**SUMMARY**

## USC ID/s:

## Datapoints

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 0.6439 | 4.7644 | 236.176 | 293.616 |
| 64 | 0.9256 | 7.3558 | 237.696 | 293.616 |
| 128 | 2.0306 | 11.0207 | 475.608 | 727.24 |
| 256 | 8.3059 | 18.4248 | 1109.232 | 1957.2 |
| 384 | 8.5205 | 28.2486 | 2390.488 | 4196.616 |
| 512 | 16.5196 | 31.6887 | 4049.32 | 6952.072 |
| 768 | 31.0286 | 59.6196 | 1506.36 | 2572.664 |
| 1024 | 43.6858 | 84.893 | 4935.872 | 1257.992 |
| 1280 | 41.4266 | 82.2497 | 14919.46 | 11818.16 |
| 1280 | 48.3683 | 78.5944 | 5568.328 | 10155.15 |
| 1536 | 52.4103 | 113.0318 | 13994.7 | 38839.26 |
| 2048 | 80.4164 | 184.2989 | 14753.17 | 19703.28 |
| 2560 | 95.6147 | 191.416 | 37732.27 | 99762.17 |
| 3072 | 123.236 | 256.2836 | 45354.99 | 1708.12 |
| 3584 | 149.5928 | 302.6013 | 110493.4 | 60464.63 |
| 3968 | 172.7917 | 339.3693 | 106361.7 | 152760.3 |
| 6144 | 285.7493 | 575.3012 | 99818.92 | 117547.2 |
| 7680 | 456.5451 | 873.6132 | 134710.3 | 29535.28 |
| 9216 | 606.6692 | 1068.071 | 202017.8 | 81964.68 |
| 10752 | 822.1965 | 1381.544 | 493393.7 | 149192.5 |
| 12288 | 1013.107 | 1769.297 | 636643.3 | 143165 |

## Insights

### Graph1 – Problem Size (M+N) vs Memory

Figure

Figure

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

Basic: Polynomial

Efficient: Linear

#### Explanation:

Based on the first graph above (Figure 1), it initially appears that the Basic and Efficient algorithms have similar memory utilization. However, for our approach, this resided with low sequence input sizes obtained from the project sample test cases. As the size of the DNA sequences increase, we initially observed a polynomial increase in both methods. However, Figure 1 illustrates that the Efficient approach was utilizing more memory than the Basic approach between input sizes of roughly 2000-6000. Moreover, upon sampling additional data, Figure 2 illustrates the Efficient approach trending toward a more linear O(M+N) space to solve the Sequence Alignment Problem. Ensure test validity, we sampled additional data points. The secondary graph above (Figure 2) supports this theory by illustrating when the Efficient algorithm diverges from the Basic algorithm (when M+N is greater than 6000). The Basic algorithm will continue to use a space complexity of O(MN), while the Efficient algorithm space complexity trends towards a O(M+N) complexity, therefore showing a more linear pattern.

### Graph2 – Problem Size (M+N) vs Time

Figure

Figure

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

Basic: Polynomial

Efficient: Polynomial

#### Explanation:

A polynomial pattern was observed for time complexity for both approaches. Similar to memory usage, time complexities for both Basic and Efficient algorithms grow with an increase of input size due to additional memory allocation. Allocating memory requires time to copy memory from cache, to disk, and back again which significantly slow the programs. With an increased input size, the reading and writing operations can significantly impact the program. The Efficient algorithm trades performance for space as it needs to generate more stack frames due to its recursive nature (i.e. the division of DNA sequences into substrings). With the divide-and-conquer approach, a much smaller 2-dimmensional array was required to find the optimal value.

## Contribution

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

<USC ID/s>: <Equal Contribution>