**Q) Explain how you implemented Brute Force and Dynamic Programming method**

**Ans: Brute Force**

* In Brute Force Approach, for a given problem, it tries all possible solutions and picks the desired solution.
* We find the longest repeated substring in a given text. Repeated substrings may not overlap. If more than one substring is repeated with the same length, print the first one.
* Here, I have used counter to check the number of occurrence’s of a particular subsequences, if the counter >=2, the longest repeated subsequence is generated.
* In brute force method, for each index pair[I, j], (i>0, j<=n) of the string. we have to count the length of the non-overlaping repeated substring starting from these 2 indices.
* And then get the first repeated substring with the longest length.

**Dynamic Programming**

* Can be solved by using either Memoization or Tabular method. But in this program, I have used Tabulation method.
* We find the longest repeated substring in a given text. Repeated substrings may not overlap. If more than one substring is repeated with the same length, print the first one.
* Here the aim is to find the longest repeated subsequence in a given string. i.e., how many times does a substring appear in a string without having to be next to each other.
* It follows Bottom Up Approach.
* We have a condition i != j which says indexes are not same.
* In 1st case, if there is a match, add 1 and take the values from the previous diagonal element.
* In 2nd case, if there is no match, take the maximum of either previous row element or column for a particular cell.

|  |  |  |
| --- | --- | --- |
| Input | Dynamic Programming | Brute Force |
| ATACTGGA | 3.08ms | 5.19ms |
| BANANA | 3.37ms | 6.09ms |

**Q) Document the time complexity and space complexity of your code that implements Longest Repeated Subsequence using brute force and dynamic programming. Briefly discuss which one is better and why?**

**Ans:**

**Brute Force**

* The running time in the worst case is O(n!).
* The space complexity is O(2^n).

**Dynamic Programming**

* The running time is O(n^2) . In some cases we can solve it in O(n^2 lgn).
* The space complexity is O(n^2).

Dynamic Programming is much faster than Brute force.

* Dynamic Programming works by reusing the computation results generated in previous steps and optimizes the result in bottom up approach and further uses this result to enhance and optimize the solution.
* Thus, it saves lot of repetitive work and saves computation time. Although I have used Tabulation method in this case which uses bottom up approach, Memoization method is faster and is used to store the intermediate results for faster processing and saves computation time and efforts.
* In the case of Brute force, which is also called as Backtracking Method, it tries all possible solutions and picks up a desired solution which is slower when compared to Dynamic Programming.
* Generally brute force is used in a scenario when a problem cannot be divided into smaller sub-problems or sub-problems may not be interrelated.

**References:**

1. Introduction to Algorithms- By H. Cormen
2. [www.geeksforgeeks.com](http://www.geeksforgeeks.com)
3. [www.stackoverflow.com](http://www.stackoverflow.com)
4. <https://www.youtube.com/watch?v=sSno9rV8Rhg>