SUMMARY

USC ID/s:

1612544813, 8030911799, 3118988459

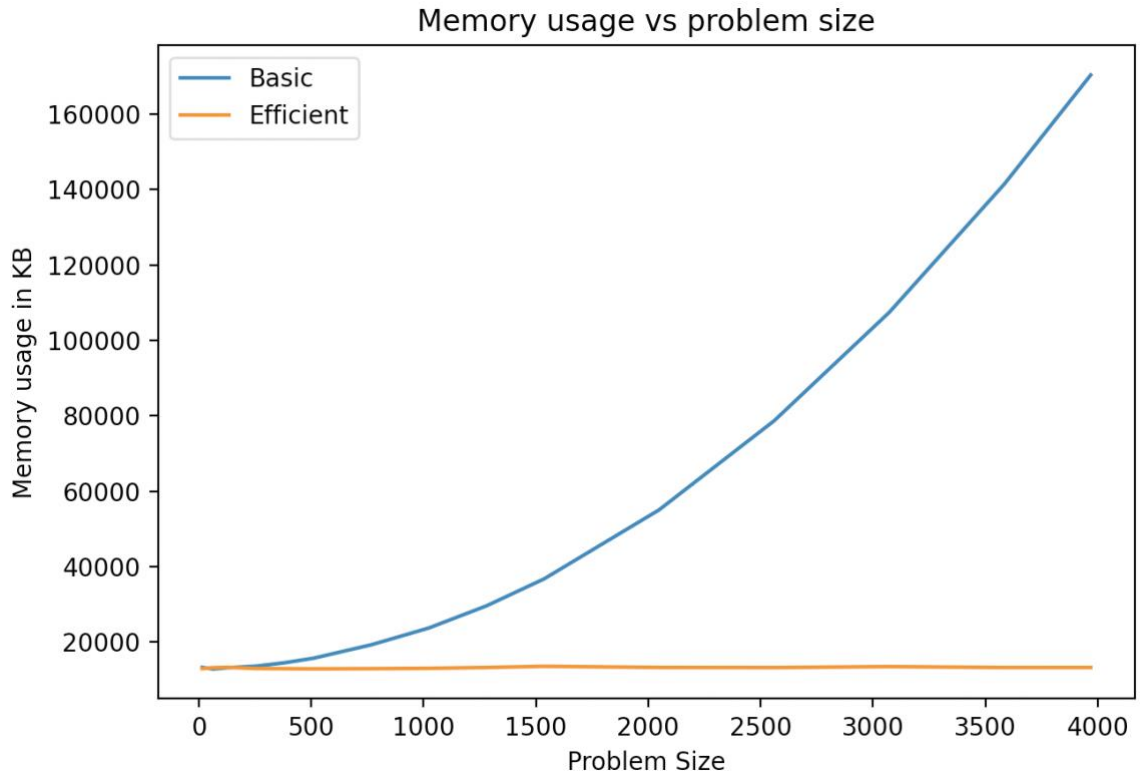
Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.06818771362304688	0.14328956604003906	13316	13004
64	0.7100105285644531	1.4729499816894531	12896	13244
128	2.562999725341797	5.424261093139648	13236	13344
256	9.851932525634766	19.587993621826172	13660	13036
384	22.084951400756836	42.342185974121094	14572	12988
512	41.734933853149414	74.86295700073242	15776	12944
768	90.54708480834961	168.98202896118164	19320	12992
1024	161.5297794342041	294.96192932128906	23776	13084
1280	250.86712837219238	468.72615814208984	29636	13300
1536	367.06089973449707	672.4338531494141	36772	13620
2048	669.88205909729	1366.0931587219238	55080	13312
2560	1055.1140308380127	1997.3900318145752	78704	13276
3072	1533.3991050720215	2788.4278297424316	107444	13540
3584	2169.255256652832	4276.236057281494	141448	13288
3968	2565.0150775909424	4885.197162628174	170348	13300

Insights

- The memory space of the basic algorithm increases polynomially as the problem size increases
- The memory of the efficient algorithm increases linearly with respect to the increase of the problem size
- The CPU time of both algorithms increase polynomially with respect to the increase of the problem size

Graph1 – Memory vs Problem Size (M+N)



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

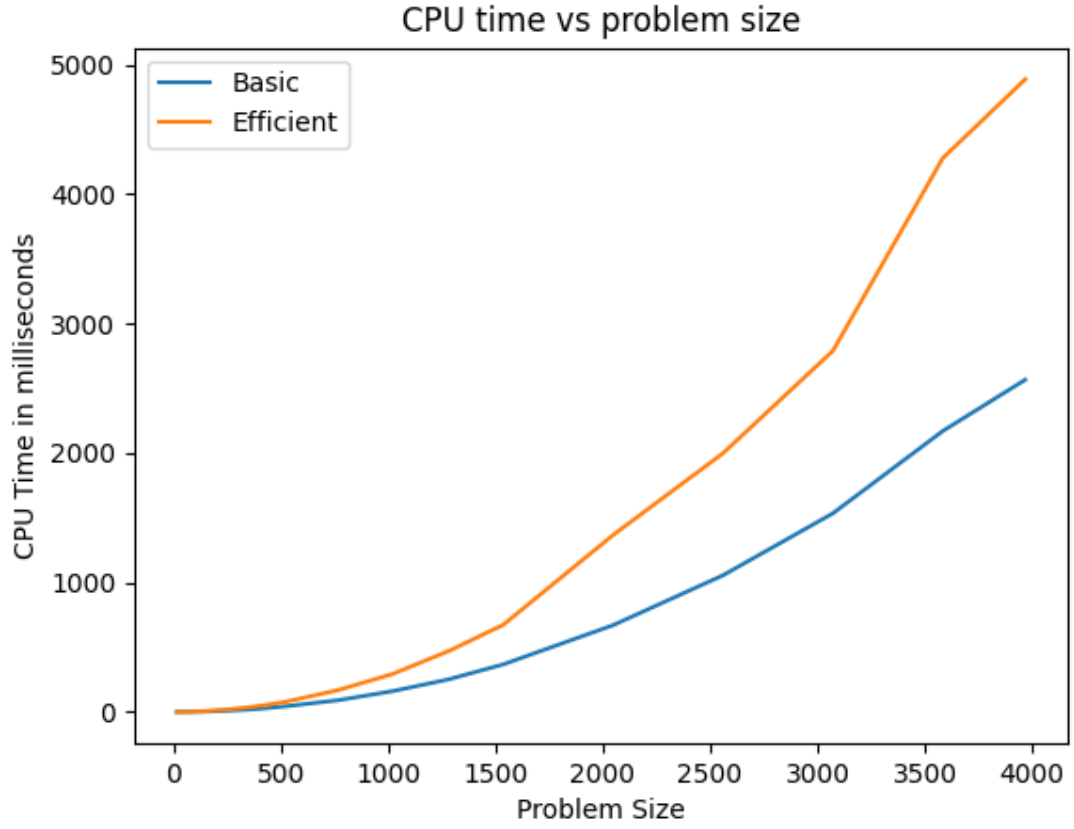
Basic: Polynomial

Efficient: Linear

Explanation:

With the blue upward curve shown in Graph 1, we can find that the memory of the basic algorithm increases polynomially as the problem size increases. Also, the orange line shows that the memory of the efficient algorithm increases linearly with respect to the increase of the problem size. With the result, we would say the efficient algorithm deals with the memory much more efficiently than does the basic algorithm. Such an observation can be found because the basic algorithm uses C*m*n = O(m*n) memory and the efficient algorithm uses C*n = O(n) memory (n is the length of the 2nd string). We can reduce the memory space for the efficient algorithm because we do not have to store all the intermediate values of the dynamic programming table for the purpose of retrieving the actual strings, since we are using divide and conquer algorithm.

Graph2 – Time vs Problem Size (M+N)



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

Basic: Polynomial
Efficient: Polynomial

Explanation:

For both of the basic and the efficient algorithms, the CPU Time tends to increase polynomially with respect to the increase of the problem size. However, we can find in Graph 2 that the basic algorithm takes approximately twice as less CPU time as the efficient algorithm because the efficient algorithm has approximately twice the number of operations of that of the basic algorithm. The time complexity for the basic algorithm is C*m*n = O(m*n) because we iterate two for loops for each string. The time complexity for the efficient algorithm is C*m*n + C*m*n/2 + ... = 2*C*m*n = O(m*n) since we divide the problem size into a half for every iteration until the problem gets trivial and then conquer them in constant time. Note that even though the efficient algorithm needs more CPU time than basic algorithm, the time complexity of both algorithm are the same.

Contribution

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

<1612544813>: <Equal Contribution>

<8030911799>: <Equal Contribution>

<3118988459>: <Equal Contribution>