**SUMMARY**

## USC ID/s:

2448443089, 1031572511

## Datapoints

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 0.04 | 0.109196 | 14272 | 14284 |
| 64 | 0.351 | 0.748158 | 14336 | 14420 |
| 128 | 1.652 | 3.051996 | 14532 | 14400 |
| 256 | 6.4487 | 10.4599 | 15060 | 14456 |
| 384 | 12.938 | 22.043228 | 15984 | 14376 |
| 512 | 20.354 | 38.407087 | 17176 | 14384 |
| 768 | 50.1482 | 85.232735 | 20500 | 14568 |
| 1024 | 91.5852 | 153.283119 | 25068 | 14528 |
| 1280 | 143.049 | 230.59392 | 30972 | 14432 |
| 1536 | 216.5642 | 326.365948 | 38080 | 14656 |
| 2048 | 389.9429 | 579.987049 | 56632 | 14484 |
| 2560 | 575.454 | 944.667816 | 79904 | 14676 |
| 3072 | 822.052 | 1309.22699 | 108776 | 14668 |
| 3584 | 1138.2751 | 1811.086893 | 134508 | 14800 |
| 3968 | 1441.1550 | 2168.390751 | 163280 | 14904 |

## Insights

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

A graph with a blue line

Description automatically generated

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

Basic: Polynomial

Efficient: Linear

#### Explanation:

### The basic algorithm increases memory usage polynomially with input size because it builds a full dynamic programming (DP) table of size 𝑚×𝑛, leading to O(mn) space complexity and memory usage that can reach tens or hundreds of megabytes for large inputs. In contrast, the efficient algorithm applies a divide-and-conquer approach that only computes and stores partial score rows during recursion, reducing its space complexity to O(n). As a result, its memory usage grows much more slowly and typically remains within a few megabytes, even as the input size scales significantly.

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

A graph with a line graph

Description automatically generated with medium confidence

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

Basic: Polynomial

Efficient: Polynomial

#### Explanation:

The CPU time for both basic and efficient algorithms shows a polynomial increasing as the problem size increases. They both takes O(mn) time complexity, but the efficient algorithm based on divide and conquer does not reduce its time complexity due to more recursive calls. So, more calculations result in higher CPU time for efficient algorithm, making it slower than the basic algorithm when dealing with large inputs.

## 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>

2448443089: Equal Contribution

1031572511: Equal Contribution