SUMMARY

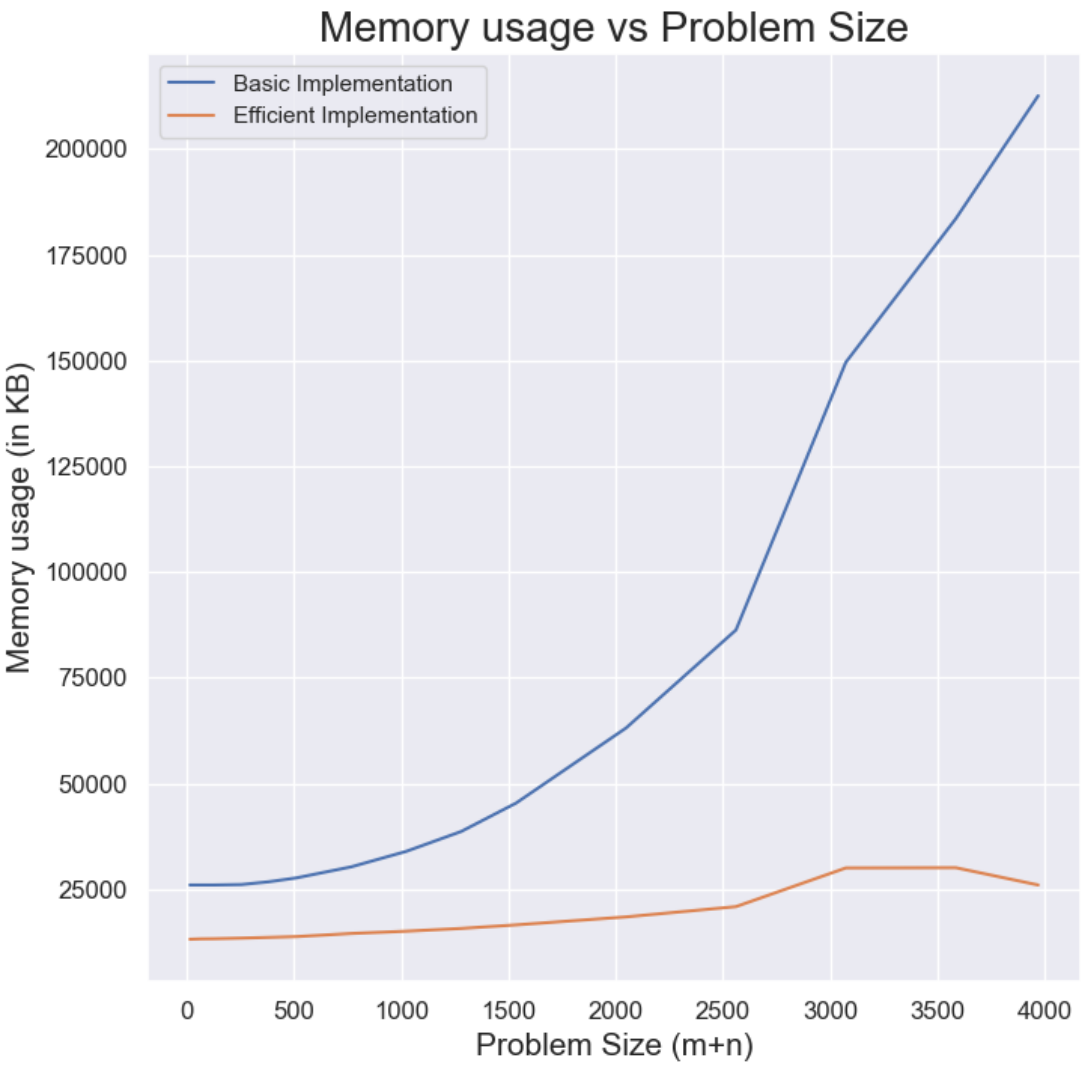
USC ID/s:
1038581442
6298935190

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.073	0.2801	26048	13216
64	0.6206	1.6289	26048	13304
128	2.316	5.5108	26048	13340
256	8.9171	19.1281	26152	13492
384	20.2351	41.0323	26808	13664
512	35.8031	72.2697	27720	13844
768	83.7047	163.358	30336	14584
1024	151.33	288.4071	33992	15124
1280	238.4241	449.4662	38692	15776
1536	346.9191	646.4672	45416	16616
2048	622.7582	1161.9859	63156	18504
2560	987.756	1841.6812	86316	20928
3072	1474.3912	2644.6772	149600	30072
3584	1968.7421	3634.66	183520	30132
3968	2413.3	4482.1451	212584	26048

Datapoints

Insights

Graph1 – Memory vs Problem Size (M+N)



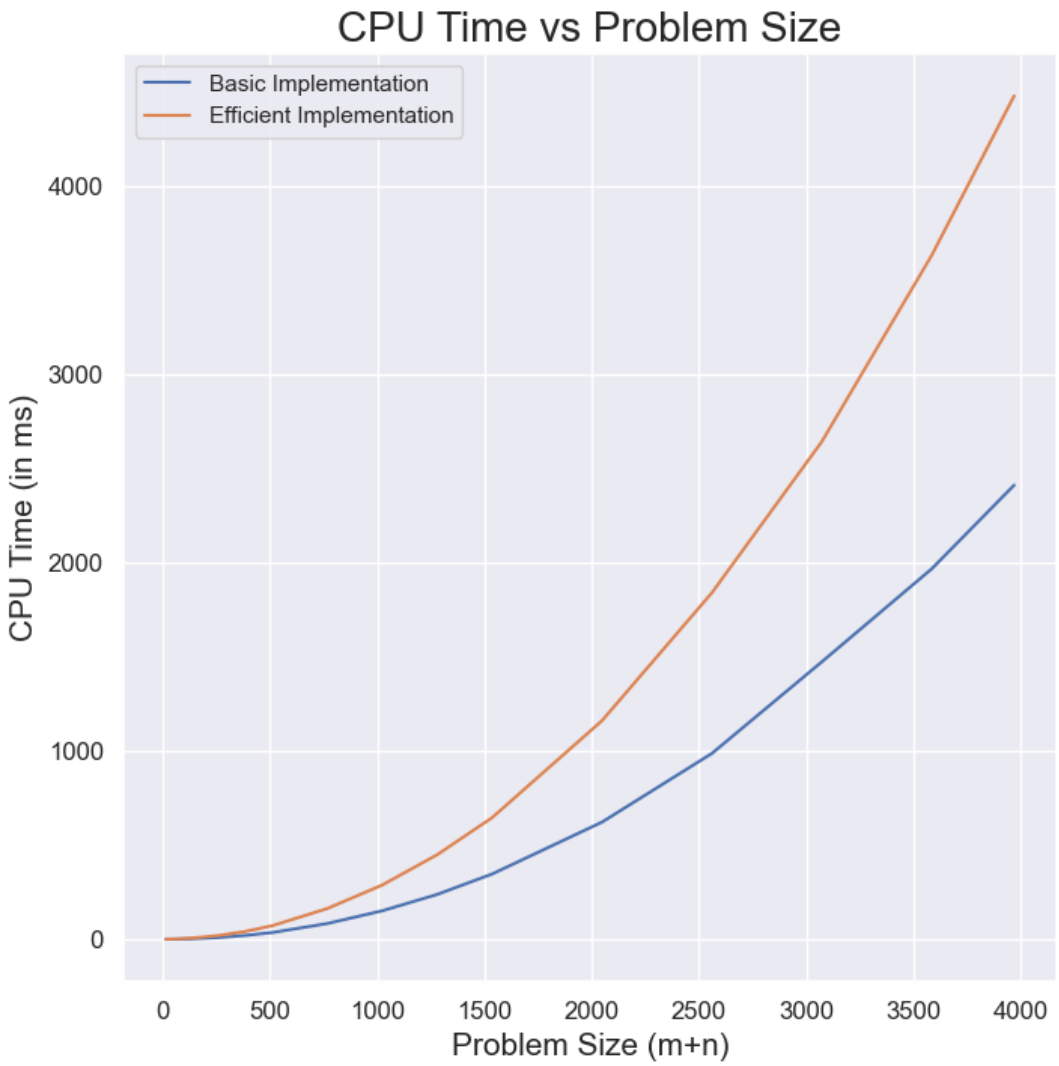
Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Exponential

Efficient: Linear

*Explanation: The efficient algorithm uses the memory-efficient method to compute the solution. It does not require the entire m*n matrix to store the costs of the sub-problems, but only requires the last two columns. Hence, the memory usage follows a linear trend for the memory-efficient algorithm, while it follows an exponential trend for the basic algorithm. The space complexity of basic algorithm is O(m*n), and for the space-efficient algorithm, it is O(m+n).*

Graph2 – Time vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Exponential

Efficient: Exponential

*Explanation: The runtime for the algorithm is O(n*m), which follows an exponential curve. The trend is the same for both versions of the algorithm, but shows a higher increase for the memory-efficient version. This is because the memory-efficient version has the additional complexity to compute the splitting point in the optimal alignment, when solving subsequent sub-problems.*

Contribution

1038581442: Equal Contribution

6298935190: Equal Contribution