SUMMARY

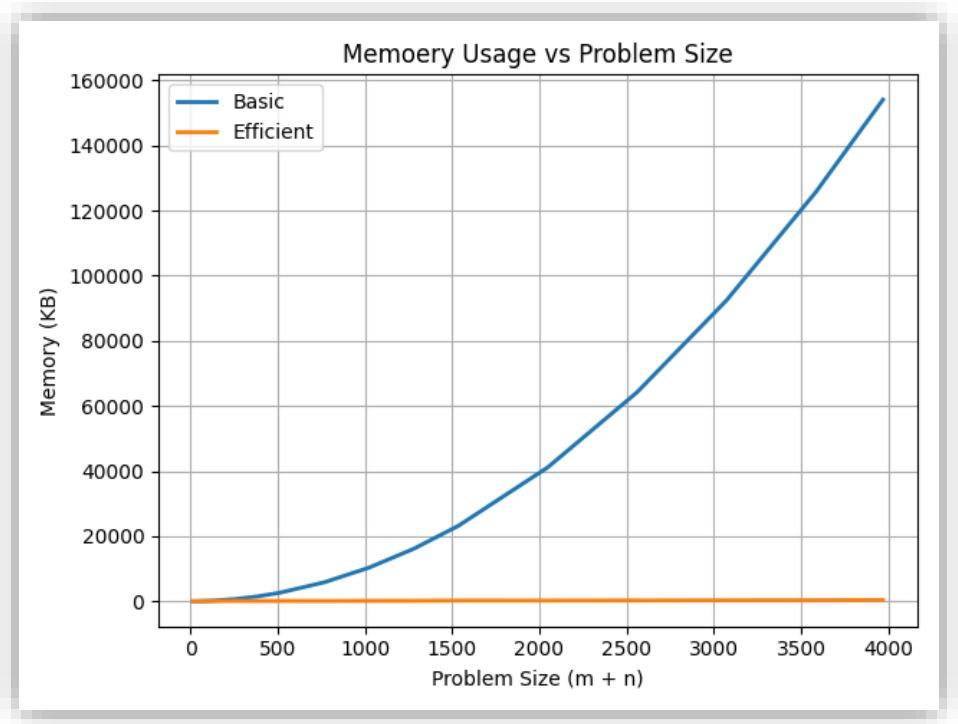
USC ID/s: 8203536510

Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0	0	1.25	1.3
64	4	5	42.41	42.46
128	24.76	20.02	169.4	55.84
256	72.39	87.23	661.05	125.79
384	164.21	227.7	1473.49	92.2
512	317.53	398.03	2600.36	106.78
768	920.5	1062.01	5822.59	88.49
1024	1882.42	1901.91	10322.3	144.22
1280	3431.12	3366.83	16111.47	151.39
1536	4611.38	5465.55	23172.88	230.43
2048	9125	8785.78	41135.65	228.03
2560	14660.34	14401.04	64225.53	256.9
3072	21152.95	21002.39	92435.54	303.8
3584	30193.51	27977.83	125739.4	341.59
3968	35180.08	36145.8	154098.54	386.55

Insights

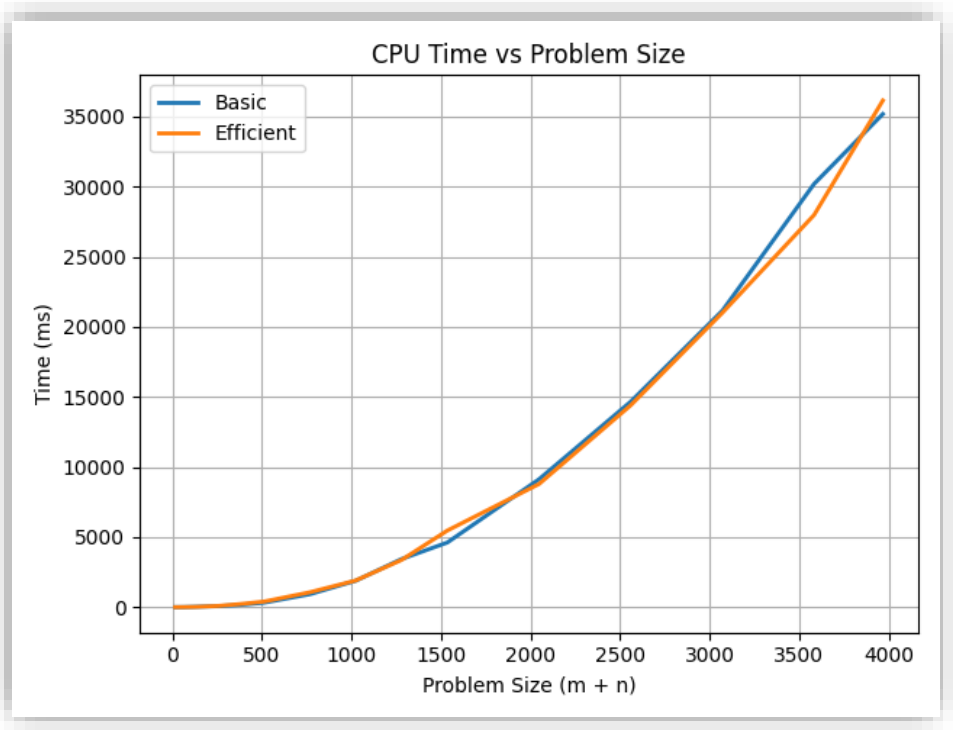
Graph1 – Memory vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)
Basic: Polynomial
Efficient: Linear

*Explanation: The points represent the peak memory usage (worst-case) of both algorithms as the input increases. From the graph we can see that the basic algorithm's memory consumption grows polynomially as the input size increases in contrast to the efficient algorithm that grows linearly and maintains a straight line. In the basic implementation we build a dp-table of size (x+1) * (y+1) so the memory usage grow quadratically O(n^2) whiles the efficient implementation only stores two rows of y+1 length and a recursion stack which is typically O(logx) which amounts to a usage of O(n) space.*

Graph2 – Time vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)
Basic: Polynomial
Efficient: Polynomial

Explanation: The points represent the total runtime of both algorithms as the input increases. From the graph we can see that the basic and efficient algorithm's runtime grows polynomially as the input size increases, this is due to both implementations computing the dp-table of (x+1)(y+1) length, however the efficient is a bit slower due to working one row at a time rather than a double loop and the recursion plus swapping overhead.*

Contribution

<USC ID/s>: 8203536510 <Equal Contribution>* I worked on this project by myself.