**SUMMARY**

## USC ID/s: 1818275093, 7105760667, 3031225968

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 0.997 | 0.000 | 14628 | 14592 |
| 64 | 0.997 | 1.994 | 14676 | 14672 |
| 128 | 2.991 | 6.952 | 14812 | 14672 |
| 256 | 12.023 | 23.935 | 15040 | 14668 |
| 384 | 27.926 | 50.838 | 14944 | 14684 |
| 512 | 47.847 | 98.737 | 15292 | 14744 |
| 768 | 135.428 | 253.323 | 15080 | 14852 |
| 1024 | 262.298 | 428.565 | 15704 | 14932 |
| 1280 | 346.074 | 665.849 | 15124 | 14876 |
| 1536 | 529.981 | 889.007 | 15504 | 15060 |
| 2048 | 937.953 | 1642.088 | 15764 | 14992 |
| 2560 | 1634.571 | 2547.263 | 16168 | 15156 |
| 3072 | 2071.046 | 3597.144 | 16492 | 15244 |
| 3584 | 2877.036 | 4960.328 | 16652 | 15376 |
| 3968 | 3601.247 | 5765.041 | 16588 | 15376 |

## Datapoints

## Insights

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

Chart, line chart

Description automatically generated

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

Basic: Linear

Efficient: Linear

#### Explanation:

The basic algorithm initializes and fills a matrix of size O(mn), meaning it takes O(mn) space

Each recursive substring will have at least one character less than its superstring, so the maximum size of the call stack is O(max(m, n)). Each level compares max(m, n) alignments, but uses bottom-up optimization to cache only two columns of the dp matrix (each being a single index of string x being compared to many values of indices y) at a time. Its runspace is thus O(2\*max(m, n)), or O(max(m, n)) asymptotically, less than that of the basic algorithm

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

Chart, line chart

Description automatically generated

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

Basic: Polynomial

Efficient: Polynomial

#### Explanation:

The basic algorithm has to initialize a table of size O(mn) and cross it in O(m + n) time, resulting in a runtime of O(mn) + O(m + n)

The highest level of the space-efficient algorithm performs O(2mn) operations on its highest level (comparing XL to all of Y and XR to all of Y). This algorithm is top-heavy, so the runtime of its highest level determines the runtime of the whole algorithm, O(2mn), greater than that of the basic algorithm.

## Contribution

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

<USC ID/s>: <Equal Contribution>

<1818275093, 7105760667, 3031225968>: <Equal Contribution>