**SUMMARY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 0.999689 | 0.102519 | 15724 | 15824 |
| 64 | 0 | 0.997066 | 15736 | 15812 |
| 128 | 1.998186 | 4.146575 | 15884 | 15808 |
| 256 | 8.063316 | 16.112327 | 16372 | 15888 |
| 384 | 16.999244 | 35.502195 | 17208 | 15864 |
| 512 | 32.057285 | 64.319490 | 18352 | 15840 |
| 768 | 74.018716 | 144.399404 | 21696 | 15888 |
| 1024 | 139.220714 | 334.811925 | 26460 | 15792 |
| 1280 | 214.568376 | 453.048229 | 32260 | 15876 |
| 1536 | 311.930656 | 609.175682 | 40012 | 16004 |
| 2048 | 595.869541 | 1094.847202 | 58772 | 15964 |
| 2560 | 967.764139 | 1708.559989 | 82692 | 16160 |
| 3072 | 1268.923044 | 3028.587818 | 110952 | 16048 |
| 3584 | 1711.432933 | 3412.905216 | 147868 | 16216 |
| 3968 | 2143.638849 | 4129.018068 | 174180 | 16128 |

## Datapoints

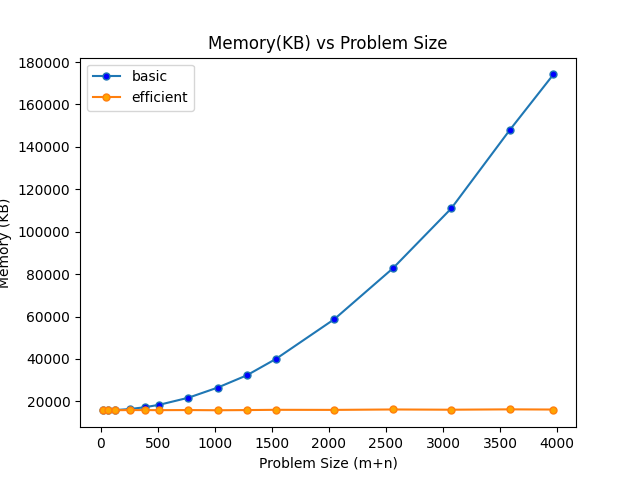
## Insights

From above table we can say that the time taken to solve the problem using dynamic programming (basic) is increasing in exponential way along with the increase in problem size. And the memory used is also increasing in exponential manner as the problem size is increased.

Whereas, the dynamic program along with divide and conquer (efficient) uses constant memory and does not increase as the problem size increases. And the time taken in this also increases exponentially as the problem size increases.

So, we can say that by making the basic algorithm efficient, we mean in the way it uses memory and not the time. Because basic algorithm will take a large space as the problem size increases and efficient version of it will keep the memory constant with the increase in problem size which makes the program memory efficient.

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



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

Basic: Exponential

Efficient: Constant

#### Explanation:

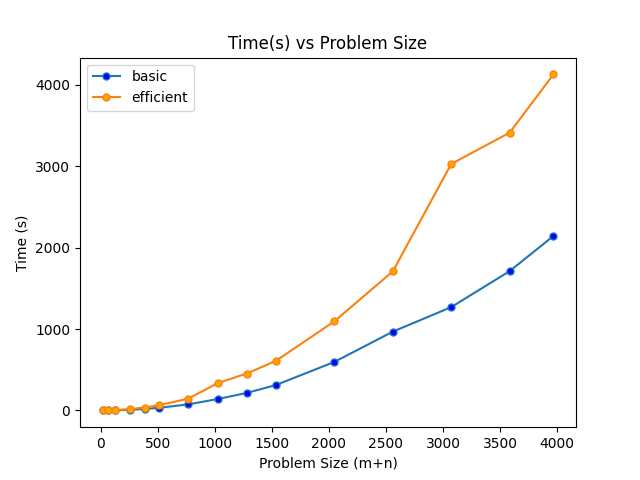
In above graph, the x-axis represents the Problem size (m + n) where m and n are the length of two strings. The y-axis represents memory used by the program to execute in KB. And the blue line represents the plot of basic algorithm and the orange line represents the plot for efficient algorithm.

So, from the graph plotted by the program, we can say that the memory used by the basic version of the program increases exponentially as the problem size increases.

And the memory used by the efficient algorithm stays constant even with the increase in the problem size.

So, from these data, we can conclude that the efficient algorithm will use less memory when the problem size increases which is an important factor to look at while designing an algorithm.

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



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

Basic: Exponential

Efficient: Exponential

#### Explanation:

In above graph, the x-axis represents the Problem size (m + n) where m and n are the length of two strings. The y-axis represents time taken by the program to execute in milliseconds. And the blue line represents the plot of basic algorithm and the orange line represents the plot for efficient algorithm.

So, from the graph plotted above, we can say that the time taken by the basic algorithm as well as time taken by efficient algorithm is increasing in exponential manner with the increase in problem size.

Actually looking carefully at the graph, we can know that the time taken by efficient algorithm is little bit more that the time taken by basic algorithm when the problem size increases.

But that difference is minor compared to the memory the efficient algorithm saves which we saw in the above graph.

So overall we can conclude that the efficient algorithm will save a good amount of memory when the problem size is large even if it takes little bit more time to execute than the basic algorithm.