**Longest Common Subsequence**

In this assignment, we are building two programs which solve two most common problems with dynamic programming, the Rod Cutting Problem and Longest Common Subsequence. This document will contain the explanation of the Longest Common Subsequence. First of all, what exactly is dynamic programming? Dynamic programming is a method that is often used by programmers to solve a really complex problem by breaking it down into collection of simpler subproblems, solving each subproblem just once, and storing it into a data structure, such as an array, list, table, etc. Therefore, the next time same subproblem comes up, we can look up the solution that we have already obtained in the data structure. This method really saves computation time.

The main goal of dynamic programming is obtaining an optimal solution. This type of problems is called optimization problems. Let’s say that we run a company and we want to obtain the maximum profit by selling our products. How do we manage our selling strategy in order to obtain the highest profit? Or how to get the largest or longest value of certain data? Dynamic programming will come in handy in this kind of situation. According to the book CLRS, there are four sequences of developing a dynamic programming solution:

1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Calculate the value of an optimal solution, typically using bottom-up method
4. Construct an optimal solution from calculated information

There are several approaches in order to obtain the optimal solution with dynamic programming. Basically, there are two properties of dynamic programming, **Optimal Substructures** and **Overlapping Subproblems**.

* **Optimal Substructures**

Like I have mentioned above that the goal of dynamic programming is to find optimal solutions, so the first step to solve or find optimal solutions by dynamic programming is to characterize the structure of an optimal solution. A given problem displays optimal substructures if optimal solution of the problem can be achieved by using optimal solutions of its subproblems.

* **Overlapping Subproblems**

Dynamic programming is used when solutions of same subproblems are needed all over again. Like I have mentioned above, dynamic programming stores calculated solutions in a table so we don’t have to recalculate it all over again. Therefore, dynamic programming is not very useful when there are no common subproblems because there’s no point storing the solutions if we don’t need it again. There are two different ways to store the values in dynamic programming, Memoization and Tabulation.

* 1. **Memoization (Top Down)**

A top down approach or fancy word for it is Memoization is an approach where we apply recursive function with a small modification that it looks into a table first before calculate the solution. Whenever we need the solution, we look into the table first. If the value already exists then we return that value, otherwise we calculate the value and store it in the table.

* 1. **Tabulation (Bottom Up)**

A tabulated approach or bottom up approach is an approach where we build a table for a given problem in bottom up fashion and returns the last entry from table. Tabulation method is relatively faster than memoization because we can directly access previous states from the table, meanwhile in memoization we deal with a lot of recursions.

**Longest Common Subsequence (LCS)**

Longest common subsequence basically is a problem finding the longest sequence which exists in both given input strings. This problem is a computer science problem, but also useful in another area of study, for example, Biology. In order to compare the DNA of two different organisms, we need to find the longest common subsequence from the DNA of both organisms. By applying this, we can determine the similarity of organisms and measure how closely related the two organisms are. We can see that this method is really useful in our daily life. How can we construct a longest common subsequence?

1. Characterizing a longest common subsequence
2. Solve recursively
3. Calculate the length of an LCS
4. Constructing an LCS

In this assignment, we focus more on calculate the length of an LCS. This is where the dynamic programming concept is really useful. Instead, comparing a character one-by-one, we are using a table or two-dimensional-array to store the comparison of each character. Therefore, we can easily track the LCS result later and we can compare each character more quickly with recursion like in the 2nd step. This function takes the time of θ(mn), where m and n are the length of each string. So the longer the length of each string, the longer it will take to solve this.

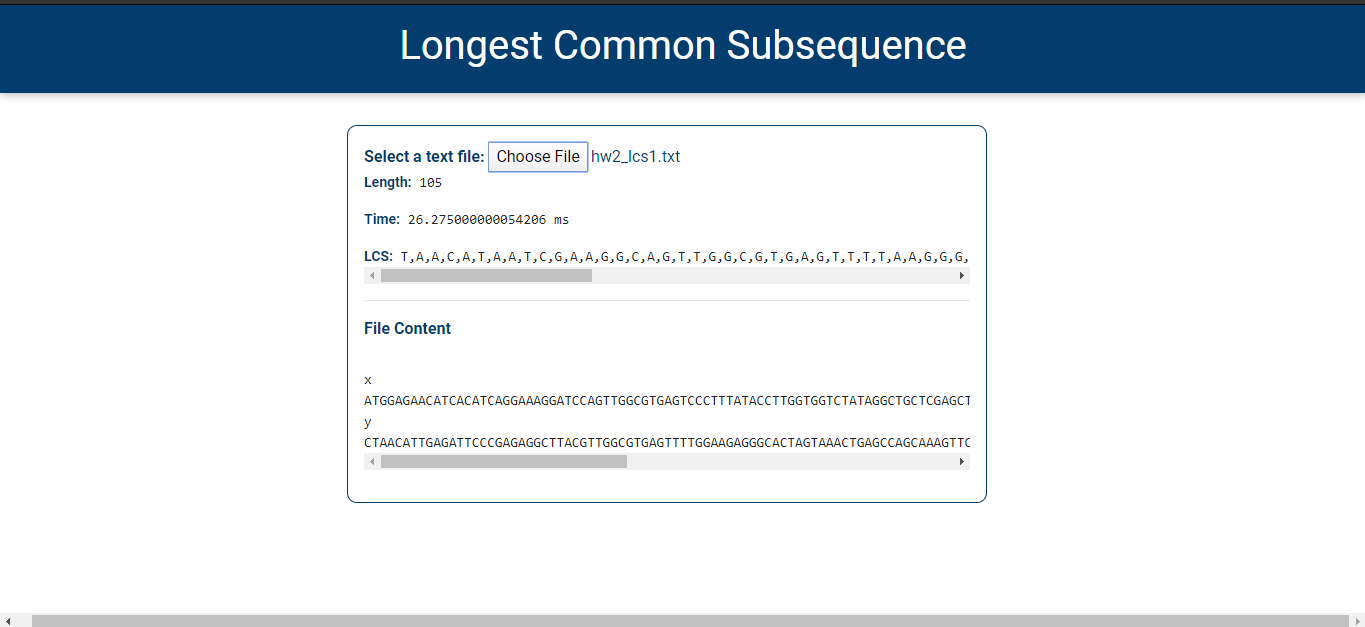
After we have gotten the length of the LCS, we can use the table that we construct while calculating the length of LCS. By tracking each character in the table with a pointer which has been constructed while calculating the length, we can achieve the LCS itself and print it out as the output. We can use recursive calls to print out the following LCS results. The construction of an LCS takes time of O(m+n), because it decrements the value of I and j at least one in each recursion process.

**Time Complexity:**

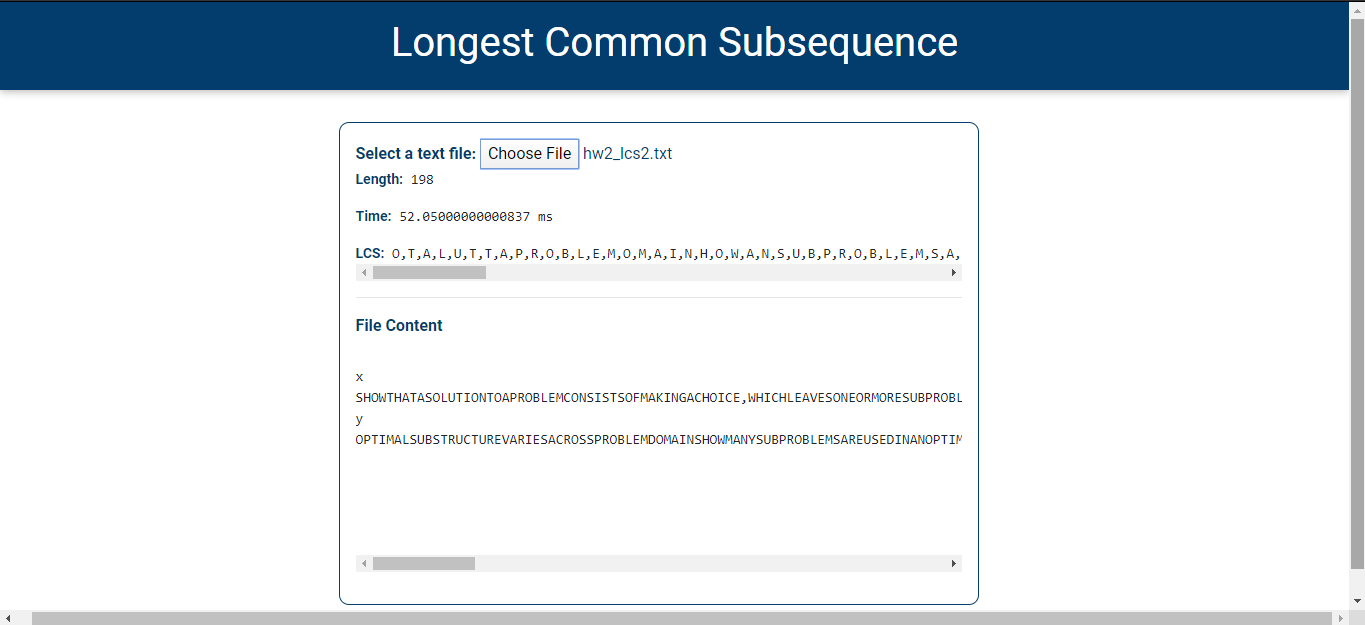
* Length of LCS: θ(mn)
* Print out LCS: O(m+n)

**Results**

LCS Case 1



LCS Case 2



We can clearly notice that the time taken in case 1 to solve the LCS from both strings is so much faster than case 2. It proves that the time complexity that we have seen above, where the time will grow as the length of the input strings get longer. It applies for calculating the length and printing out the LCS value.

**Implementation**

In this assignment, I’m implementing the Longest Common Subsequence problem through JavaScript code. Why do I use JavaScript? Because lately I’ve been taking JavaScript and web development online course and I really want to learn more about the web development process. The code is deployed via HTML file which can be viewed as a webpage and used by users. While working on this assignment, I thought of an idea on making this Longest Common Subsequence program as a webpage which can help users to find a LCS from sets of strings via online. This web application would be useful for people who works on Biology field, where they can use this webpage to access the data of a DNA.

Throughout the process of working on this assignment, I had an opportunity to implement the things that I have learned on the web development courses which are the DOM manipulation of JavaScript, the benefits of applying dynamic programming in our daily life, some skills on HTML and CSS, and some new JavaScript syntaxes.

**Conclusion**

From the analysis above about LCS and implementation that I have done, we can conclude that dynamic programming is one of the best ways to obtain an optimal solution. By applying this concept, we can actually save calculation time and memory space in order to obtaining the solution. I have learned that in Computer Science field, there is always a way to find a better method to solve a certain problem and we can reverse-engineer it in order to solve many problems. After finishing this assignment, I have gained a concrete knowledge of algorithm and programming from what I have learned during class into a real-world web application which can be useful for users who are facing the similar problem.