SUMMARY

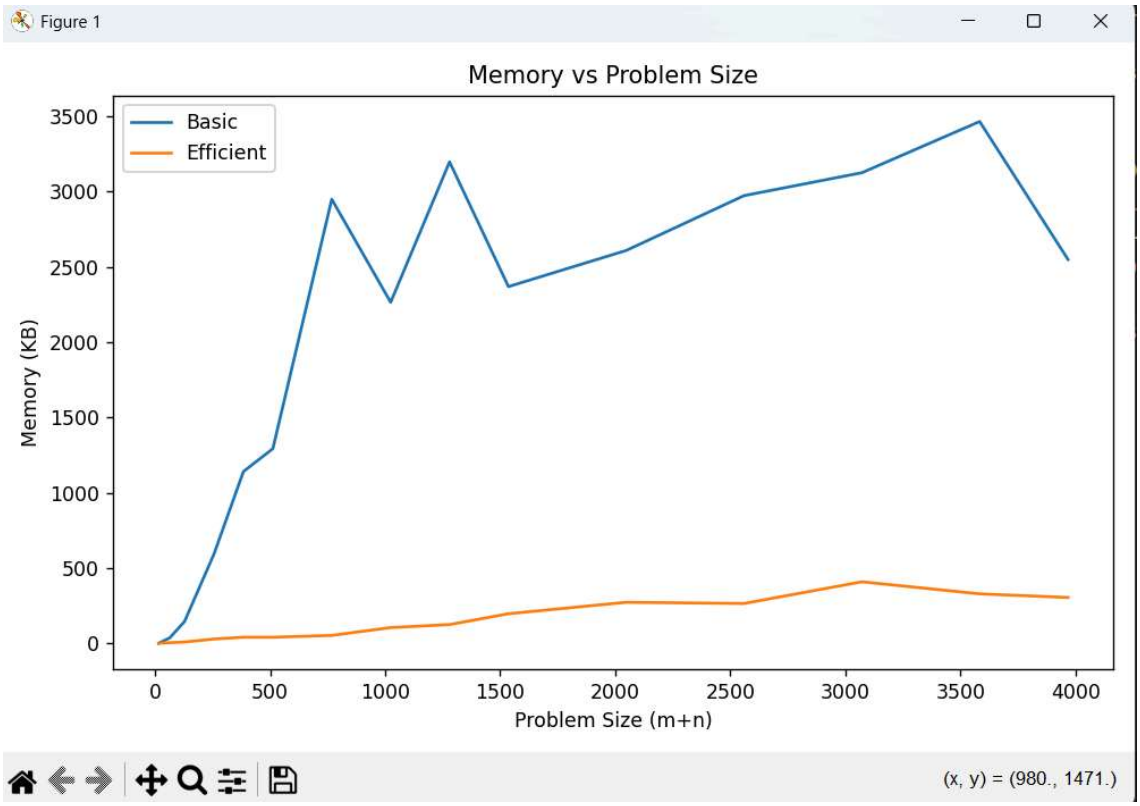
USC ID/s: 7825330815, 8554425845, 8279873920, 2868310631, 2715712589

Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.056099845096468925	0.2048015594482422	0	0
64	0.2665000502020120 6	0.9539127349853516	36	4
128	0.9389000479131937	2.536773681640625	144	8
256	3.4153000451624393	11.889934539794922	592	28
384	7.689600111916661	25.570154190063477	1140	40
512	13.630599947646626	52.176475524902344	1292	40
768	31.911200145259 5	92.838287353515 62	2948	52
1024	59.030300006270 41	170.73535919189453	2264	104
1280	92.644300078973317	281.83650970458984	3196	124
1536	134.39540006220 34	386.28864288330 08	2368	196
2048	241.84569995850325	748.15702438354449	2608	272
2560	390.946399886 1611	1191.3566589355469	2972	264
3072	550.284299999 4755	1638.1537914276123	3124	408
3584	739.701600046 8284	2179.3150901794434	3464	328
3968	933.011299930 5129	2704.071044921875	2548	304

Insights

Graph1 – Memory vs Problem Size (M+N)



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

Basic: Polynomial

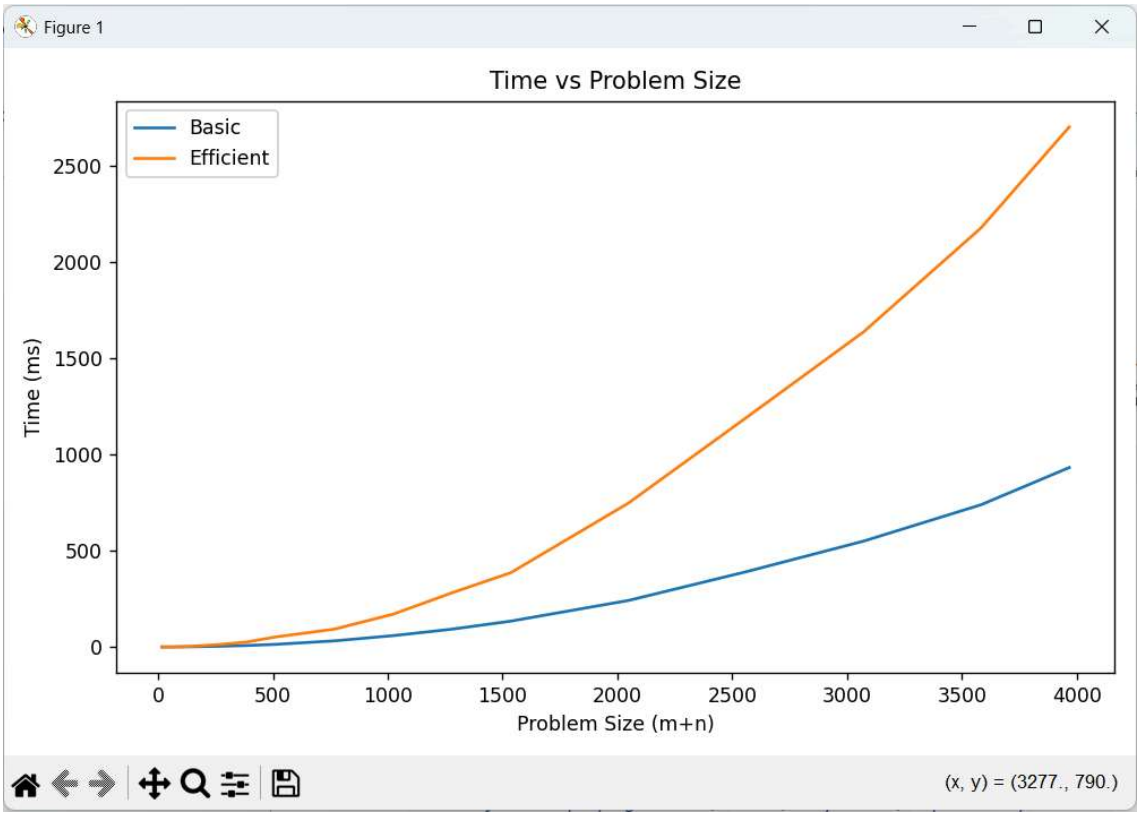
Efficient: Linear

Explanation:

Basic Algorithm takes O(mn) space complexity to store the 2d dynamic programming graph. As the problem input size increases,memory size increases rapidly from 0 at problem size 16 to 2548 KB at problem size 3968.

The Space Efficient Algorithm stores only two rows at a time (current and previous), making memory usage dependent only on O(n). It's memory curve grows linearly and extremely slowly, increasing from almost 0 KB to only 304 KB, even for the largest input.

Graph2 – Time vs Problem Size (M+N)



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

Basic:Polynomial

Efficient:Polynomial

Explanation:

The basic algorithm performs at O(mn) complexity to compute all the values of the 2d DP graph. Therefore polynomial in nature.

The efficient algorithm also takes O(mn) complexity but we see that it takes more time than the basic algorithm. The reason is that the memory efficient algorithm does not store the full DP matrix, so it recomputes partial values and performs additional row copying or swapping. This results in more overhead, making the memory-efficient algorithm slower even though it uses significantly less memory.

Contribution

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

7825330815: Equal Contribution

2715712589: Equal Contribution

8279873920: Equal Contribution

8554425845: Equal Contribution

2868310631: Equal Contribution