**SUMMARY**

## USC ID/s:

2965751869, 9281603623, 8198269928

## Datapoints

Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in mSec (Basic) | Time in mSec (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 1 | 4 | 0 | 0 |
| 64 | 8 | 12 | 91.752 | 183.504 |
| 128 | 25 | 33 | 367.024 | 825.936 |
| 256 | 76 | 99 | 1835.6 | 3303.12 |
| 384 | 107 | 132 | 4039.568 | 3578.704 |
| 512 | 142 | 152 | 1365.744 | 2954.992 |
| 768 | 147 | 233 | 4780.672 | 2244.152 |
| 1024 | 162 | 218 | 2587.784 | 3716.392 |
| 1280 | 186 | 245 | 6339.76 | 2358.536 |
| 1536 | 201 | 262 | 5059.496 | 1292.144 |
| 2048 | 249 | 551 | 9791.424 | 2485.816 |
| 2560 | 376 | 512 | 14343.41 | 1199.056 |
| 3072 | 325 | 646 | 22953.56 | 2717.064 |
| 3584 | 359 | 745 | 33754.53 | 3125.296 |
| 3968 | 375 | 637 | 50907.06 | 864.48 |

## Insights

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

Figure

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

Basic: Polynomial

Efficient: Linear

#### Explanation:

Based on the graph above (Figure 1), it initially appears that the Basic and Efficient algorithms have similar memory utilization. However, once the sequences increase to an input size of about 2000+, the two methods diverge. Figure 1 illustrates the Efficient approach trending toward a more linear O(M+N) space complexity to solve the Sequence Alignment Problem, while the Basic approach leans towards a polynomial O(M\*N) space complexity. This is caused by the implementation differences between the two methods. While the Basic approach relies on a dynamic programming method by using an array of size M\*N to calculate the optimal value and align the sequences, the Efficient Approach utilizes a clever divide-and-conquer method. This method’s purpose is to subdivide and replace the M\*N array in the dynamic programming approach with an M\*2 array in the efficient algorithm to calculate a subset of optimal values. When the entire space has been traversed, the concatenated subset of optimal values will produce the final optimal value for the full sequence alignment. Overall, the efficient was successful in computing the value of the optimal alignment utilizing O(M+N) space by sacrificing some of its runtime efficiency.

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

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 time necessary for memory allocation. Allocating memory requires time to copy memory from cache, to disk, and back again which significantly slows the programs. With an increased input size, the reading and writing operations can significantly impact the program. However, in Figure 2, it is noticeable that the Efficient algorithm runs slightly slower than the Basic given a large enough input size. The Efficient Approach trades performance for space as it needs to generate and complete the execution of additional stack frames due to its recursive nature (i.e. the division of DNA sequences into substrings via the divide-and-conquer approach). Overall, it was observed that this difference in runtime is negligible considering the space efficiency achieved as illustrated in Figure 1.

## 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