**SUMMARY**

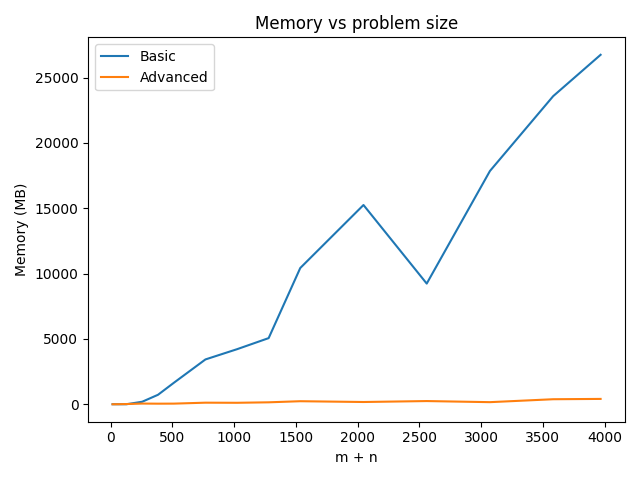
## USC ID/s: 4644918606, 8603483937, 3532632771

| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| --- | --- | --- | --- | --- |
| 16 | 0.06866 | 0.2971 | 4 | 4 |
| 64 | 0.6788 | 1.5450 | 4 | 8 |
| 128 | 2.4662 | 4.5869 | 4 | 20 |
| 256 | 9.6190 | 16.7041 | 196 | 56 |
| 384 | 20.9711 | 37.3089 | 728 | 48 |
| 512 | 37.2301 | 66.2470 | 1644 | 52 |
| 768 | 88.0949 | 146.2052 | 3432 | 124 |
| 1024 | 159.6570 | 273.8850 | 4216 | 116 |
| 1280 | 263.7820 | 413.5947 | 5064 | 152 |
| 1536 | 375.7129 | 587.7800 | 10428 | 236 |
| 2048 | 665.5622 | 1059.5851 | 15248 | 176 |
| 2560 | 1103.3220 | 1614.1851 | 9236 | 248 |
| 3072 | 1467.6480 | 2374.6121 | 17852 | 164 |
| 3584 | 2262.5839 | 3288.9881 | 23576 | 388 |
| 3968 | 2520.0982 | 4044.7841 | 26740 | 412 |

## Datapoints

## Insights

### Graph1 – Memory vs Problem Size (M+N)



#### Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)

Basic: Polynomial

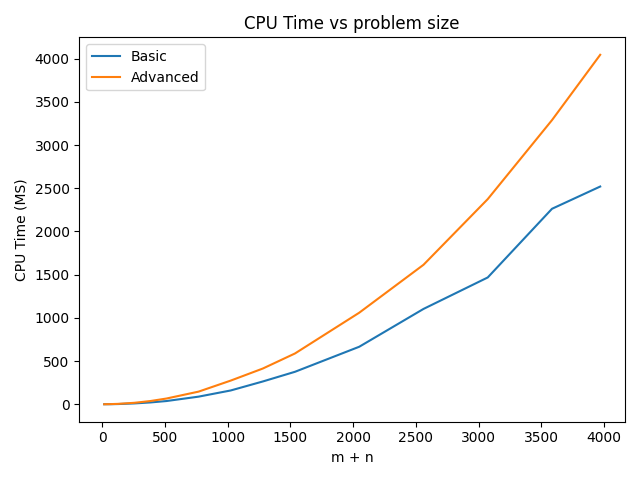
Efficient: Linear

#### Explanation:

The basic dynamic programming algorithm uses a m\*n array to memorize minimum cost of an alignment between two substrings with different sizes in order to find the optimal alignment between string 1 with size of m and string 2 with size of n. The space complexity is O(mn), which is polynomial with respect to the input size.

The efficient algorithm costs O(m) memory because it has only two columns each with (m+n) data.

### Graph2 – Time vs Problem Size (M+N)



#### Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)

Basic: Polynomial

Efficient: Polynomial

#### Explanation:

The basic dynamic programming algorithm traverses the m\*n array in the bottom-up pass to fill in all cost values of the alignments between two substrings, and then go through a top-down pass traversing the array again to find the alignment with minimum cost. The time complexity is O(mn), which is polynomial with respect to the input size.

The efficient algorithm takes O(mn) time, the efficient method takes about twice more time than the basic dynamic programming method as the efficient method needs to recursively compute the sub-problems.

## Contribution

(Please mention what each member did if you think everyone in the group does not have an equal contribution, otherwise, write “Equal Contribution”)

<4644918606, 8603483937, 3532632771>: Equal Contribution