SUMMARY

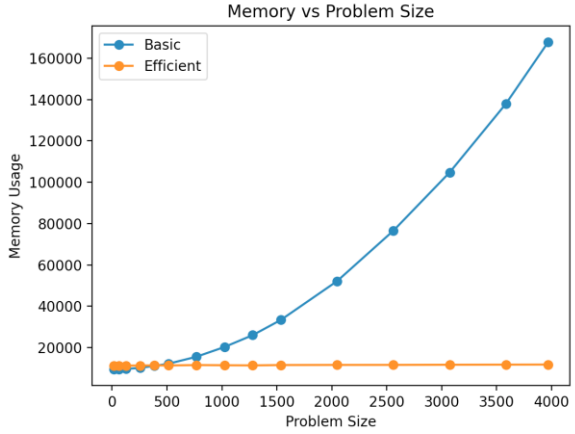
USC ID/s: 6205835871 8763565124 4590705168

Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.38814544677734375	0.7517337799072266	9532	11328
64	1.2679100036621094	3.1828880310058594	9556	11364
128	3.763914108276367	8.800029754638672	9712	11312
256	14.413118362426758	30.597209930419922	10224	11364
384	32.692909240722656	63.819169998168945	11160	11432
512	55.613040924072266	112.7629280090332	12208	11384
768	134.60111618041992	247.48706817626953	15596	11592
1024	236.64426803588867	434.27085876464844	20324	11496
1280	362.7150058746338	675.1213073730469	26048	11412
1536	524.3070125579834	971.2588787078857	33392	11596
2048	945.4779624938965	1717.7817821502686	52172	11644
2560	1495.5158233642578	2737.734794616699	76548	11656
3072	2137.5999450683594	3870.850086212158	104748	11728
3584	2903.4218788146973	5374.644041061401	137940	11784
3968	3597.1591472625732	6545.466899871826	167724	11828

Insights

Graph1 – Memory vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

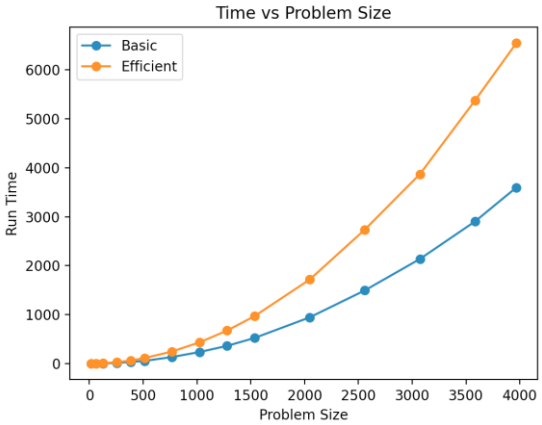
Efficient: Linear

Explanation:

The Basic Algorithm has an exponential space complexity, because we create a data structure of 2D 'table' with a size of m*n. The space needed to store and process data is directly proportional to the input size of the two sequences. The longer the input, the more memory space, and it can be a severe problem when the input size becomes tremendous.

However, the Efficient Algorithm only stores values (total alignment cost) of the optimal alignment which saves a lot of space. For this algorithm, we care about the values instead of the alignment itself that needs a table-like structure. To find the minimum alignment cost, it only uses 2 'arrays' with a total size of 2*M. We have one array of length M holding the value (cost) of the previous column, and based on that, we can then find out the value of the current array. Therefore, for each iteration, we only take up a constant memory space.

Graph2 – Time vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

Efficient: Polynomial

Explanation:

The running time of both the basic and the memory-efficient algorithms is O(m*n), because it takes constant time to determine the value in each of the m*n cells of the cost array. Also, the running time of the memory-efficient algorithm is around twice of the basic one, and this is because in the memory-efficient algorithm, some of the cells are visited multiple times during the divide and conquer stage to determine the alignment, and this blows up the running time of the memory-efficient algorithm by a factor of two, compared to the basic one.

Contribution

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

<USC ID/s>: <Equal Contribution>