SUMMARY

USC ID/s:

1127473900 (Changjun Zai)

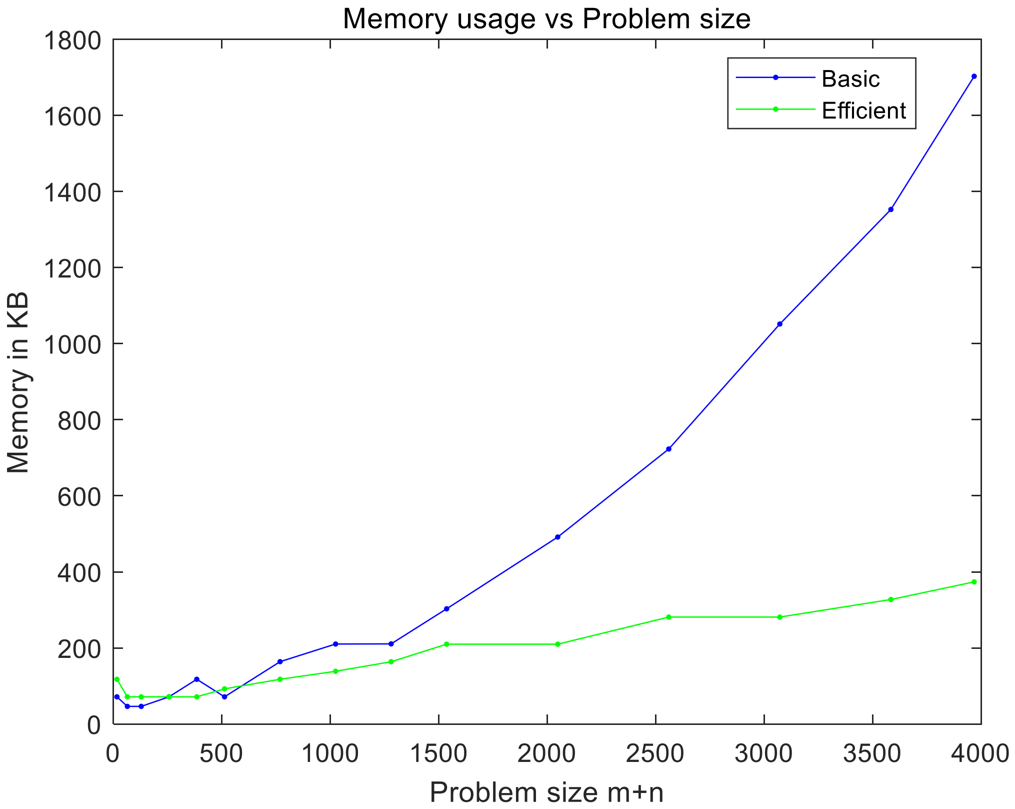
Datapoints

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	1.3212699964642525	1.356470100581646	71.48479999999998	117.6224
64	1.2886799946427345	1.3994299992918968	46.34559999999996	71.48479999999998
128	1.3431398943066597	1.5169899985194206	46.347200000000015	71.4856
256	1.6710399985313416	1.9817000031471252	71.48319999999998	71.48319999999998
384	1.9266299977898598	1.956690102815628	117.62480000000002	71.48400000000001
512	2.46586000174284	2.127999998629093	71.4856	92.4856
768	2.804729901254177	2.554719999432564	163.91600000000003	117.62960000000001
1024	3.198300004005432	2.7468900978565216	210.31199999999998	138.6344
1280	3.502420000731945	3.189549997448921	210.6704	163.7648
1536	3.662059895694256	3.7536599040031433	302.9176	209.9048
2048	4.512850001454353	4.7350500002503395	491.5304	209.90560000000002
2560	4.722819998860359	6.346749998629093	722.9624	281.1984
3072	5.665480099618435	8.0607400983572	1051.6312	281.1992
3584	6.2119400054216385	9.553379900753498	1352.7223999999999	327.3392
3968	8.620460003614426	11.164059899747372	1702.8608	373.6744

Insights

In this project, I use Java to implement the Dynamic Programming solution to the Sequence Alignment problem. I implement basic dynamic programming algorithm and memory efficient version of solution to solve the problem.
The time complexity of basic and memory efficient is both O(mn). For basic algorithm, we have a |m| * |n| matrix to store and update values while traversing string. While for memory efficient algorithm, we use a size of 2 * |m| matrix to do this, and we only need the memory column before to compute the alignment so we only need 2 columns at a time. However, efficient algorithm needs to spend more CPU time when the size of problem is large compared with basic algorithm. Therefore, when we choosing algorithm, we should consider the problem size. If the problem size is large and we can allow for a range of time increases, then we should use the memory efficient algorithm. Otherwise, we can just implement basic algorithm.

Graph1 – Memory vs Problem Size (M+N)



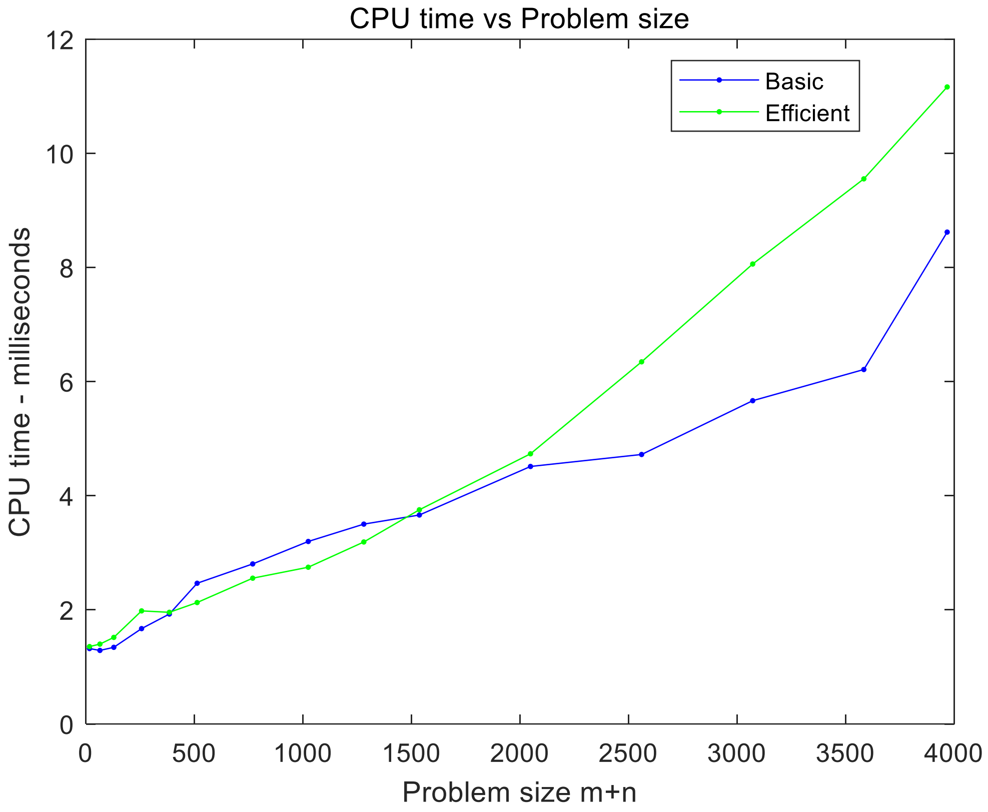
Nature of the Graph (Logarithmic/ Linear/ Polynomial/ Exponential)

Basic: Polynomial
Efficient: Linear

Explanation:

The space complexity of basic solution is O(mn), because we need an 2D array to store the cost.
Whereas, the space complexity of efficient solution is 0(m+n) for dividing dp array into 2 columns and implement recursive calls sequentially.

Graph2 – Time vs Problem Size (M+N)



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

Basic: Polynomial

Efficient: Polynomial

Explanation:

The time complexity of basic solution is O(mn), while the time complexity of efficient solution is also O(mn). However, the actual running time of efficient solution is slower than basic solution. Because efficient solution needs more time to do divide and conquer steps.

Contribution

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

I completed the whole project alone.