Design & Analysis of Algorithms Project - 2

**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| Sl. No. | Content | Page no. |
| 1 | Description | 2 |
| 2 | Longest Common Subsequence Brute Force Approach | 3 |
| 3 | Longest Common Subsequence Dynamic Programming Approach | 6 |
| 4 | Time Complexity Comparison Longest Common | 8 |
| 5 | Subsequence Dynamic Programming Approach with b and c arrays | 11 |
| 6 | References | 12 |
| 7 | Honor Code | 13 |

# DESCRIPTION

Longest Common Subsequence **Brute Force approach,** Longest Common Subsequence **Dynamic Programming approach,** Longest Common Subsequence Dynamic Programming approach with b and c arraysalgorithms are implemented as part of this project. **Prudhvi Thumma** (Student ID: 1002033401) and **Rohith Kumar Boddu** (Student ID: 1002037081) are members of this team. We both contributed equally towards the implementation of algorithms, drawing comparisons, and documentation. File operations required for the project are mainly read operations. Given LCS1 and LCS2 text files consisting of two strings in each line are used to implement LCS\_BF.py and LCS\_DP.py and LCS\_DP\_CB.py reads the LCS2 text file to implement the dynamic programming version of LCS-LENGTH(X, Y).

How to run files:

**LCS\_BF.py, LCS\_DP.py, LCS\_DP\_CB.py** are files with code to find the longest common subsequence given in the text files.

Step 1: Install python and set up a runtime environment for python (check system environment variables in the path)

Step 2: Copy the files mentioned above to any directory

Step 3: Open the command prompt, navigate to your directory with files

Step 4: Enter the python “file\_name.py” command compile and run the given files. The output will be generated displaying sorted output and the time taken to complete the given sort operation

Modules to import

* **time**: This module provides various time-related functions.
  + **time()** function returns us the current time based on which I could calculate the difference in time to perform my sort algorithms.
* **read\_file** reads content from files, which are written after generating random numbers.
* **time()** gives the current time and is used to calculate the time elapsed to find the longest common subsequence between two strings
* **open()** opens a file with the given mode.

### Contributions towards the project:

File read operations: Prudhvi Thumma

Sort algorithms: Team

Report Documentation: Rohith Kumar Boddu

Time complexity comparison: Team

# LONGEST COMMON SUBSEQUENCE BRUTE FORCE APPROACH

Run LCS\_BF.py file in command prompt as directed, which provides us the output with the time as below

## 

## 

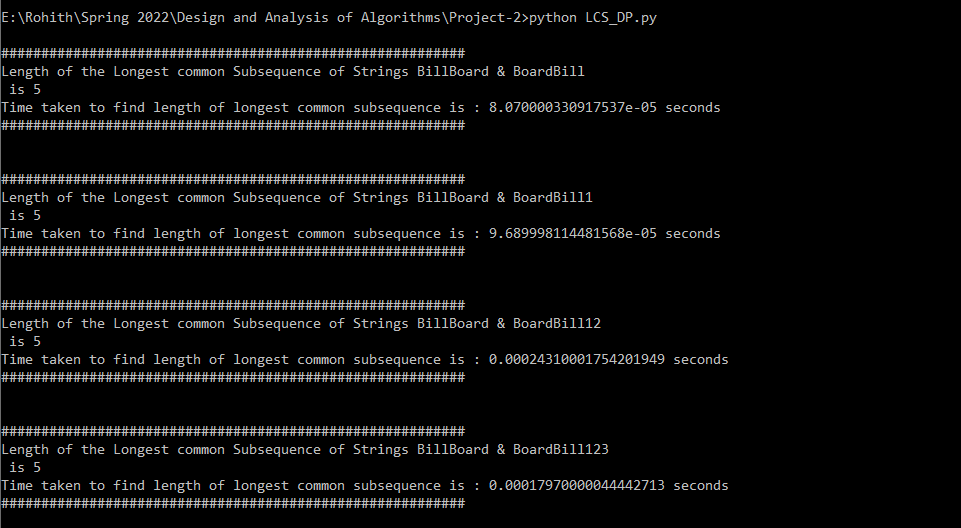
## 

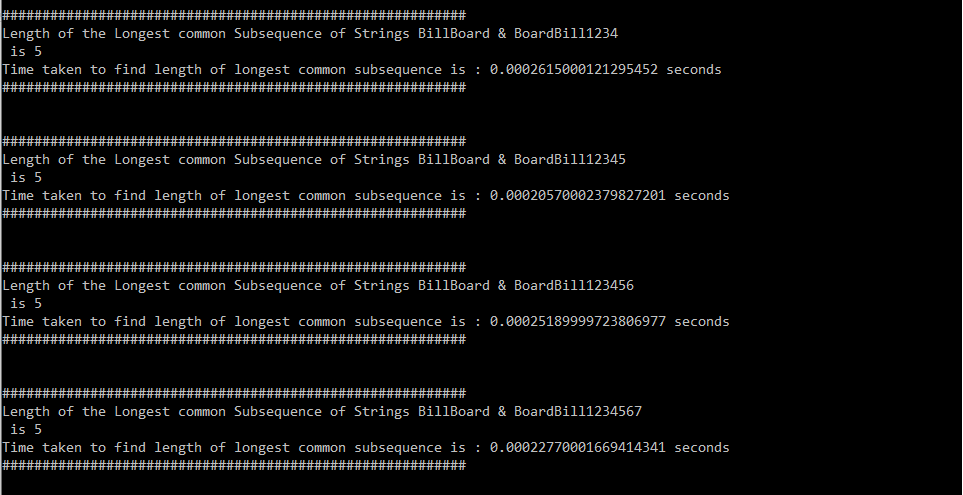
## 

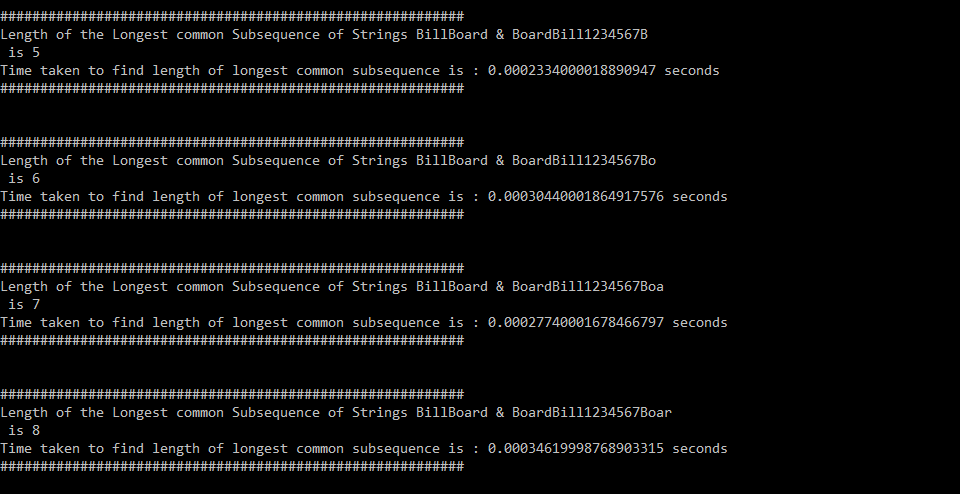
## 

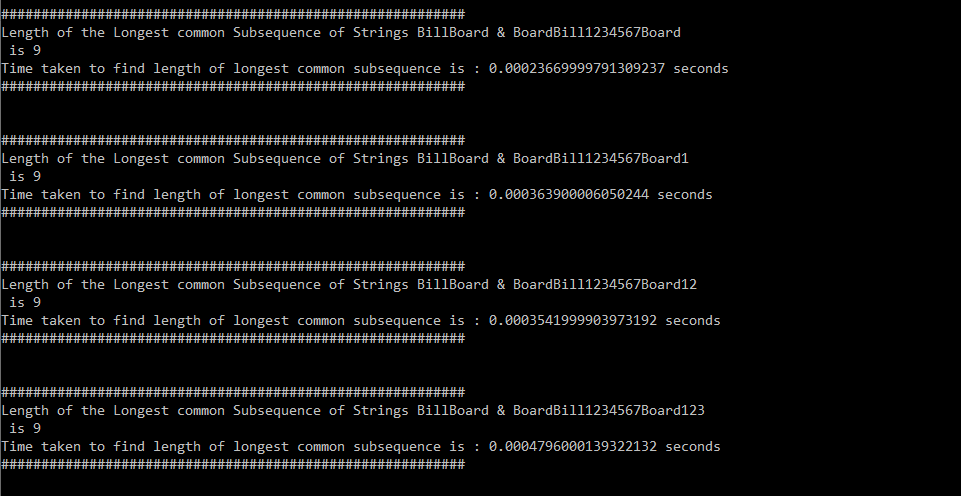
# LONGEST COMMON SUBSEQUENCE DYNAMIC PROGRAMMING APPROACH

Run LCS\_DP.py file in command prompt as directed, which provides us the output with the time below









# Time Complexity Comparison

We will analyze differences in theoretical time complexity and achieve experimental time complexity. Let's check the time complexity of individual sort algorithms

### Longest Common Subsequence Brute Force Approach Theoretical:

* Time Complexity: *O(n \* 2n)*

Theoretically, the time taken to find the longest common subsequence using the brute force approach will increase by order n \* 2n

### Longest Common Subsequence Dynamic Programming Approach Theoretical:

* Average Time Complexity: *O (n \* m)*

Theoretically, the time taken to find the longest common subsequence using the dynamic programming approach will increase by order n\*m

**Elapsed Time Comparison between Brute Force and Dynamic Programming Approach:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **String 1(X)** | **String 2(Y)** | **Length of LCS** | **Elapsed Time for LCS\_BF** | **Elapsed Time for LCS\_DP** |
| BillBoard | BoardBill | 5 | 0.0468907356262207 | 8.070000330917537e-05 |
| BillBoard | BoardBill1 | 5 | 0.08461117744445801 | 9.689998114481568e-05 |
| BillBoard | BoardBill12 | 5 | 0.11593794822692871 | 0.00024310001754201949 |
| BillBoard | BoardBill123 | 5 | 0.20049309730529785 | 0.00017970000044442713 |
| BillBoard | BoardBill1234 | 5 | 0.2539818286895752 | 0.0002615000121295452 |
| BillBoard | BoardBill12345 | 5 | 0.34770870208740234 | 0.00020570002379827201 |
| BillBoard | BoardBill123456 | 5 | 0.645578145980835 | 0.00025189999723806977 |
| BillBoard | BoardBill1234567 | 5 | 0.7031824588775635 | 0.00022770001669414341 |
| BillBoard | BoardBill1234567B | 5 | 1.06699538230896 | 0.0002334000018890947 |
| BillBoard | BoardBill1234567Bo | 6 | 1.4509756565093994 | 0.00030440001864917576 |
| BillBoard | BoardBill1234567Boa | 7 | 1.4875454902648926 | 0.00027740001678466797 |
| BillBoard | BoardBill1234567Boar | 8 | 1.3378477096557617 | 0.00034619998768903315 |
| BillBoard | BoardBill1234567Board | 9 | 0.12591052055358887 | 0.00023669999791309237 |
| BillBoard | BoardBill1234567Board1 | 9 | 0.47899389266967773 | 0.000363900006050244 |
| BillBoard | BoardBill1234567Board12 | 9 | 1.0069780349731445 | 0.0003541999903973192 |
| BillBoard | BoardBill1234567Board123 | 9 | 1.500481128692627 | 0.0004796000139322132 |
| BillBoard | BoardBill1234567Board1234 | 9 | 3.0752527713775635 | 0.00019190000602975488 |
| BillBoard | BoardBill1234567Board12345 | 9 | 5.238419532775879 | 0.00033760000951588154 |
| BillBoard | BoardBill1234567Board123456 | 9 | 8.206242322921753 | 0.00038240000139921904 |
| BillBoard | BoardBill1234567Board1234567 | 9 | 9.06346607208252 | 0.0004285000031813979 |

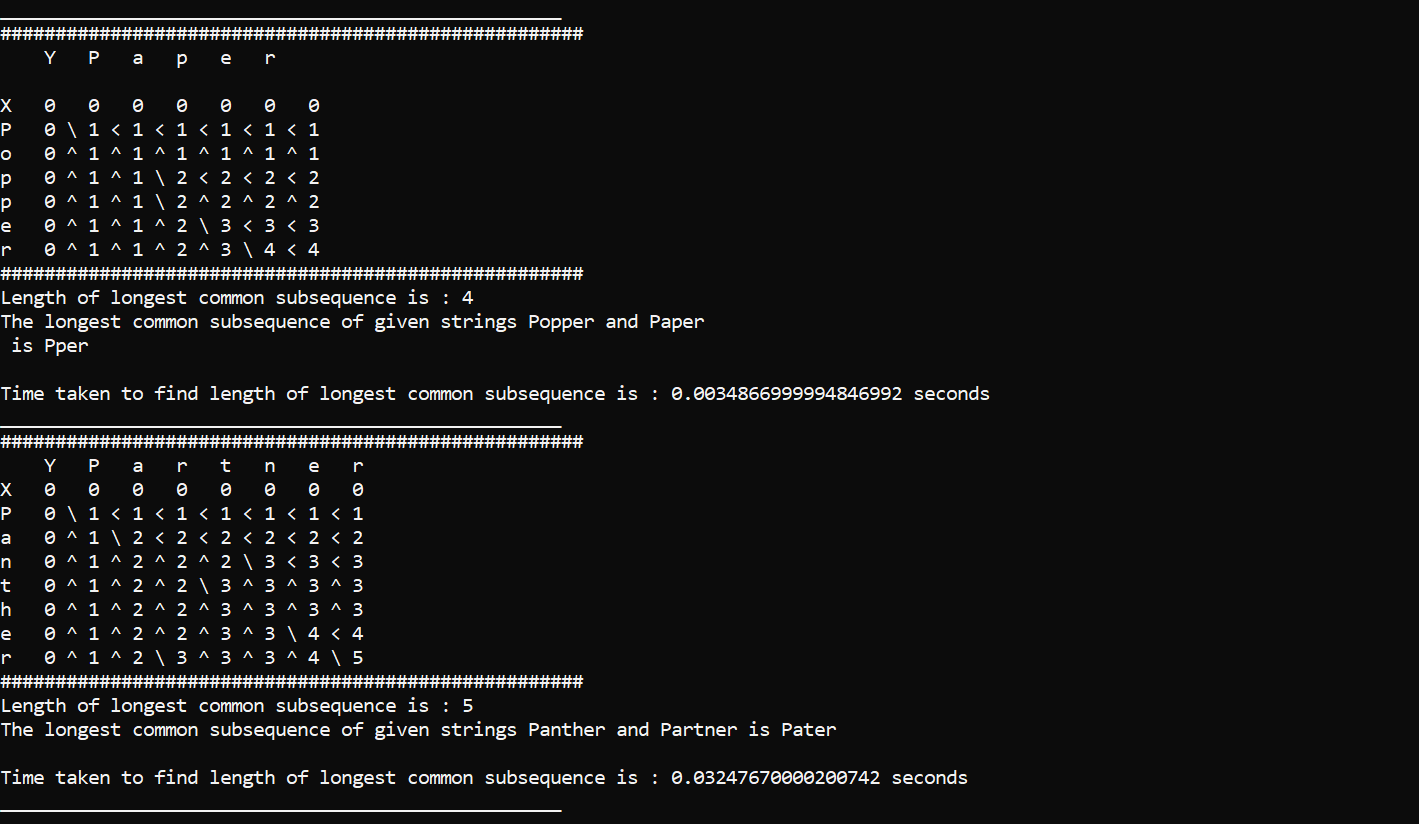
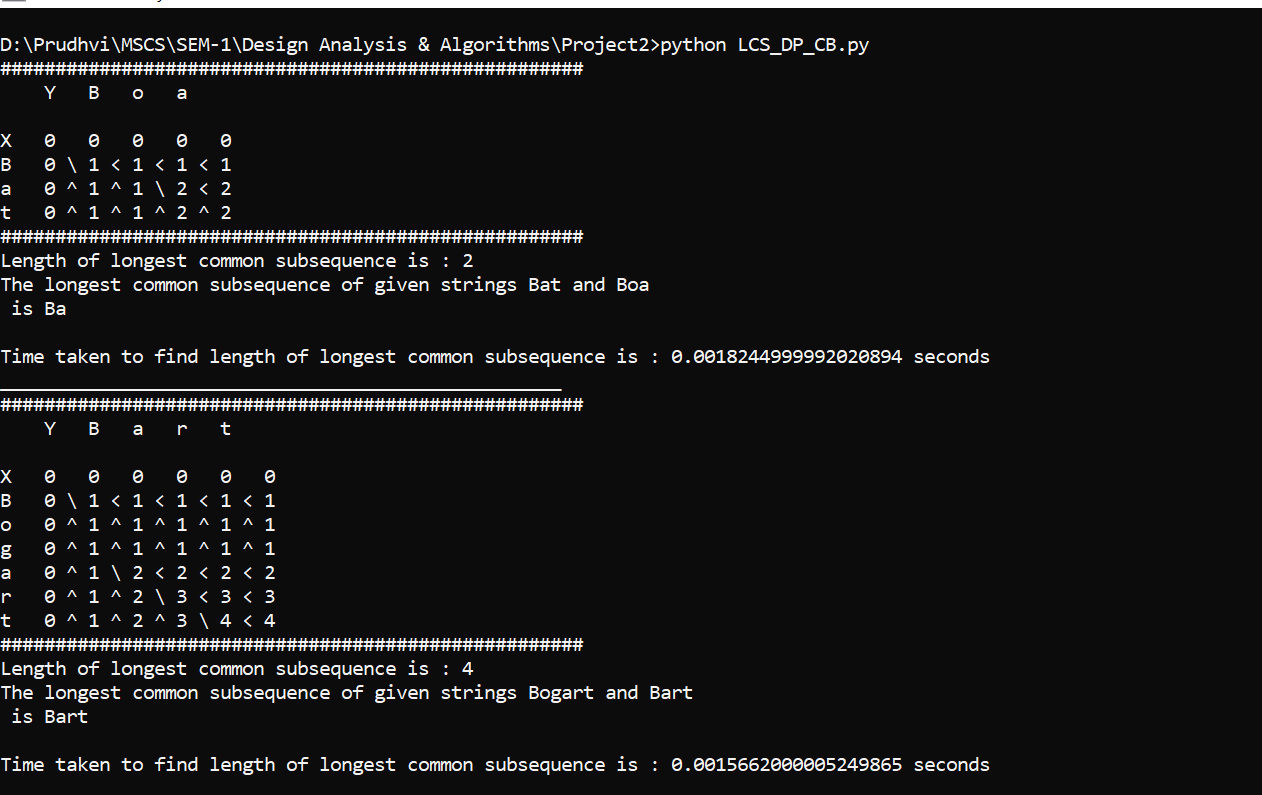
In the Brute force approach, all the subsequences are enumerated and tested which are the subsequences of Y and the longest one is picked.

In the Dynamic programming approach, we find the optimal solution using a recursive algorithm.

The above graph depicts the time difference between LCS\_BF and LCS\_DP algorithms for the given inputs.

# LONGEST COMMON SUBSEQUENCE DYNAMIC PROGRAMMING APPROACH WITH b AND c ARRAYS

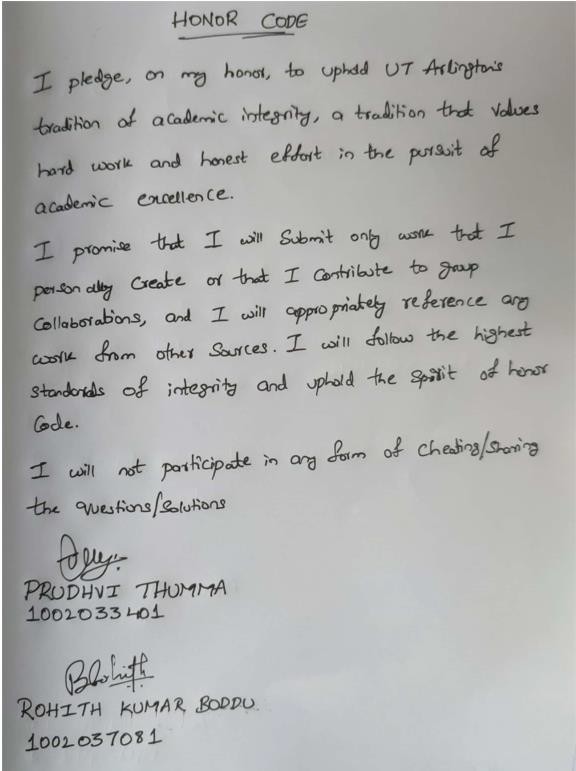
Run LCS\_DP\_CB.py file in command prompt as directed, which provides us the output with the time below



# REFERENCES

1. <https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf>
2. https://stackoverflow.com/questions/19095796/how-to-print-a-single-backslash
3. <https://docs.python.org/3/library/time.html>

# HONOR CODE



28th April 2022