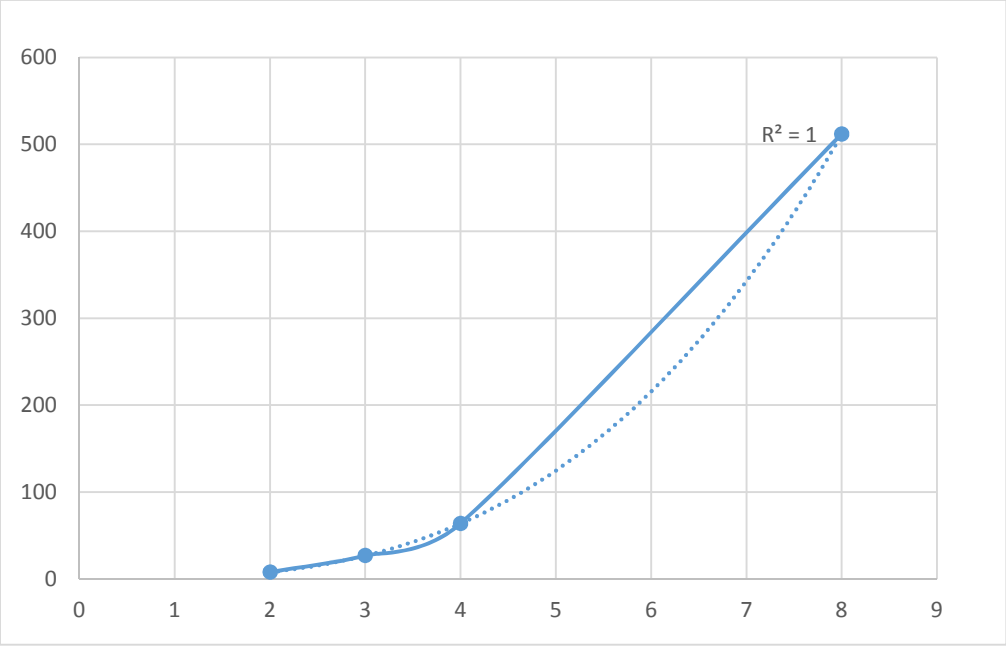
$$T\ n\ =\ \Theta(n^3)$$

Actual Run Time of Matrix Multiplication

The number of operation for the above algorithm can be seen in the output of the program.

```
 1 Matrix Multiplier created.
 2 Entered main() method.
 3 Entered readInputFile() method.
 4 Multiplying Matrix of order 2
 8 Total Number of operations = 8
 9 Multiplying Matrix of order 3
13 Total Number of operations = 27
14 Multiplying Matrix of order 4
18 Total Number of operations = 64
19 Multiplying Matrix of order 8
23 Total Number of operations = 512
```

Graphing this yields the following graph with a cubic trend line that has a R^2 value of 1.



This confirms the analysis of the data with the running time being:

$$T\ n\ = \Theta(n^3)$$

What I Learned:

The current method of analyzing the running cost of algorithms is extremely accurate. I can clearly see that $\Theta(n^3)$ running time skyrockets with an increasing order of a matrix. In addition, I relearned how to code in Java after not touching the language in over 5 years

Efficiency:

I believe that my algorithm was extremely inefficiency. The problem I was running into was there was no easy way to properly calculate the solution without implementing 3 nested loops. Matrix multiplication requires you to access row of the first matrix with each column of the second matrix and multiply the individual values. The only way I could think of in accessing the data is with multiple loops. In the future I might try experimenting with incorporating caching in the program to speed up data access. Also, incorporation a check for similar matrixes will allow the use of eigenvalues to speed up the calculation.

Justification for Design Decisions:

It was easier to code the file reader to directly read the data of the input file into arraylists. This allowed me to split reading the file and running the algorithm into separate functions. In addition I had the program directly print the result so I could check if the algorithm was running correctly. In addition, it allowed me to graph the running of the algorithm.

Relevance to Bioinformatics:

This type of analysis is important in bioinformatics because dataset can easily become very big. The results in retimes that can takes days or months to complete such as Bayesian analysis. Being able to analyze the runtime of your algorithms will allow one to pinpoint where most of time is being spent and develop a method to optimize it.