CSCI 570 - Final Project Summary

USC ID/s: 8820165551 / 6491357214 / 5849113926

1. Datapoints

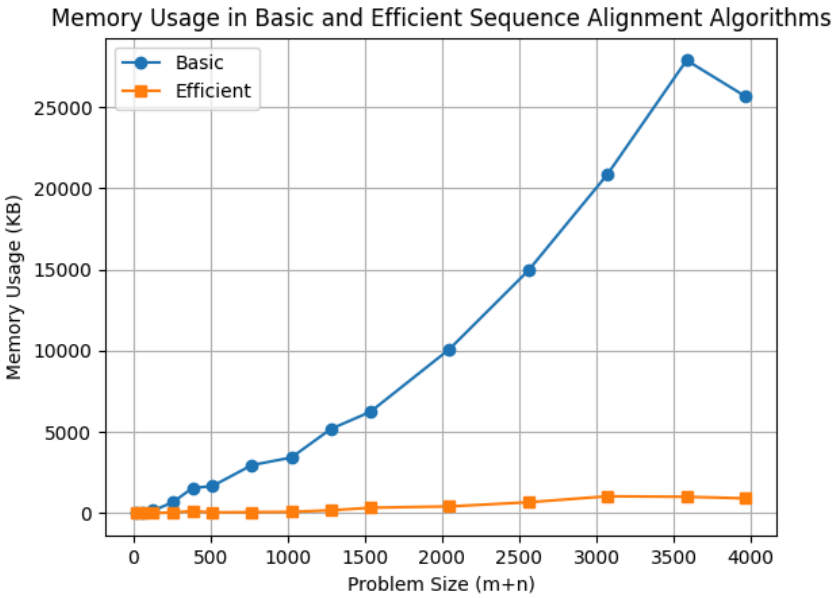
M+N	Time (ms)		Memory (KB)	
	Basic	Efficient	Basic	Efficient
16	0.044107	0.098228	0.0	0.0
64	0.280857	0.629902	48.0	0.0
128	1.226902	2.174139	160.0	0.0
256	4.645109	6.909847	672.0	48.0
384	9.516954	16.407013	1552.0	128.0
512	18.127203	29.452085	1664.0	48.0
768	41.704893	65.733671	2960.0	64.0
1024	74.683905	117.842197	3424.0	80.0
1280	117.423058	176.045895	5184.0	176.0
1536	165.526152	252.722740	6240.0	336.0
2048	309.430838	456.459045	10096.0	416.0
2560	477.304697	740.299702	14944.0	672.0
3072	670.652151	1000.197887	20864.0	1040.0
3584	930.646181	1396.594048	27872.0	1008.0
3968	1168.347836	1783.634186	25632.0	912.0

[Table 1. Comparison of time and memory usage between the Basic and Efficient sequence alignment algorithms across varying problem sizes (m + n). Time is measured in milliseconds (ms) and memory in kilobytes (KB).]

1.1 Insights

Memory usage in the Basic version grows polynomially with the problem size, whereas the Efficient version consistently maintains very low memory consumption, regardless of the input size. This is due to its space-optimized design that avoids constructing the full dynamic programming table.
In contrast, CPU time for both versions increases polynomially with the problem size. However, the Efficient version tends to be slightly slower at larger scales, due to additional recursion steps involved in divide-and-conquer steps.
Therefore, although both versions exhibit similar time complexity trends, the Efficient version clearly outperforms the Basic version in terms of scalability and memory efficiency.

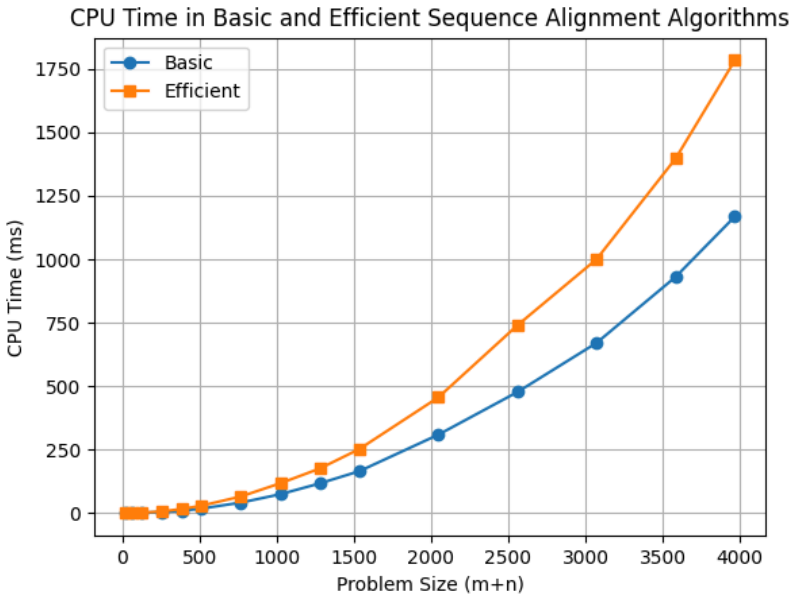
2. Graph1 – Memory vs Problem Size (m+n)



[Figure 1. Memory usage comparison between the Basic and Efficient sequence alignment algorithms.]

2.1. Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)
Basic: **Polynomial** (m*n)
Efficient: **Linear** (m+n)
Explanation: The Basic algorithm allocates and fills the full (m+1)*(n+1) DP table, so its memory usage grows as Θ(m*n). As a result, the blue line, representing the memory usage of the Basic algorithm, shows quadratic growth. The Efficient method, by contrast, keeps only two rows in memory at any time, yielding Θ(m+n) memory usage. As such, the orange line, representing memory usage of the Efficient algorithm, shows linear growth.

3. Graph2 – Time vs Problem Size (m+n)



[Figure 2. CPU time comparison between the Basic and Efficient sequence alignment algorithms.]

3.1. Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)
Basic: **Polynomial** (m*n)
Efficient: **Polynomial** (m*n)
Explanation: Both algorithms require Θ(m*n) computation time - the Basic version due to its nested loops (j=1…n, i=1…m) and the Efficient due to the divide-and-conquer steps, plus two matrix retrievals per recursion. As a result, the Efficient algorithm curve (orange) sits slightly above the Basic algorithm (blue) curve, despite sharing the same Θ(m*n) time complexity.

4. Contribution

Equal Contribution