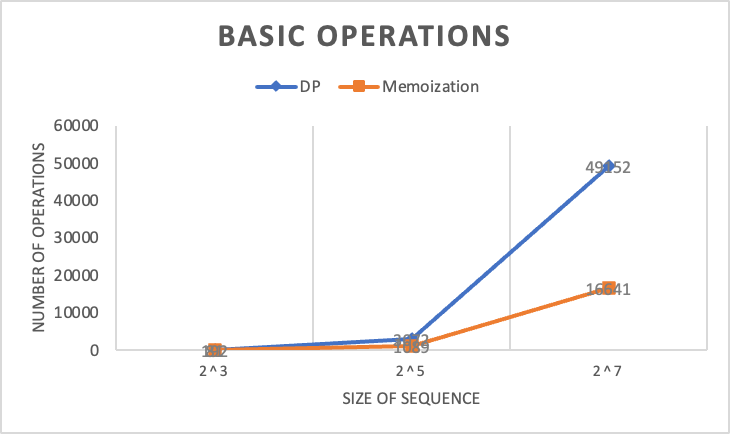
Sydnee Elliott

CS3310

Project 2

1. See code turned in alongside this document. Below are the graphs representing the estimated basic operations, the actual number of basic operations, and the run time of both the DP and Memoized D&C



Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

1. I found the DP was faster than Memoized D&C. I was surprised that this was the case because both programs have a O(m\*n) and DP uses a lot more operations than memoized D&C. After analyzing, I believe that the reason that DP ended up being generally faster is in part due to D&C relying on recursion. Recursion has more overhead than iterative algorithms like in the dynamic programming case. The extra overhead is due to repeated function calls and additional memory used for D&C memoization. Both things aren’t as much of an issue with DP in this case because it is not recursive. DP uses a table to store the subproblem values and avoids recursive calls to the function, avoiding this overhead. It is much faster using the DP approach to directly access the solutions to the subproblems rather than using lots of recursive calls.

In my code, I have the constant C and the estimated time as outputs that I’ll refer to answer part 1 of this question. For inputs of all sizes, I found that DP has a smaller constant C, which correlates with it having a faster run time than D&C. For example, with m & n being size 2^7, DP has a constant C of 1.2 e^-7 while D&C has a constant C of 8.6 e^-7. D&C has a larger constant C due to the reasons listed above, with the overhead from recursion.

My estimated times were very consistent for DP because the number of basic operations are very predictable. It will fill up the entire table every time. So, knowing the size of the two inputs, estimating times was both easy and accurate. My estimated times for D&C were close but not exact like DP was. I think it is harder to accurately determine run time because it’s harder to definitively determine how many subproblems in the memoization cache are used.

1. DP used significantly more basic operations than Memoized D&C did. The main reason for this is the memoization cache. DP does not use a cache and will fill up the entire table, storing the solutions to the subproblems which requires a large amount of computations and basic operations. Whereas, in memoized D&C, we use a cache to store our subproblem solutions and if the result is already stored in the cache, then we do not have to repeat steps and calculations, saving us basic operations and calculations.

For the same reason, memoized D&C uses significantly less operations than non-memoized D&C due to the cache. Without the cache, D&C is repeatedly solving subproblems it already has, adding to the number of basic operations performed. While memoized D&C stores those solutions and avoids recomputation, saving basic operations.

My estimated basic operations for DP were very consistent. For the Gene Sequencing DP, the number of basic operations was always 3 \* m \* n. 3 basic operations per square in the table, with the table being the size of the two inputs, m and n. It was easy to calculate the basic operations and the results were consistent and accurate. My estimated basic operations for memoized D&C were close but not spot on, the same reasons listed in question 2. With the memoization cache it makes it harder to accurately predict the time or constant C, so my predictions were not as accurate.