SUMMARY

USC ID/s:

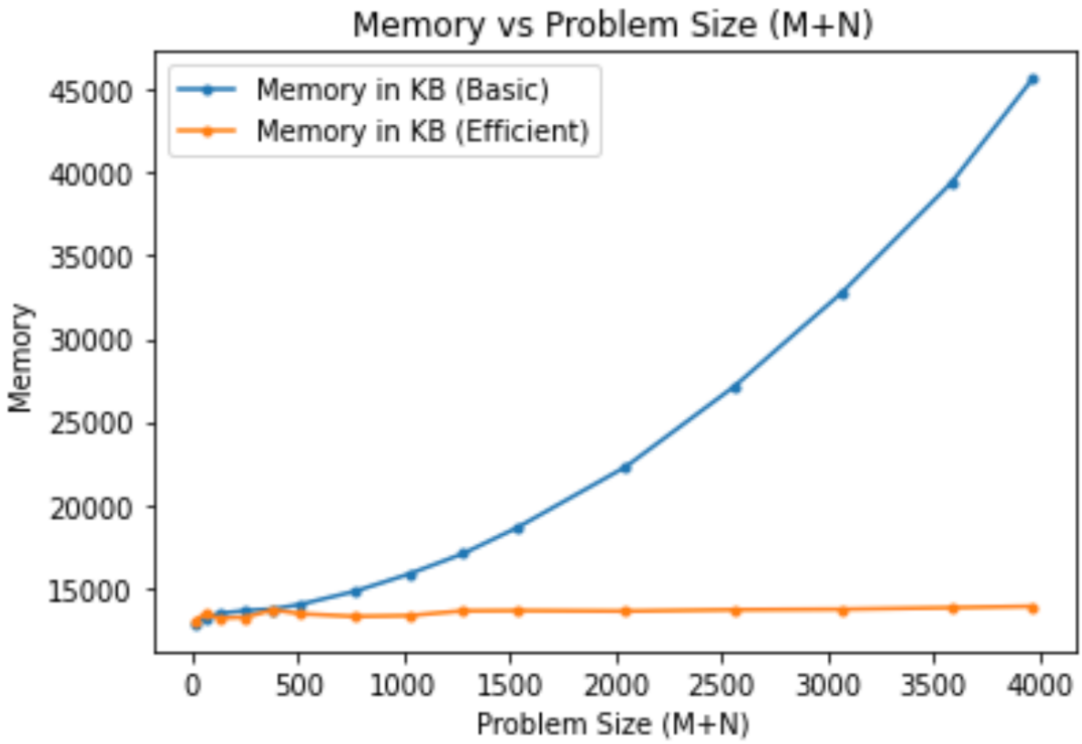
4578711335
9504103749
1582852239

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.0591278076171875	0.141143798828125	12832	13040
64	0.6399154663085938	1.644134521484375	13200	13456
128	2.3469924926757812	5.753278732299805	13456	13184
256	9.624004364013672	21.44622802734375	13632	13232
384	19.51909065246582	42.93084144592285	13728	13680
512	32.2873592376709	71.22468948364258	14000	13440
768	63.15779685974121	148.13709259033203	14784	13280
1024	112.11371421813965	261.74283027648926	15824	13328
1280	194.504976272583	434.71503257751465	17056	13616
1536	266.2010192871094	606.8577766418457	18624	13632
2048	518.5286998748779	1171.9391345977783	22288	13600
2560	733.9468002319336	1686.4230632781982	27120	13680
3072	1141.8938636779785	2578.8679122924805	32816	13712
3584	1246.2120056152344	2977.7140617370605	39376	13808
3968	1802.5999069213867	4127.495050430298	45648	13888

Datapoints

Insights

Graph1 – Memory vs Problem Size (M+N)



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

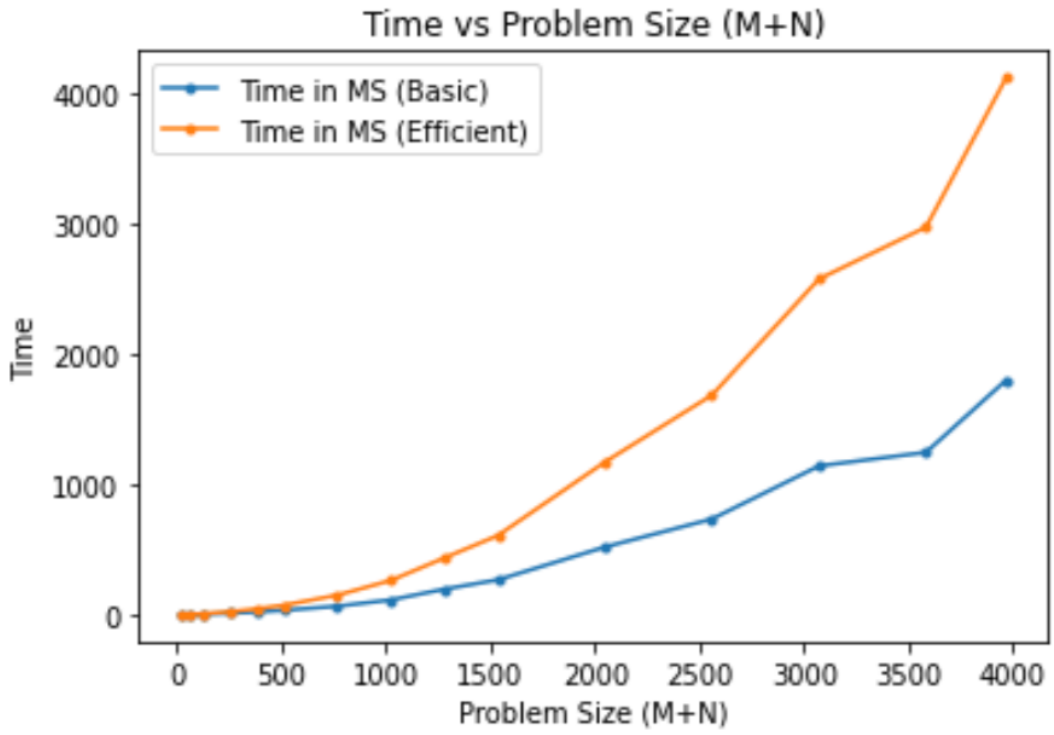
Basic: Polynomial

Efficient: Linear

Explanation:

From the graph, we can see that when the input sizes are small, there's no big difference in memory consumption of basic and efficient algorithm. However when the input size grows, the basic algorithm takes more memory – O(m*n) as we maintain the costs for every conceivable alignment whereas in the efficient algorithm, the memory remains same because we use only limited – 2 columns for computing the cost, here we partition the first string from the middle part and by looking at each index of the second string to divide and selecting the one that provides the lowest cost. Therefore, memory in efficient algorithm is linear while it is polynomial in basic.

Graph2 – Time vs Problem Size (M+N)



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

Basic: Polynomial

Efficient: Polynomial

Explanation:

For small input sizes, the time taken by basic and efficient algorithm is nearly same. Whereas as the input size increases, memory efficient algorithm takes more time to compute the cost where as basic algorithm take less time to compute the result. For memory efficient algorithm we divide the first string from the middle, and then we locate the division point of the second string by looking at every index to divide, and then selecting the index which returns the lowest cost. Here, the time taken for this process is polynomial, plus time for stack maintenances. Coming to basic algorithm, a 2-D matrix is being maintained to determine the cost of all possible combinations in strings backtracking is used to get the exact sequence alignment in this process of cost calculation 2 nested for loops are used the thus time taken is polynomial.

Contribution

Koushik Reddy Konda - 4578711335 - Equal Contribution

Venkata Rohit Kumar Kandula – 9504103749 - Equal Contribution

Rishi Kiran Reddy Nareddy – 1582852239 - Equal Contribution