**SUMMARY**

## USC ID/s:

## Swapnil Arya (6456245219)

1. Yash Dani (5341480442)
2. Geet Shingi (6015146447)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 0.067 | 0.143 | 13132 | 13192 |
| 64 | 0.719 | 0.958 | 13132 | 13396 |
| 128 | 2.623 | 3.176 | 13656 | 13432 |
| 256 | 10.618 | 11.140 | 14976 | 13280 |
| 384 | 23.906 | 24.086 | 16124 | 13172 |
| 512 | 48.866 | 42.803 | 15808 | 13264 |
| 768 | 100.237 | 92.874 | 17468 | 13380 |
| 1024 | 206.521 | 164.814 | 19188 | 13264 |
| 1280 | 326.094 | 247.368 | 18712 | 13588 |
| 1536 | 416.700 | 356.807 | 28300 | 13540 |
| 2048 | 782.342 | 640.295 | 35464 | 13572 |
| 2560 | 1117.002 | 988.946 | 36288 | 13652 |
| 3072 | 1576.419 | 1396.259 | 56576 | 13408 |
| 3584 | 2199.690 | 1975.909 | 55196 | 13920 |
| 3968 | 2782.845 | 2453.460 | 79836 | 13960 |

## Datapoints

## Insights

1. The memory vs problem size graph below depicts that the efficient version of the algorithm consumes significantly less memory than the basic version as the input size grows. While the efficient algorithm shows a linear increase in memory, the basic version shows a polynomial increase in memory with input size.

2. The time vs problem size graph below shows very similar performance of both the algorithms as the input size grows. The nature of the graph for both the algorithm is Polynomial.

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

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

Basic: Polynomial

Efficient: Linear

#### Explanation:

The basic algorithm stores the entire memoized matrix in the memory, which should increase as the input size increases, thus warranting an increase in the memory usage.

The efficient algorithm has the space complexity as O(n), which is the length of string2. To calculate the break point, it only uses memory of size n\*2.

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

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

Basic: Polynomial

Efficient: Polynomial

#### Explanation:

The basic algorithm will take more time as the input size increases as loops are used to fill the memoized matrix. The loops continue until whole of string1 and string2 is processed. Thus the time complexity is O(m\*n), where m and n is the size of both the input strings.

The efficient algorithm continues until whole of the input strings are processed. The processing includes breaking of the input string into 2 and calculating the break points in the either half. Thus the time complexity is O(m\*n), where m and n is the size of the both the input strings.

## 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>

6456245219: Equal Contribution

5341480442: Equal Contribution

6015146447: Equal Contribution