**The Rod Cutting Problem**

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 Rod Cutting Problem. 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? Dynamic programming will come 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.

**The Rod Cutting Problem**

Let’s say that we run a business of selling rods to the customer. One day, we have a piece of rod of some size that we want to sell and we want to cut into several parts then sell it in order to get the highest revenue from it. We already have a price list for each length of cut and the rod lengths are always integral number of inches. There is a possibility that cutting into smaller piece can obtain higher revenue than cutting into bigger piece or if the price is large enough, the optimal solution might be not cutting it at all. Therefore, the question is what is the best strategy that we can apply in order to get the maximum revenue with the correct choice of cuts?

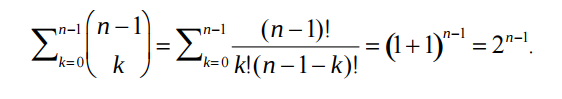
In this case, we will apply two methods to solve this problem. Brute force and dynamic programming bottom-up approach. The purpose of using two different approaches is to compare and inspect the time difference between the two and their efficiency.

* Brute Force

In brute force approach basically, we compare the price of each length of all different pieces recursively and returning the maximum value of each comparison. The maximum revenue r(n) on rod of size i = n, can be achieved by:

1. Starting with length of rod i = 1 and going up to size i = n. Repeat step 2 and 3
2. Deciding whether a cut at i has to be made or just leaving it uncut.
3. Repeating from 1 for the rest of the rod length (n-1)
4. Calculating r(n) in every case at the end
5. Maximum value among all calculated rod is returned as the maximum revenue.

We notice by applying this concept we already can achieve the maximum revenue from the certain length of rod we wish to sell. However, this method is extremely inefficient as we calculate every case of cuts without storing the solutions so we can use the answer of each calculated problem later to save computing time. The time complexity of this method is really slow. There are 2n-1 combinations to cut up a rod because we can choose to cut or not cut after each of the first n-1 inches. The number combinations can be derived from this calculation



Source: <http://www.cs.uml.edu/~kdaniels/courses/ALG_503_F12/DynamicRodCutting.pdf>

We can conclude that the time complexity of this method is O(2n), which is an exponential time complexity. This approach could be useful if we want to cut our rod in length of n = 1 or n = 2, but if we want to cut our rod in length of 30 or greater, the time will grow exponentially and will take much longer time to solve it. It is not a good idea to use this, but we still have hope and that hope is called dynamic programming which we will talk about in the next part.

* Bottom-Up Approach

This is the new hope of solving rod cutting in better and more efficient way. This is where dynamic programming concept will come in handy. The property of dynamic programming that we encounter is the overlapping subproblems by applying bottom up approach. This approach typically depends on some natural notion of the size of a subproblem, it is solving any particular subproblem depends only on solving smaller subproblems. The main difference of this approach with brute force is instead we count every single case of cut, we store the calculated result in a table. Hence, we can just look it up in the table later whenever we find the same problem.

1. Create an array r, which will play a part to store the solutions of the subproblems.
2. Assigning the first index of array r to zero
3. Solve each subproblem of the length of the rod. The approach used to solve a problem is similar to brute force approach which is finding the maximum value from the rod length of n-1. Instead of making a recursive call, we can find the solution in the array r[j-i].
4. Save the solution in the array r
5. Return r[n] which is the optimal value

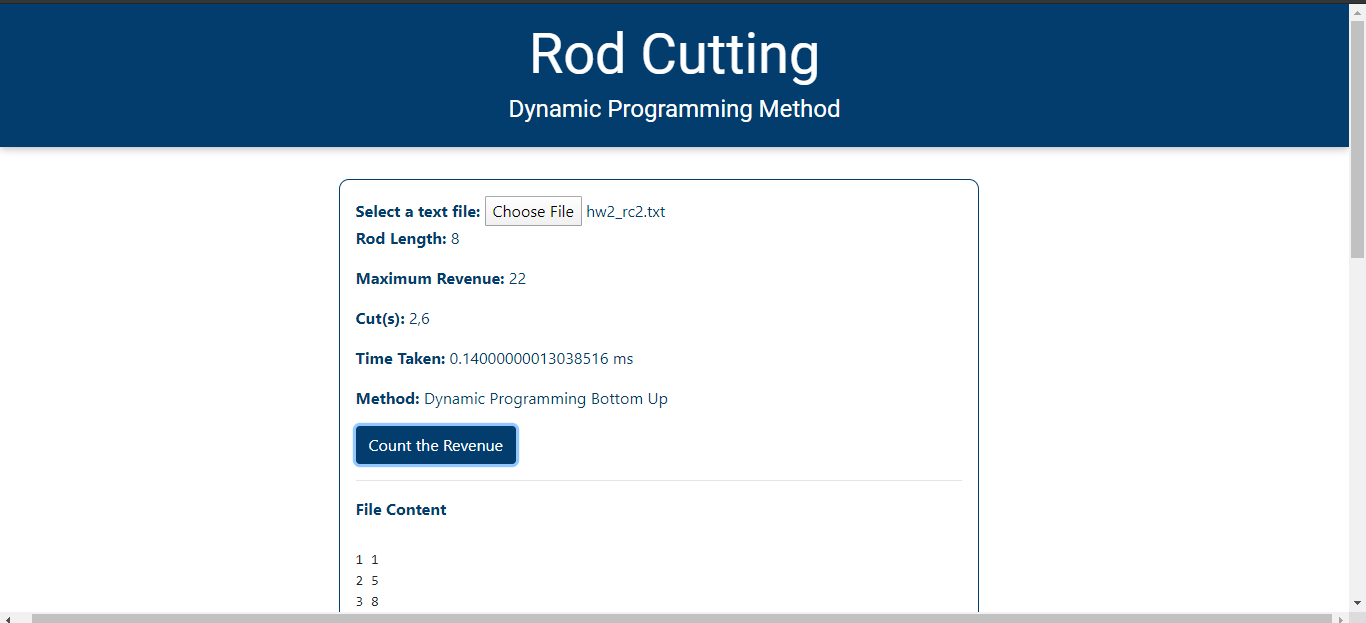
Due to its nested loop structure, this approach costs O(n2). Although this solution has a quadratic time, it’s still so much better than the time that we see in brute force approach. By applying dynamic programming, we notice a huge improvement of time taken to solve this. It is very useful in real life application where we can save time to solve a problem with dynamic programming.

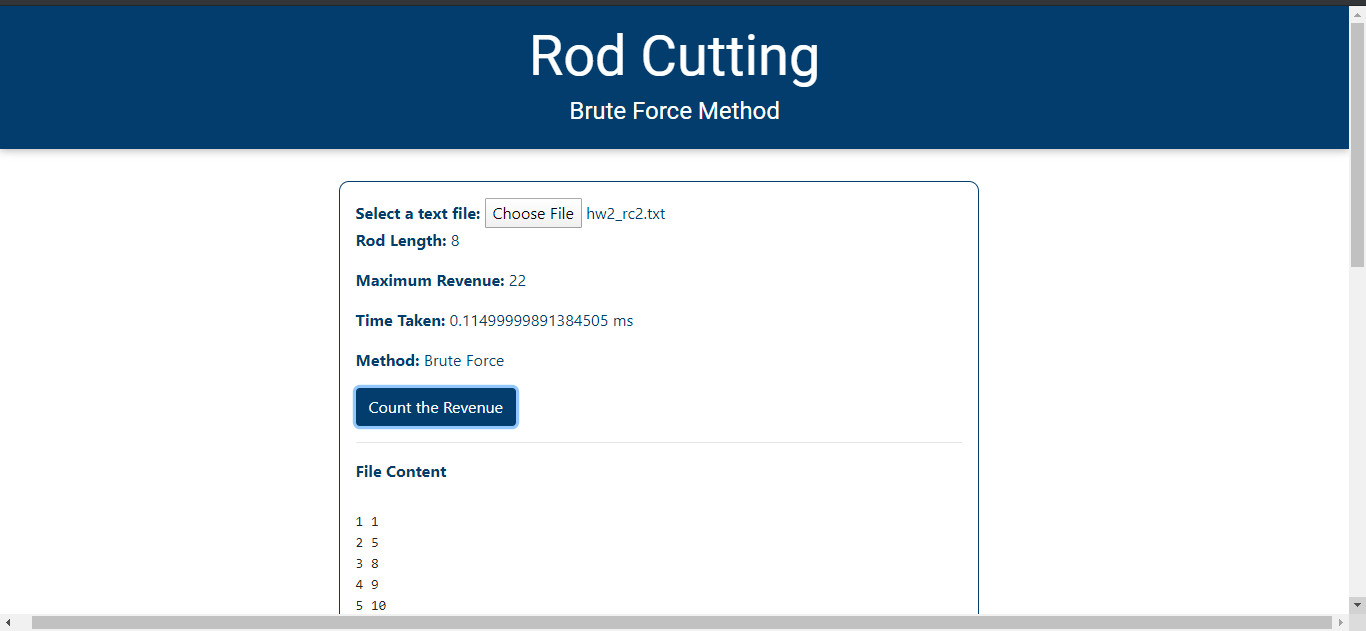
**Time Complexity**

* Brute Force: O(2n)
* Dynamic Programming Bottom Up: O(n2)

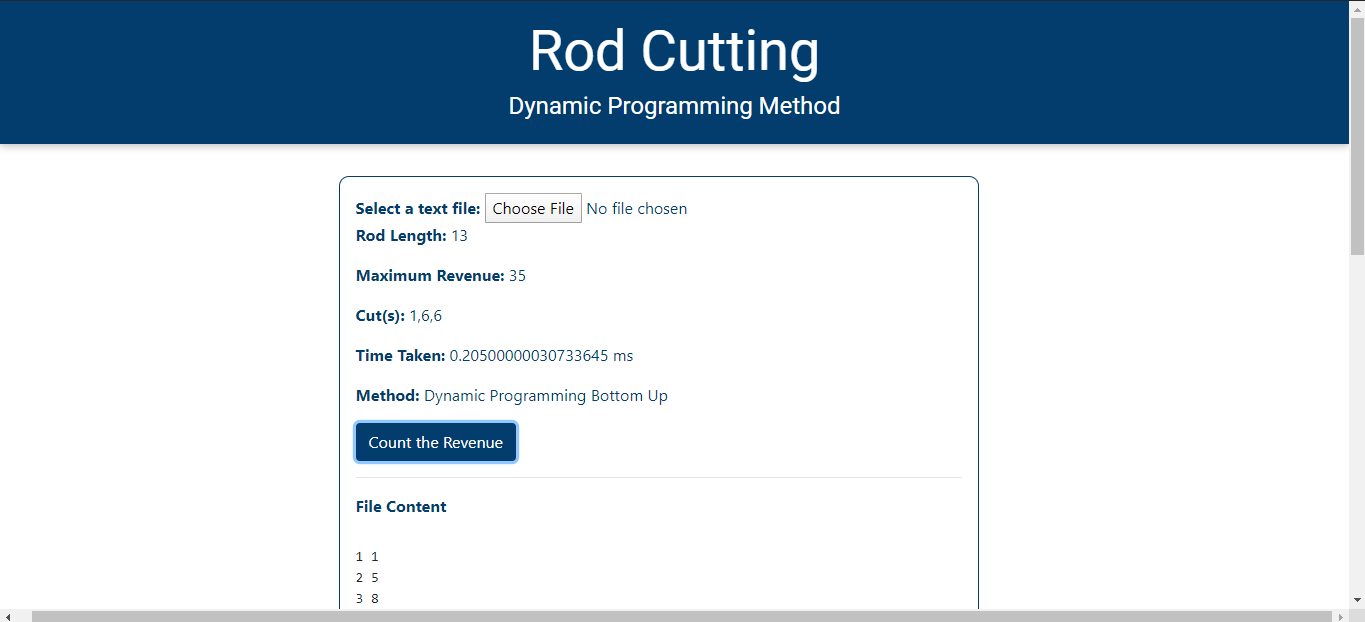
**Results**

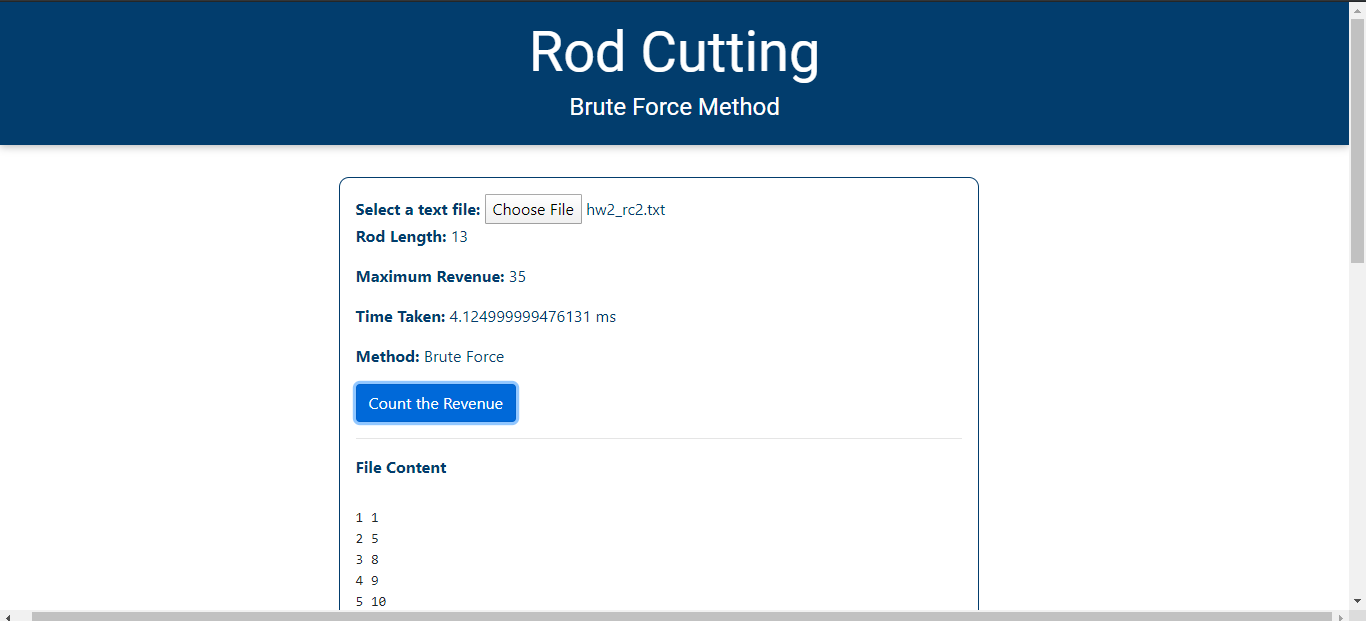
Rod Length: 8

****

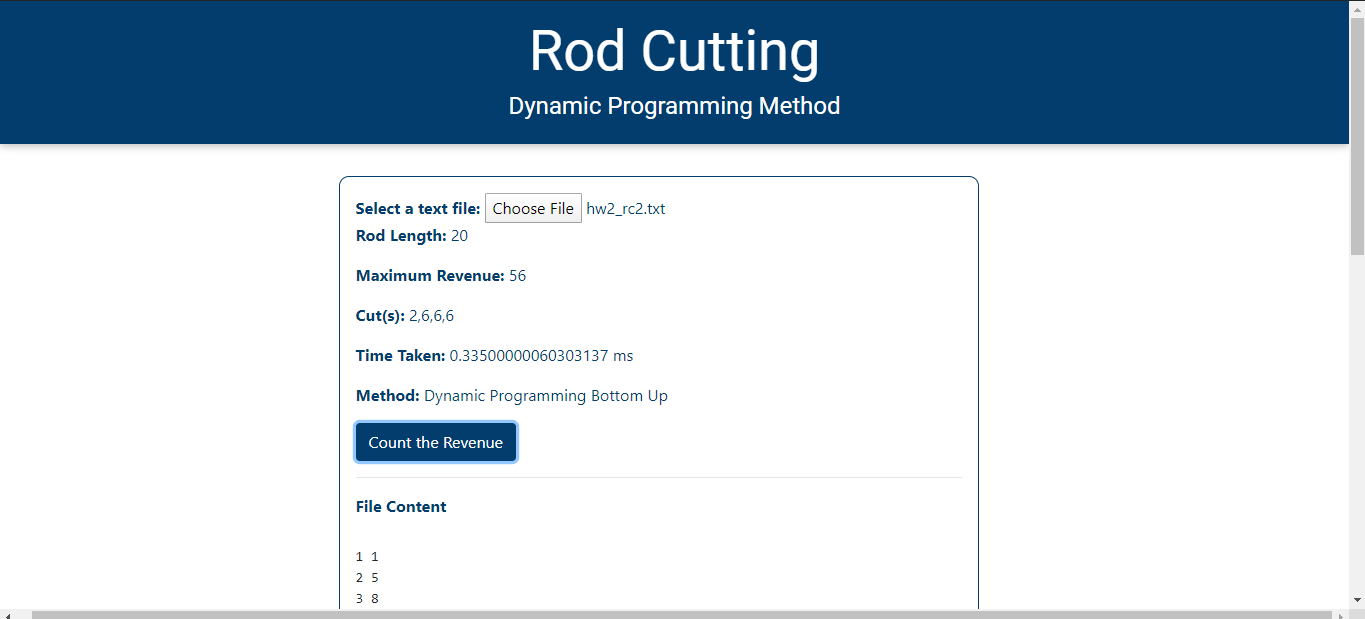


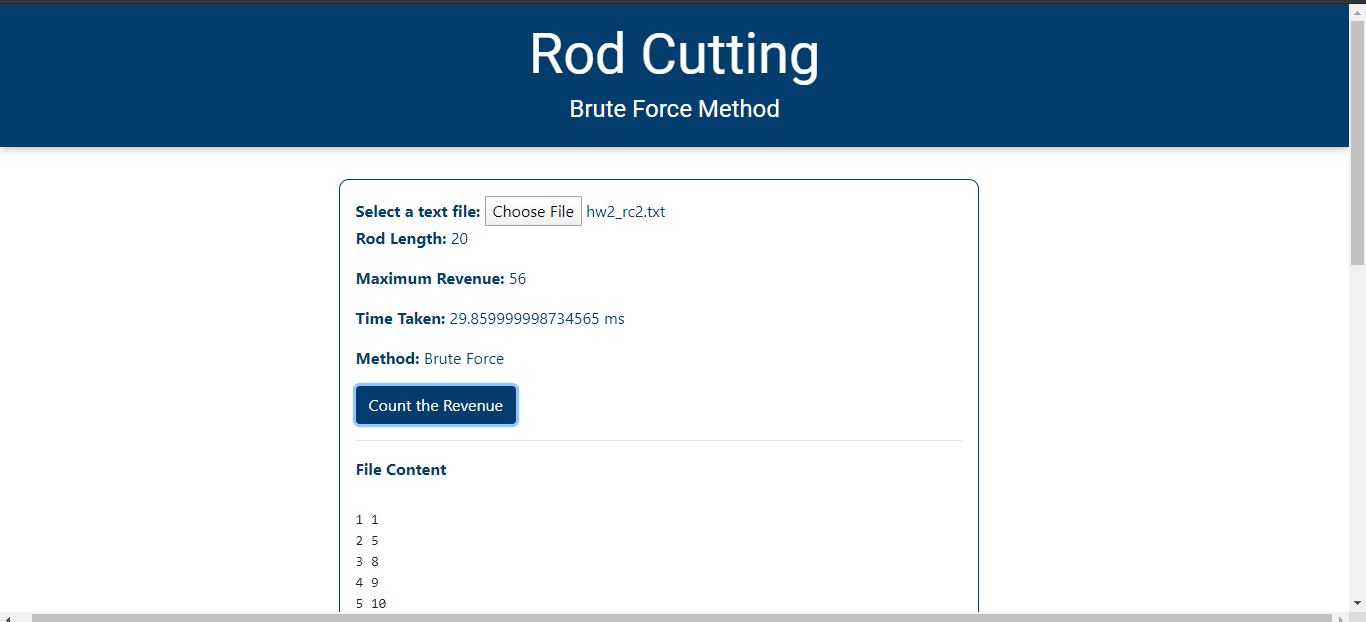
Rod Length: 13





Rod Length: 20





From the results above, we notice that when the user input is 8, the time taken in brute force method is similar and even faster to the time taken in dynamic programming approach. However, when the user input is 13, the time in brute force significantly. The time taken on brute force approach shows quite a great difference from the DP approach, while the time on DP approach is still similar from the first case. Finally, on the third case where user input is 20, the time taken in brute force approach takes a huge grow, it really takes a lot more time to solve it. Meanwhile, the time on DP approach still shows a quite constant difference from the first and second case.

It proves that by implying two different approaches, brute force and dynamic programming, really can change the efficiency for solving a problem.

**Implementation**

In this assignment, I’m implementing the Rod Cutting 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 Rod Cutting program as a webpage which can help users to find the maximum revenue of cutting a rod via online. This web application would be useful for people who works on business field, which can help them to find the maximum profit for their business. By implementing two approaches of solving the same problem, I can prove the benefit of dynamic programming to the users about its efficiency.

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 Rod Cutting problem and implementation that I have done, I 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. By comparing two different approaches of the same problem, we can obtain a vastly different execution time. This will be a problem if we have a millions of data to be calculated. 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.