SUMMARY

USC ID/s:

Aaron Trusty – 2967685514

Aman Thirthahalli Bhat – 4181579368

Rhythm Girdhar - 6742001330

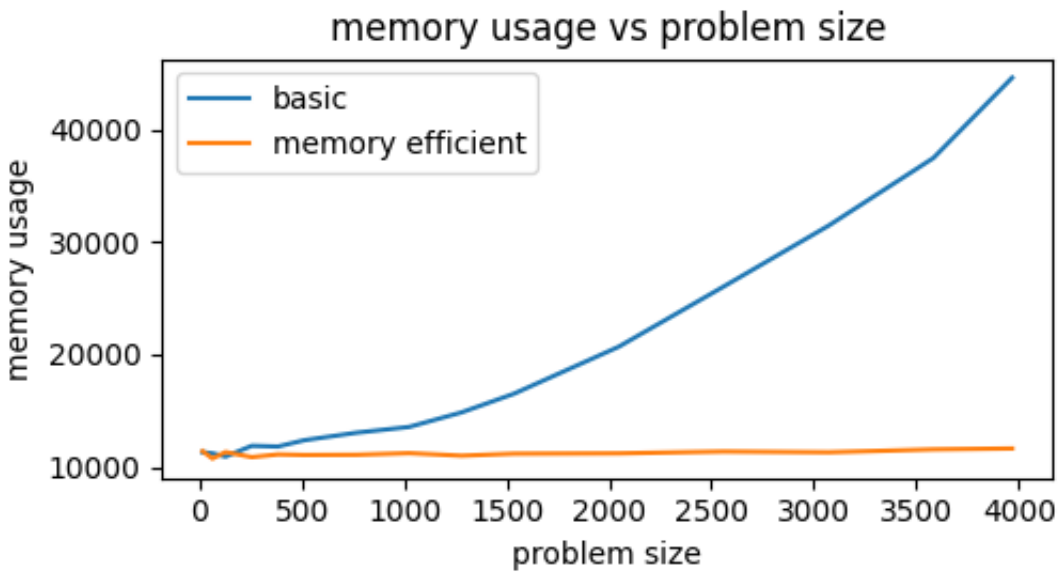
Datapoints:

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.24819374084472656	0.308990478515625	11296	11392
64	0.843048095703125	1.4088153839111328	11200	10736
128	1.7681121826171875	3.857851028442383	10912	11312
256	6.237030029296875	13.095855712890625	11872	10864
384	14.740943908691406	28.390169143676758	11808	11104
512	24.191856384277344	51.00297927856445	12368	11056
768	56.84804916381836	111.78183555603027	13040	11072
1024	102.91600227355957	201.05814933776855	13552	11216

1280	159.95407104492188	314.4800662994385	14848	10992
1536	230.38005828857422	441.2107467651367	16512	11168
2048	414.8707389831543	793.6289310455322	20704	11200
2560	673.0489730834961	1245.0778484344482	26064	11360
3072	931.8509101867676	1774.7640609741211	31488	11296
3584	1291.0089492797852	2459.3830108642258	37520	11552
3968	1587.6417160034218	3090.3830528259277	44672	11632

Insights:

Graph1 – Memory vs Problem Size (M+N)



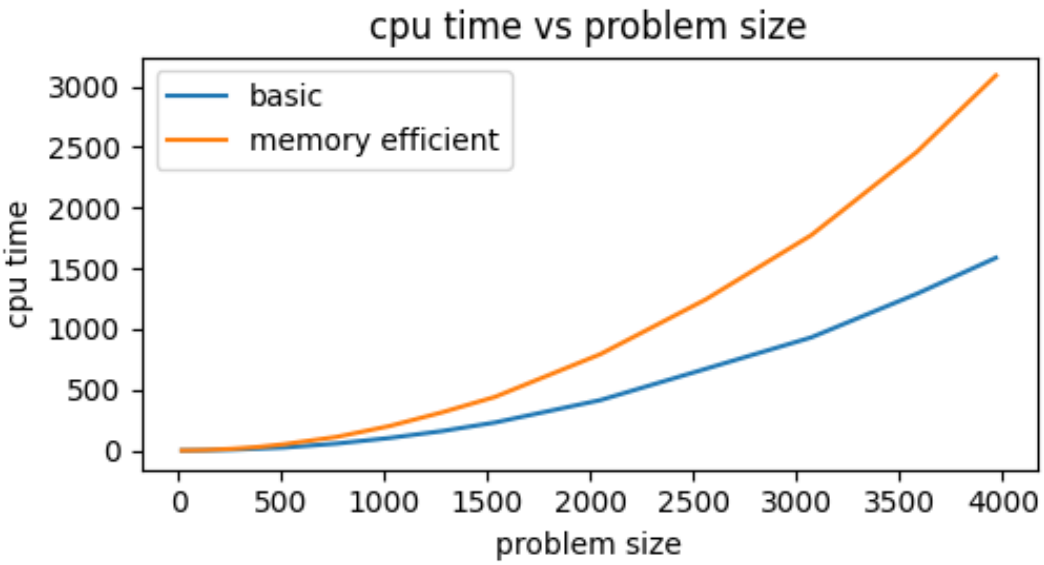
Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

Efficient: Linear

Explanation: The memory efficient algorithm consumes significantly less memory than the basic version because it uses two columns of the memoization table O(2*m) instead of using the whole O(m*n) table in the basic version. Additionally, we are using the longer string as our X which is split into X-L and X-R and the midpoint is found in the shorter string. In this way, the space and time of mid-point calculation is a function of the shorter string which is an optimization over the regular way of using the first string as X and the other one as Y. Moreover, in the efficient version we do not need to store all the values for the entire solution space, and instead only what is necessary is kept. As shown in the orange line on the graph, this has a tremendous impact on the memory utilized by the algorithm. In our testing of the efficient solution, we got memory values in around the high 10k to low 11k range of kilobytes.

Graph2 – Time vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

Efficient: Polynomial

Explanation: Basic algorithm performs better than the memory efficient one because it calculates the DP table and uses it to calculate the optimal cost of sequence alignment. However, in the memory efficient version, we are using the DP table to just get the split for the shorter string and then use Divide and Conquer on top it. The memory efficient algorithm is more computation-intensive and hence, it uses more CPU time. However, we believe that they are on the same asymptotic order (being exponential) as both blue and orange lines follow the same overall growth trend. It seems as if these lines only have some constant differentiating factor *C* as shown in the lecture on asymptotic complexity. As the problem size grows, both basic and efficient solutions will grow asymptotically in a similar fashion.

Contribution:

Aaron Trusty – 2967685514 – Equal Contribution

Aman Thirthahalli Bhat – 4181579368 – Equal Contribution

Rhythm Girdhar – 6742001330 – Equal Contribution