SUMMARY

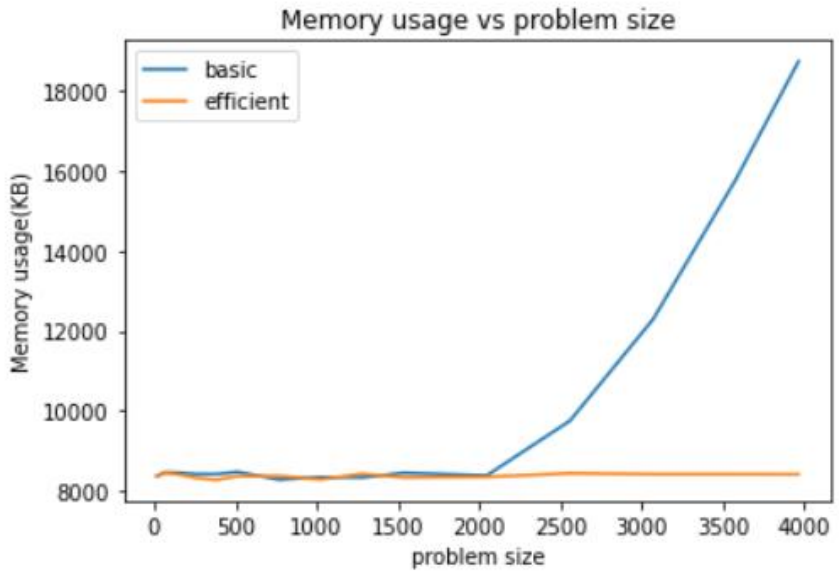
USC ID/s: 8471244017 2048001508 7840118186

Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.25	0.259	8368	8356
64	5.202	2.876	8456	8444
128	1.718	7.797	8452	8420
256	9.58	23.226	8416	8320
384	16.098	57.522	8412	8272
512	46.476	111.533	8468	8360
768	107.078	222.495	8276	8368
1024	158.783	375.523	8336	8280
1280	263.439	534.274	8328	8420
1536	254.007	720.653	8444	8332
2048	499.741	1275.677	8380	8344
2560	791.039	2010.349	9748	8432
3072	1132.322	2818.402	12288	8412
3584	1466.973	3880.031	15804	8412
3968	1778.605	4751.308	18744	8408

Insights

Graph1 – Memory vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

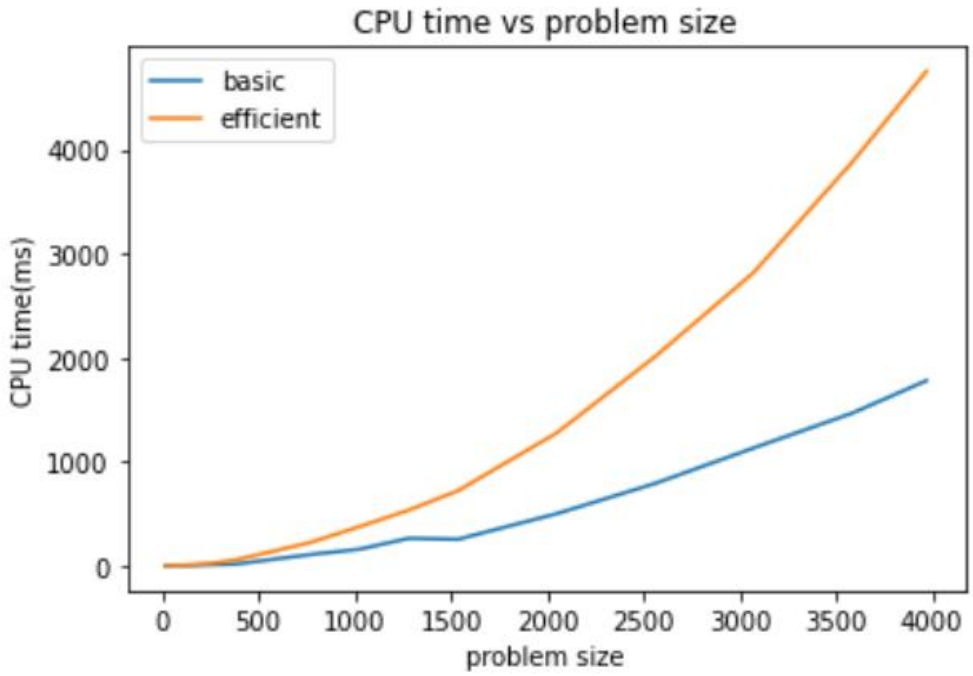
Efficient: Linear

Explanation:

Basic: We build the whole table of size O(mn) by tabulation in order to find the min cost and the alignment, and we also use this table to find the minimum cost sequence alignment.

Efficient: Here we use divide and conquer method to find the minimum cost, since we only need to keep two rows or two columns of the whole table, so we only need to use O(m) or O(n) to memorize two columns or two rows.

Graph2 – Time vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

Efficient: Polynomial

Explanation:

Basic: Similarly, we need to build the whole table of size mn, and for each (i, j) in the table, it take O(1) to find the min cost of (i, j). As for finding the sequence alignment, we start from the top right of the table to the bottom left, which takes O(m) or O(n) time. So in total the runtime is O(mn).

Efficient: Firstly we cut the first string into two equal size pieces, to find the matching point of the second string, we also need to go through every (i, j), but in a memory efficient method. So the total time is mn + mn/2 + mn/4 + ... = O(mn). Also, it takes longer than basic method as an exchange of saving memory.

Contribution

Equal Contribution