**SUMMARY**

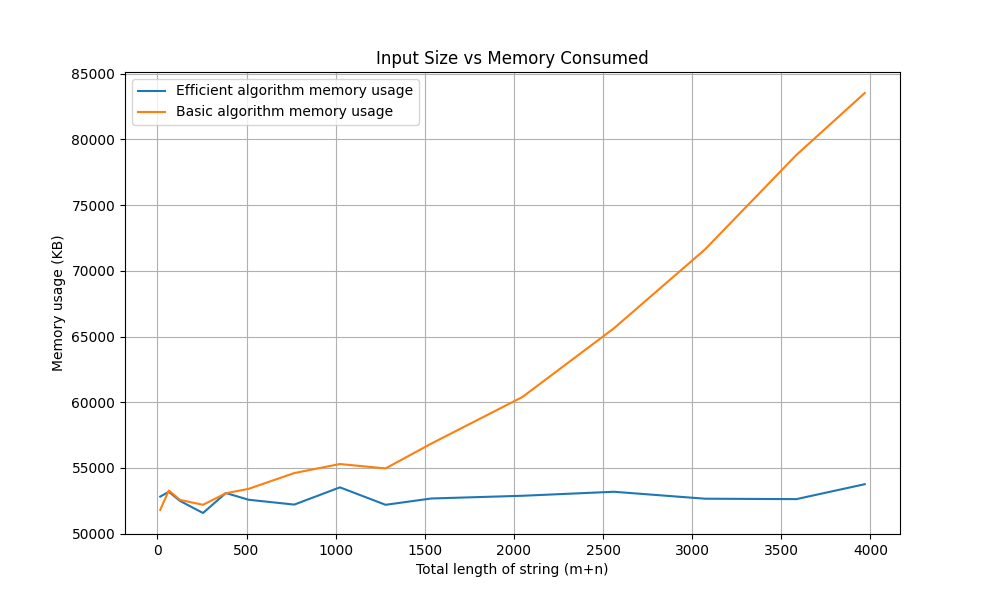
**USC ID/s: 6494261668, 6104797766, 1198271852**

# Datapoints:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 0.19311904907226600 | 0.3619194030761720 | 51808 | 52816 |
| 64 | 1.1949539184570300 | 2.5169849395752000 | 53280 | 53184 |
| 128 | 4.092931747436520 | 9.493112564086910 | 52560 | 52480 |
| 256 | 15.527009963989300 | 37.24479675292970 | 52192 | 51568 |
| 384 | 35.37416458129880 | 87.18681335449220 | 53072 | 53088 |
| 512 | 61.556100845336900 | 153.5470485687260 | 53408 | 52576 |
| 768 | 142.36807823181200 | 348.70004653930700 | 54608 | 52208 |
| 1024 | 259.28807258606000 | 608.7350845336910 | 55296 | 53520 |
| 1280 | 391.5109634399410 | 946.2659358978270 | 54960 | 52192 |
| 1536 | 568.5429573059080 | 1376.0950565338100 | 56848 | 52672 |
| 2048 | 1007.9638957977300 | 2395.1637744903600 | 60400 | 52880 |
| 2560 | 1588.2270336151100 | 3724.602222442630 | 65616 | 53184 |
| 3072 | 2198.4591484069800 | 5230.1530838012700 | 71632 | 52656 |
| 3584 | 3079.756021499630 | 7189.682960510250 | 78816 | 52624 |
| 3968 | 3785.0418090820300 | 8940.824031829830 | 83536 | 53760 |

# Insights:

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

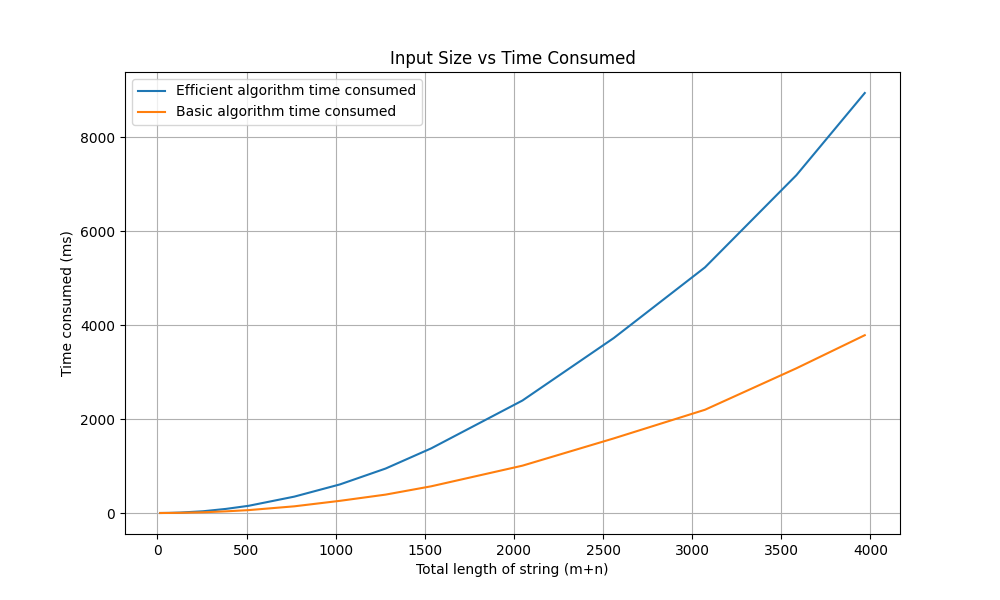


### Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)

* + - Basic: Polynomial
    - Efficient: Linear

#### Explanation: Basic algorithm has space complexity of O(mn) whereas efficient algorithm has space complexity of O(min(m,n)) where m and n are the lengths of input string. Thus, memory consumption by the efficient algorithm is significantly less compared to the basic algorithm.

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



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

* + - Basic: Polynomial
    - Efficient: Polynomial
    - Explanation: Both efficient and basic algorithm take O(mn) time. However, the number of computations done in efficient algorithm are twice than that of the basic one. In addition to that we also need to take into account the time required to manage the recursive calls made in our Divide and Conquer approach.

# Contribution:

* 6494261668: Equal Contribution
* 6104797766: Equal Contribution
* 1198271852: Equal Contribution