SUMMARY

USC ID/s:

1612544813, 8030911799, 3118988459

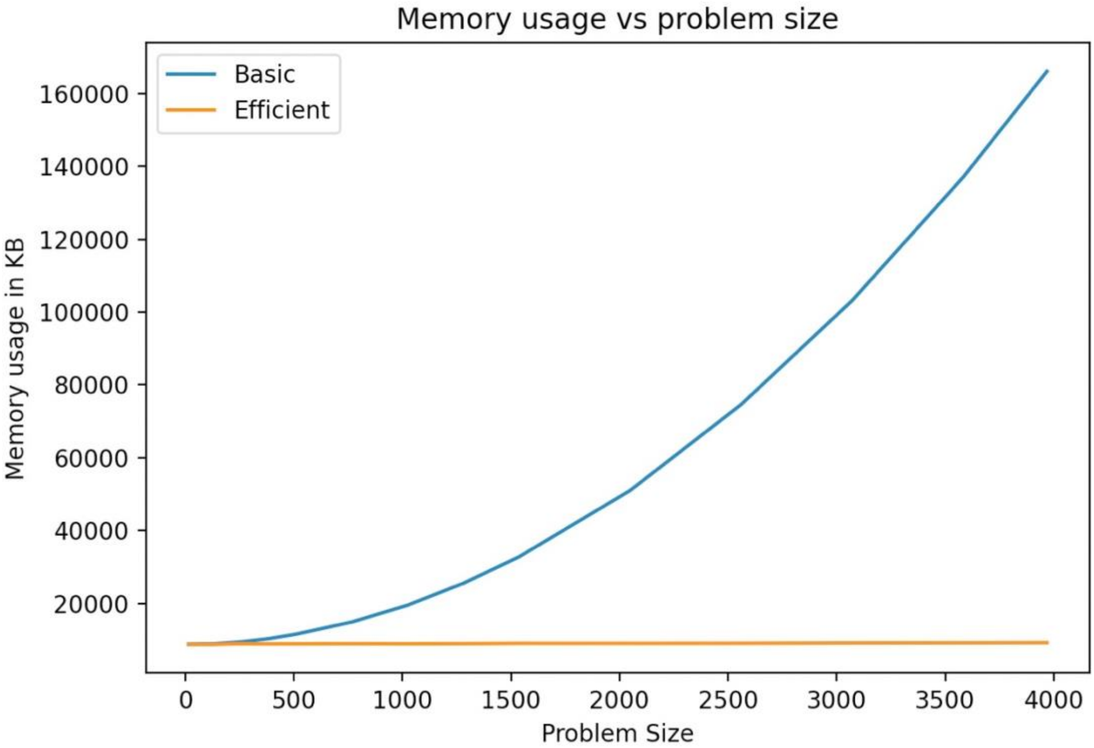
Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.11396408081054688	1.0638236999511719	8796	8788
64	1.2269020080566406	2.2268295288085938	8860	8752
128	5.005121231079102	7.969856262207031	8880	8732
256	16.334056854248047	30.01713752746582	9412	8920
384	37.6439094543457	69.4739818572998	10296	8872
512	66.31302833557129	120.90301513671875	11564	8892
768	148.85520935058594	278.0191898345947	14896	8928
1024	271.76713943481445	492.6936626434326	19504	8896
1280	419.8942184448242	792.4799919128418	25496	8936
1536	616.9121265411377	1092.8592681884766	32744	9028
2048	1117.7570819854736	2029.78515625	50980	9020
2560	1803.8499355316162	3276.2231826782227	74616	9044
3072	2567.2709941864014	4411.396026611328	103220	9144
3584	3396.6751098632812	6197.128057479858	137176	9172
3968	4329.18119430542	7917.374849319458	166112	9216

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)



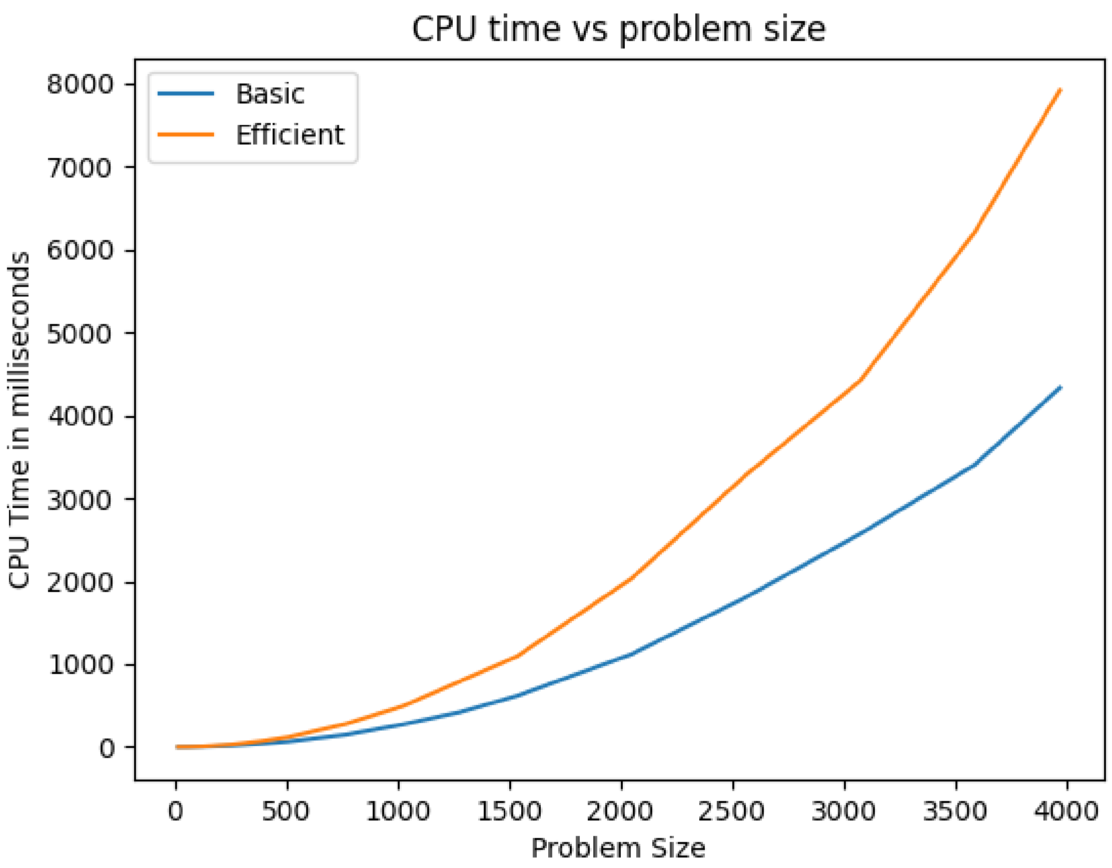
Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)

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>