SUMMARY

USC ID/s:

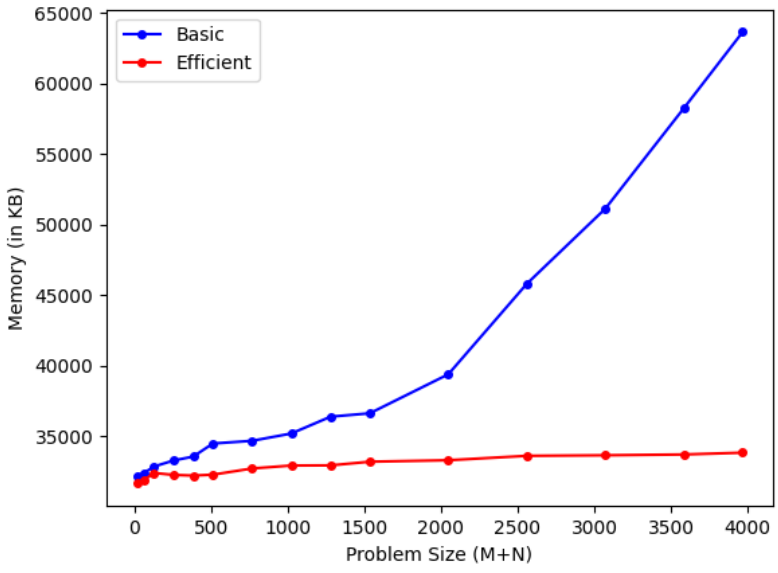
9920441456

Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.1769	0.2382	32152	31648
64	0.8969	0.9491	32400	31896
128	2.9991	8.7461	32848	32392
256	11.5051	15.9731	33280	32256
384	25.2581	28.4772	33552	32208
512	55.2471	58.3827	34480	32272
768	118.8879	137.2061	34672	32712
1024	248.9919	277.1569	35192	32920
1280	295.1219	333.4828	36384	32932
1536	467.4561	485.7898	36624	33192
2048	748.1908	863.4851	39392	33296
2560	1152.2241	1301.2651	45808	33608
3072	1674.6771	1977.0821	51136	33648
3584	2240.4189	2577.6912	58256	33704
3968	2752.9859	3291.9738	63664	33832

Insights

Graph1 – Memory vs Problem Size (M+N)



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

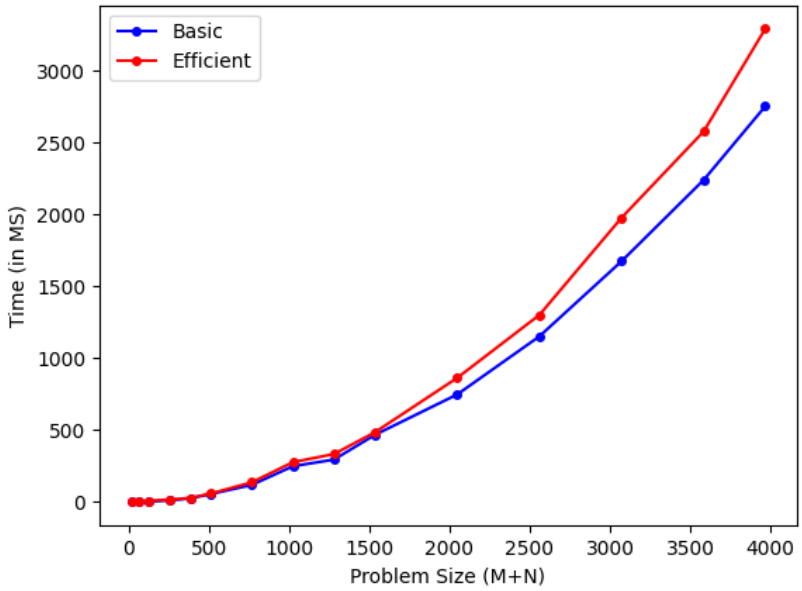
Basic: Exponential

Efficient: Linear

Explanation:

The space complexity for the basic algorithm is O(mn), while for the memory-efficient version, it is reduced to O(min(m,n)). The memory-efficient algorithm avoids the need to store the entire m*n matrix typically used for holding sub-problem costs. Therefore, memory-efficient version exhibits a linear trend, in contrast to the basic algorithm, where space complexity follows a exponential trend.

Graph2 – Time vs Problem Size (M+N)



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

Basic: Exponential

Efficient: Exponential

Explanation:

The time complexity of the algorithm is O(n*m), and it exhibits an exponential growth pattern. Both versions of the algorithm demonstrate this trend; however, the increase is more pronounced in the memory-efficient version. Because the memory-efficient version need additionally determine the splitting point in the optimal alignment while addressing subsequent sub-problems.

Contribution

(Please mention what each member did if you think everyone in the group does not have an equal contribution, otherwise, write “Equal Contribution”)

9920441456: Equal Contribution