**List of sources referred:**

Class Slides

**Code Structure :**

In the file LCS\_DP\_1.java, we have created a class named LCS\_DP\_1 which contains a main method: the program’s starting point. User input and file reading will be done in this method. A method named LCS is created which is responsible to find the length of the Longest Common Subsequence. In this method, we will iterate the 2nd String for every character of the 1st string. If both the characters are equal we will increment our length. Call to this method will be done from the main method.

In the file LCS\_DP\_2.java, we have created a class named LCS\_DP\_2 which contains a main method: the program’s starting point. User input and file reading will be done in this method. A method named LCS is created which is responsible to find the length of the Longest Increasing Common Subsequence. In this method, we will iterate the 2nd string for every character of the 1st string. If both the characters are equal we will check whether the previous character added to the longest increasing subsequence is lesser than or equal to the current character. If so, we will increment our length. Call to this method will be done from the main method.

**Time Complexity :**

LCS\_DP\_1: O(n\*m) where n and m are length of the strings respectively.

LCS\_DP\_2: O(n\*m) where n and m are length of the strings respectively.

**Experimental Results:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **STRING A** | **STRING B** | **LENGTH \_DP1** | **COMMON**  **SEQUENCE\_DP1** | **TIME\_DP1**  **(nano seconds)** | **LENGTH \_DP2** | **COMMON**  **SEQUENCE\_DP2** | **TIME\_DP2**  **(nano seconds)** |
| BAT | BOAT | 3 | BAT | 8711800 | 2 | BT | 8934000 |
| BAIRT | BRAID | 2 | BI | 9027000 | 2 | BR | 9805100 |
| PRINTED | PARTED | 5 | PRTED | 8127400 | 3 | ERT | 8756600 |
| MOTOR | METER | 3 | MTR | 8676800 | 2 | MT | 9364400 |
| ABCDE | FGHIJ | 0 | - | 6600 | 0 | - | 6800 |
| DHJIKL | ABEFOP | 0 | - | 11300 | 0 | - | 21500 |
| AAABBBC  CCTTY | GGHHAAA  BBBCCC | 9 | AAABBBCCC | 8640200 | 9 | AAABBBCCC | 7835500 |
| ABCDEFG  HIJ | JIHGFEDCB  A | 1 | A | 8759200 | 1 | A | 617500 |
| AABCDRE  T | ABBCCDDR  REETT | 7 | ABCDRET | 9060800 | 6 | ABCDRT | 7787900 |
| ADGKNPR TTWYY | ABBDEGHK  MMNPPER  SSTTUVW  WXYYZ | 12 | ADGKNPRTTWYY | 8394800 | 12 | ADGKNPRTTWYY | 7804400 |

**Conclusions:**

* Even though the time complexities are theoretically the same, based on the length of the two strings, the program for  finding the longest common subsequence in the ascending order takes more time compared to the latter in most of the cases.
* For strings of greater length and also if the length of the common substring returned, we observe that the time taken  for DP2 program is less compared to DP1.

Post completion of individual tasks, we have coordinated and shared the knowledge regarding the algorithm for each problem and generated the above report