Analysis of Algorithms for Solving Binomial Coefficient Formula

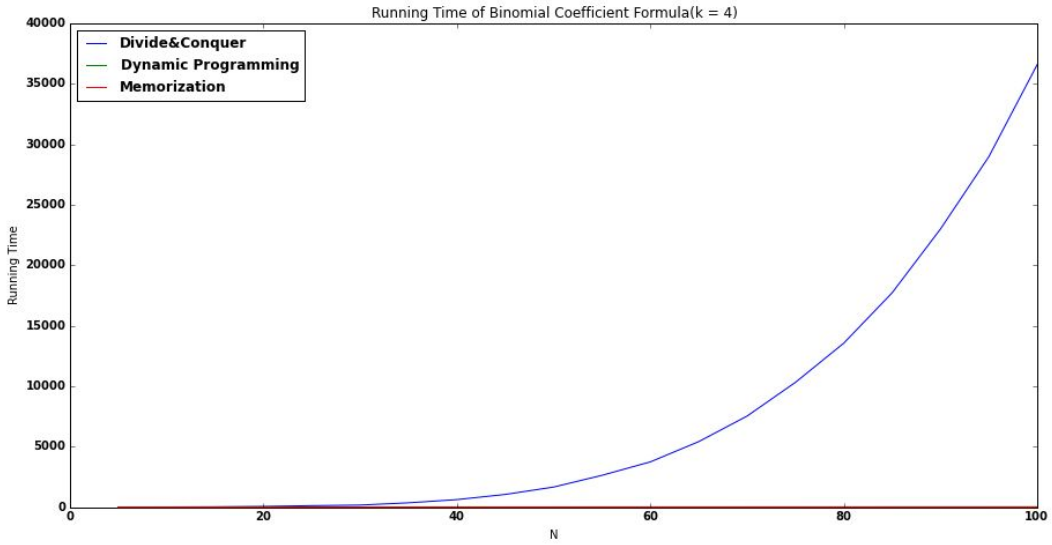
Introduction

In this assignment, I'm going to compare different approach to evaluation of binomial coefficient formula from Discrete Structures. There are three methods by which this formula can be evaluated. The first method is the traditional recursive method of divide and conquer and is inefficient due to the exponential growth of number of calculations and recursion stack overhead. The second way of evaluating the formula is by dynamic programming which is bottom up approach which solves subproblems from smallest to largest until the solution is reached. The third approach is by memorization. In this method, the recursive calls are only applied if the values for the parameters given to recursive sub-routine are never calculated before while solving the problem.

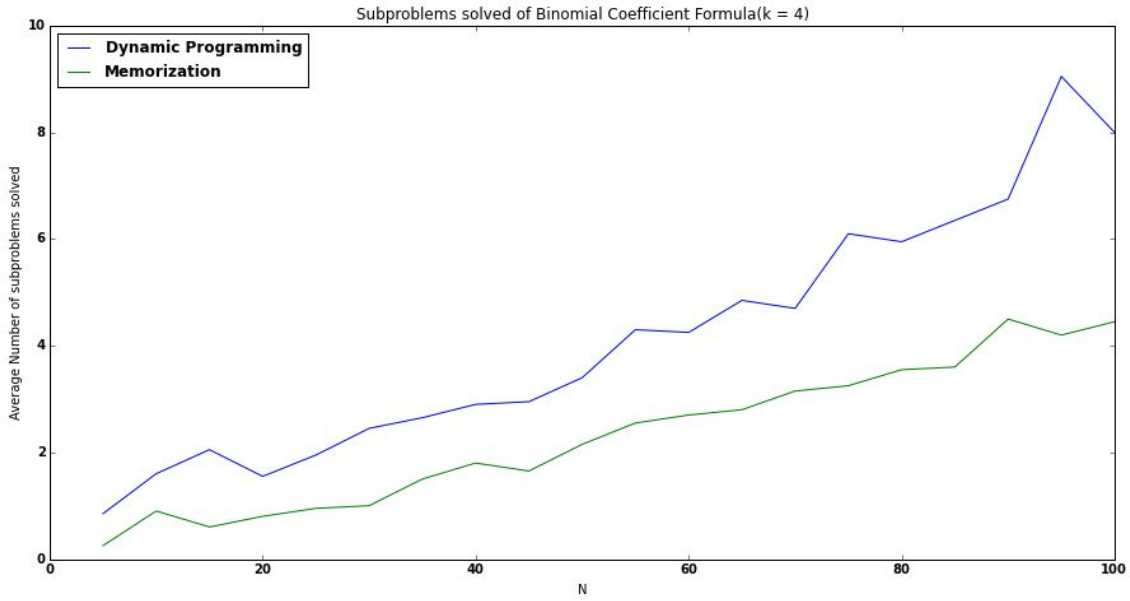
Analysis

I tried evaluating the the binomial coefficient formula by three methods mentioned above. I made *n* as a variable ranging from 5 to 100 with the increment applied as 5, so that the variables would be 5, 10, 15...100. I chose to make the value of *k* as a constant to make analysis easier. The range for n and the value of k can be altered by passing command line arguments to the program (follow README.txt for the details on how to compile and run the program). For every n and k=4, I ran 20 iterations of those three algorithms to get average running time a more accurately. I used python library *matplotlib* for plotting the running time of the algorithms. After running the program, the running time of divide and conquer method seemed to be exponentially rising compared to other two algorithms.

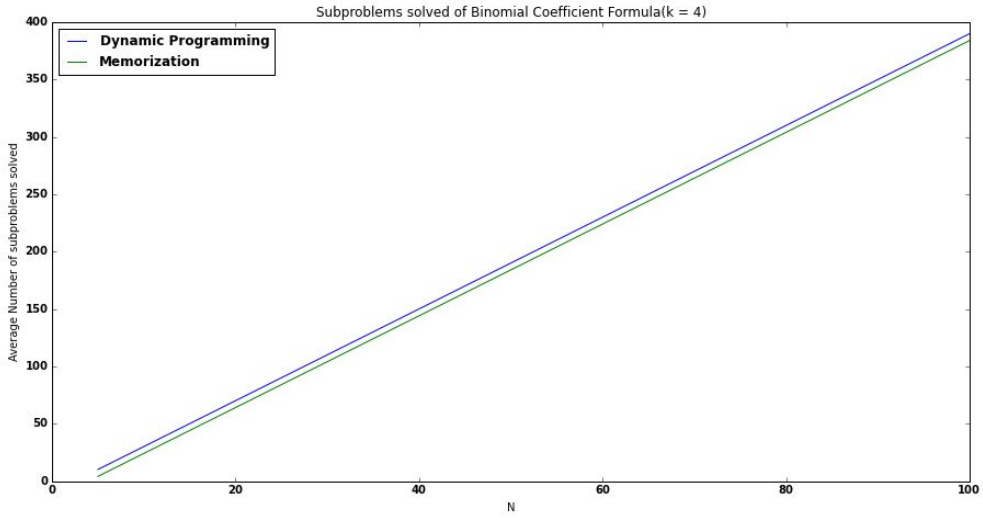
The graph below suggests that, there is no point in using the divide and conquer algorithm for calculating binomial coefficient involving larger n values.



Now, Let's compare the other two algorithms and see which one performs better. I ignored the divide and conquer algorithm and plotted other two algorithms to see which one has better running time.



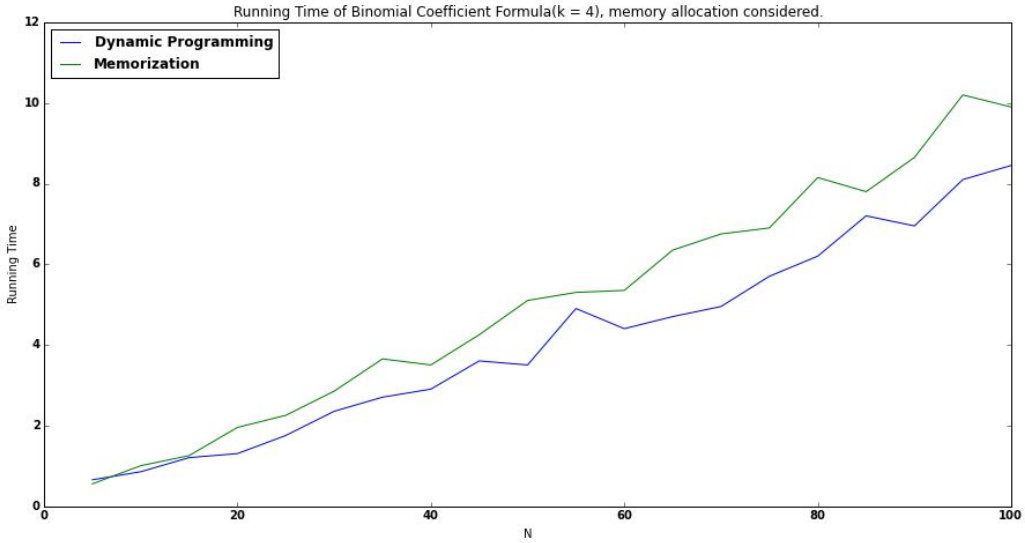
The memorization technique proved to be little bit faster, but not that much when compared to the divide and conquer algorithm. To see why the memorization technique performed better than the usual dynamic programming approach, I counted the total number of subproblems solved while solving each pair of n and k.



It seemed to me that, the number of subproblems that needed to be solved in order to reach final solution is little higher in usual dynamic programming (precisely 6 in my experiment) than

that of memorization approach. This might be the reason the memorization technique was slightly faster than the usual dynamic programming.

But, if the cost of memory allocation of 2D array while evaluating the binomial coefficient using memorization technique is considered, the running time is little higher in memorization than in the dynamic programming algorithm. The following graph shows the comparison considering the array allocation time in memorization.



Conclusion

Though divide and conquer approach to evaluating binomial formula is easy to understand and implement, it goes through number of redundant calculations and recursion overhead. This is why, other two algorithms are better choice. Even though from the experiment it can be seen that the memorization algorithm performed slightly better, there is not much difference in usual dynamic programming and memorization.