SUMMARY

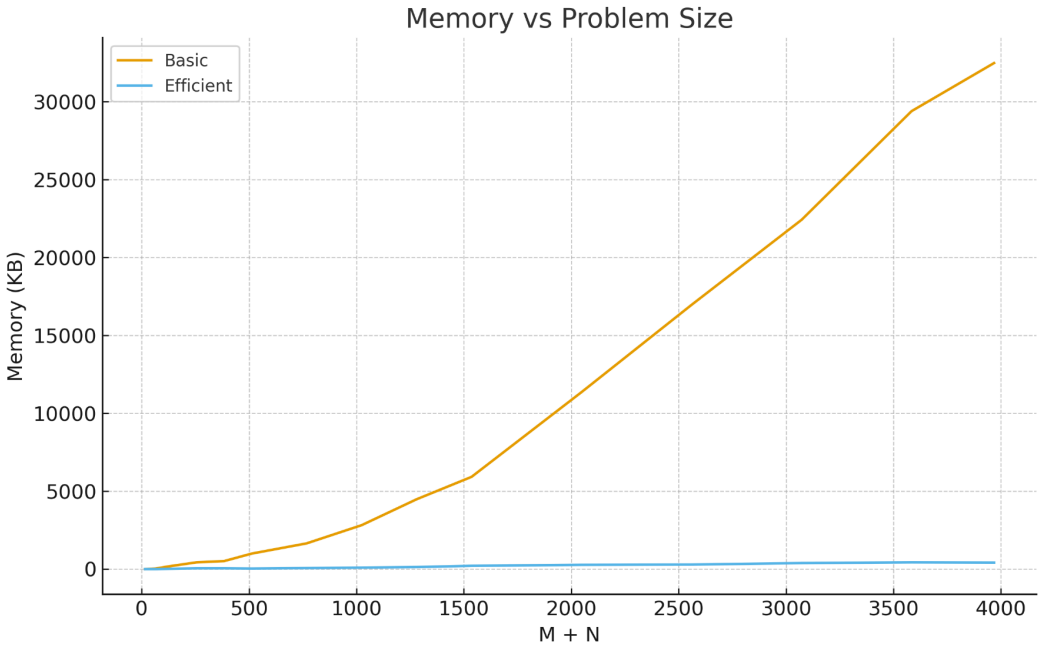
USC ID/s:
5793476658
9026431618
2129850673
9165768219
1352194030

Datapoints:

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.04792213439941406	0.11587142944335938	0	0
64	0.5259513854980469	0.9920597076416016	32	0
128	1.9409656524658203	5.26881217956543	176	16
256	7.562875747680664	12.443065643310547	432	48
384	16.81208610534668	32.032012939453125	512	48
512	30.6396484375	53.88593673706055	992	32
768	72.8600025177002	126.38688087463379	1648	64
1024	131.99520111083984	216.3529396057129	2816	96
1280	210.10684967041016	341.33219718933105	4480	128
1536	302.217960357666	467.32211112976074	5920	208
2048	542.8388118743896	843.0030345916748	11360	272
2560	858.4489822387695	1264.1420364379883	16960	288
3072	1231.2161922454834	1755.3150653839111	22416	384
3584	1669.658899307251	2355.687141418457	29392	432
3968	2103.646993637085	3030.2340984344482	32480	416

Insights

Graph1 – Memory vs Problem Size (M+N)



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

Basic: Polynomial

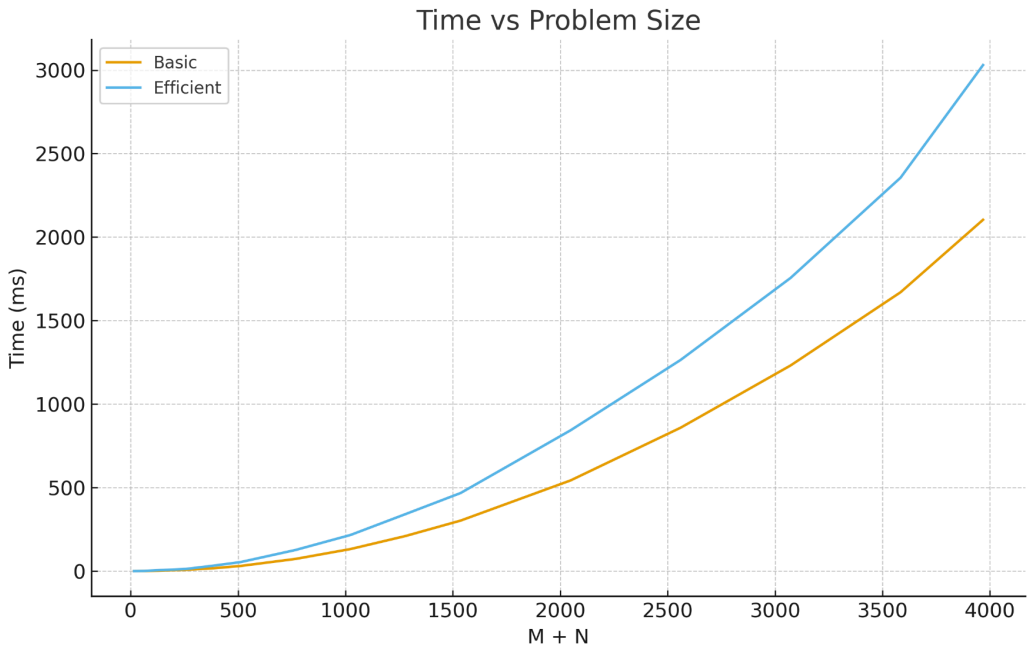
Efficient:Constant

Explanation:

For the basic algorithm, we stored a (M+1) * (N+1) DP table, so the theoretical memory complexity is Theta(M*N). This matches with what we see in the graph. The orange basic line grows in a quadratic fashion, and this is more clear if we look closely at the datapoints: for instance, as the problem size doubles from 1024 to 2024, the basic memory goes from 2816 KB to 11360 KB (nearly 4 times more); as the problem size doubles from 1280 to 2560, the basic memory goes from 4480 KB to 16960 KB which is also nearly 4 times more, more than a factor of 2 (expected for linear growth). It's also obvious in the graph that the orange curve doesn't grow exponentially fast, so the basic curve grows polynomially following the theoretical memory complexity.

For the efficient algorithm, we avoid storing the full (M+1) × (N+1) DP table and instead keep only a small fixed number of rows at any time. This reduces the theoretical memory complexity to theta(N). The graph confirms this: even as the problem size doubles, the memory usage increases only slightly. For example, from 96 KB at size 1024 to 272 KB at 2048, and from 128 KB at 1280 to just 288 KB at 2560. These small increases come from overhead rather than actual scaling with input size. As shown in the graph, the efficient curve stays nearly flat, consistent with the expected near-constant memory complexity.

Graph2 – Time vs Problem Size (M+N)



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

Basic:Polynomial (Quadratic-like)
Efficient:Polynomial (Quadratic-like)

Explanation:

The basic algorithm fills a (M+1) * (N+1) DP table once. There are Theta(M*N) cells, and filling each cell does constant time work (calculating 3 candidate costs and taking the minimum). So the theoretical running time is Theta(M*N). This matches with the graph: the orange Basic curve starts near 0 ms and then increases more than linearly as M + N grows. For example, when the problem size doubles from 1024 to 2028, the time increases from about 132 ms to about 543 ms (nearly 4 times more), which is consistent with quadratic growth rather than linear or exponential growth.

The efficient algorithm also evaluates all (M+1) × (N+1) DP states, but instead of storing the entire table, it keeps only a small number of rows in memory. Since each DP cell still requires constant-time work (computing three candidate costs and taking the minimum), the theoretical running time remains theta(M × N). This matches what we see in the graph: the efficient curve starts close to zero and then grows more than linearly as M + N increases. For example, as the problem size doubles from 1024 to 2028, the time rises from about 216 ms to about 843 ms, almost a 4× increase again consistent with quadratic growth rather than linear or exponential behavior. Although the memory usage is lower, the time complexity remains polynomial because the same total number of DP computations must be performed.

Contribution

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

5793476658: Equal Contribution
9026431618: Equal Contribution
2129850673: Equal Contribution
9165768219: Equal Contribution
1352194030: Equal Contribution