**Programming Assignment 2 Report**

In this assignment the performance of 2 approaches for finding the Longest Repeated Subsequence of a string are compared:

1. Brute Force
2. Dynamic Programming

**Brute Force:**

In the Brute Force implementation, we start from the first element of the String and then compare it with rest of the characters of the string. If any one of the remaining characters match with the first character it is concatenated to a result string and the matching character is removed from the String at the position it was found. After the first iteration through the String the second character in the String is taken for comparison and this continues till the end of the String .The resulting Longest Repeated Subsequence is returned as the result String. The length is length of the subsequence.

**Dynamic Programming:**

In the Dynamic programming we use a bottom-up approach. Here a lookup table is created from the input String. The values in the lookup table are filled in the following manner. Whenever there is matching character in the string and the indices are not the same then a value is inserted in the lookup table if they are not matching we take the maximum of the values present at the (i, j-1) and (i-1, j). When a character is matching in the string then the value is incremented and stored in the table. The value present at the end of the table is the length of the longest repeated subsequence. For printing the subsequence we use the look table. We start from the bottom and check wherever there is decrement in the value in the lookup table. Where there is a change we take that index and take the character present at that index in the string and add the character to the result String. We return the final result string as the Longest Repeated Subsequence.

**Space and Time Complexity:**

For the time complexity for Brute Force method:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Characters** | 10 | 40 | 100 | 150 |
| **Time Taken** | 351959ns | 414825ns | 503776ns | 623947ns |

For the time complexity for Dynamic Programming method:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Characters** | 10 | 40 | 100 | 150 |
| **Time Taken** | 141981ns | 269921ns | 305985ns | 513756ns |

From the above values we could say that as the value of n increases Dynamic programming approach performs better than Brute Force method.

The space complexity of brute force constant as we do not store any other value other than the result, as the length of input string increases so does the length of the output string. But in the case of Dynamic Programming the space complexity increases as the input string length increases as the lookup table size also increases.

The time complexity of Brute Force method is O(2n) and Dynamic programming is O(n2).

The space complexity of Brute Force method is O(n) and Dynamic programming is O(n2).

The references are:

* <https://stackoverflow.com/questions/7569335/reverse-a-string-in-java>
* <https://www.tutorialspoint.com/java/java_strings.htm>
* https://www.tutorialspoint.com/java/util/java\_util\_arraylist.htm

Some of the concepts of Dynamic Programming have been taken from the below link:

* http://www.techiedelight.com/longest-repeated-subsequence-problem/