SUMMARY

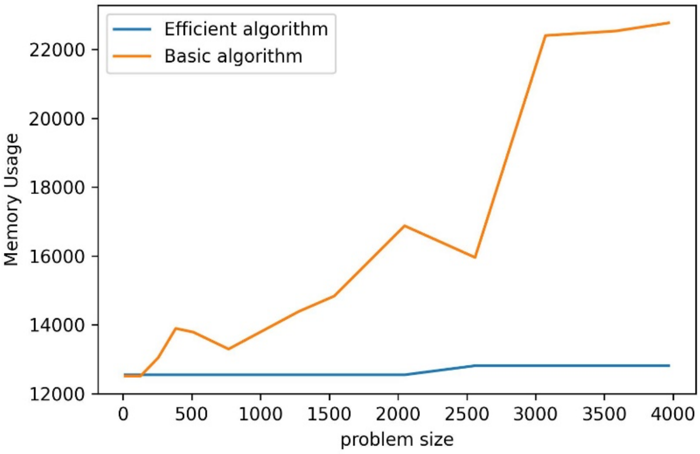
USC ID/s: 2806674386, 3549958057, 4389700845

Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.30	0.48	12508	12544
64	0.94	1.88	12508	12544
128	3.56	6.30	12508	12544
256	11.70	26.28	13036	12544
384	26.57	50.90	13892	12544
512	47.65	88.62	13784	12544
768	110.76	192.43	13292	12544
1024	206.67	335.60	13840	12544
1280	324.60	518.76	14388	12544
1536	467.30	737.70	14832	12544
2048	866.36	1350.48	16876	12544
2560	1350.40	2146.67	15956	12808
3072	1940.63	3005.74	22408	12808
3584	2641.00	4183.75	22540	12808
3968	3308.58	5126.91	22776	12808

Insights

Graph1 – Memory vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

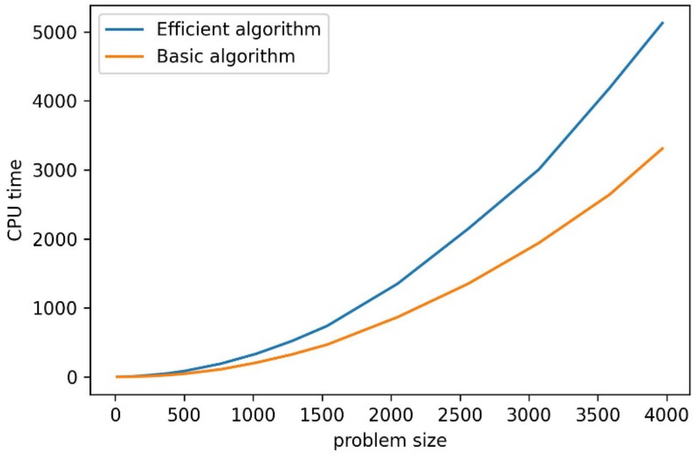
Basic: Polynomial

Efficient: Linear

*Explanation: The efficient solution takes only O(N * 2) space every time whereas basic algorithm uses O(N * M) space every single instance. The sudden memory usage blips are because of python's memory management processes which trigger garbage disposal only after certain points.*

*The efficient version is considerably more memory efficient as it grows linearly with the input size. Since we only need two rows of string length (the solution needs only values from the previous computed row), the other rows can be safely discarded. The basic version grows at O(m * n). Its growth rate is so much faster that it makes the efficient version look static even though it is linear.*

Graph2 – Time vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

Efficient: Polynomial

*Explanation: Both algorithms run for O(N * M) time. The efficient algorithm is slightly higher because it has the extra overhead of shifting column values to maintain O(N * 2) memory consumption.*

It can also be noted that the efficient version takes longer to run for a given pair of inputs. This is because at the first level of divide and conquer, we're computing the entire solution space, but at despite that, we would still be performing computations with the problem size decreasing by half in every subsequent level. So, overall the computation required would be twice that of the basic version.

Contribution

2806674386:	Equal Contribution
3549958057:	Equal Contribution
4389700845:	Equal Contribution