Sidney Sanders

Professor Venu Dasigi

CS 4120 – Design and Analysis of Algorithms

1 December 2021

**Project 2 – Dynamic Programming**

**Implementation and Experimental Analysis**

The two programming problems analyzed in this analysis are the Fibonacci sequence and matrix chain multiplication. To analyze these problems, I used different dynamic programming styles to compare the different complexities that can be seen and the time it takes each one to complete its task. One approach I used is the recursive approach. It is the most common and usually the simplest to design. The recursion goes through recalling itself to solve a set of subproblems to get the desired output. Another approach I used was the top-down approach. It is designed on a more hierarchical set up and flows down from the root and breaks down the problem into smaller and easier documented problems to solve. The last approach I used is the bottom-up solution, this is the opposite of the top-down approach. It works by linking solutions to subproblems until it reaches a root solution for the entire problem given.

Dynamic programming is a truly helpful method for tackling issues. It includes parsing problems into more modest covering sub-issues, putting away the outcomes figured from the sub-issues, and reusing those outcomes on bigger lumps of the problem. Dynamic programming styles identify with various other key ideas in software engineering in intriguing ways. A recursive arrangement that continues to reply to sub-issues which were at that point processed is called memoization, which is fundamentally the root of dynamic programming. Another variety is the point at which the sub-issues don't collide, in which case the strategy is known as divide and conquer.

Utilizing this method requires a sufficient amount of training. Most utilizations of dynamic programming are easy to miss and take expertise to find. It doesn't easily fall into place for me by any means. Learning these models took a considerable amount of time and thought. Becoming familiar with dynamic programming and how it fits into a broader critical thinking structure has made me see things differently, and that alone makes it worthwhile to learn.

Data expectations vs results:

Fibonacci sequence:

My expectations for the Fibonacci program are that the basic recursive function would take the most time as it isn’t very complex. My expectation for the recursive function calls I believe was correct, the recursive function did take longer than both other dynamic programming styles. I believe the top-down approach to the Fibonacci program was next of the time frame solution, my times were having issues but the top-down would take a bit longer to complete the tasks than the next dynamic approach. That approach was the bottom-up approach, which I thought would be slower than the top-down approach was I believe the quickest at coming to a solution to the problem.

 

 

 

Matrix Chain Multiplication:

My expectations for the matrix chain multiplication were pretty similar to my initial expectation of the Fibonacci sequence where the recursive would have the longest time report followed by bottom-up and then finally top-down. The data I got from running all three styles was that bottom-up was the quickest. Following that the recursive style which surprised me that it completed quicker than the top-down approach.

 