**SUMMARY**

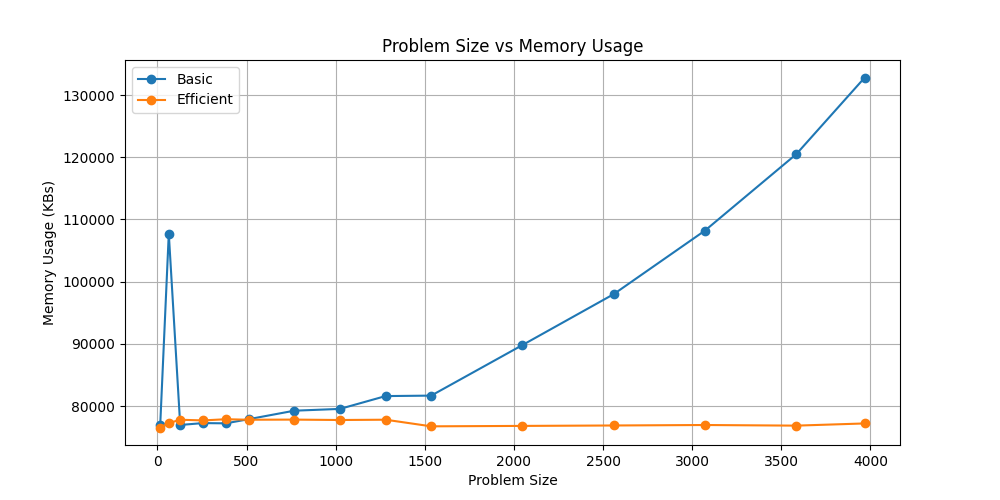
## USC ID/s - *Matthew Schulz*: , *Ryan Marr*: , *Vedant Raval*: 4328073870

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| M+N | Time in MS (Basic) | Time in MS (Efficient) | Memory in KB (Basic) | Memory in KB (Efficient) |
| 16 | 0.00 | 0.00 | 77024 | 76576 |
| 64 | 1.00 | 4.00 | 107744 | 77276 |
| 128 | 4.07 | 7.99 | 76984 | 77824 |
| 256 | 16.03 | 28.01 | 77284 | 77712 |
| 384 | 37.05 | 62.51 | 77244 | 77904 |
| 512 | 71.65 | 111.65 | 77912 | 77824 |
| 768 | 185.26 | 247.22 | 79276 | 77844 |
| 1024 | 283.22 | 448.46 | 79564 | 77780 |
| 1280 | 452.35 | 711.24 | 81624 | 77828 |
| 1536 | 677.71 | 1059.61 | 81700 | 76764 |
| 2048 | 1216.81 | 1926.92 | 89828 | 76832 |
| 2560 | 1919.74 | 2933.79 | 98008 | 76900 |
| 3072 | 2745.08 | 4237.79 | 108228 | 76976 |
| 3584 | 3721.50 | 5817.31 | 120500 | 76876 |
| 3968 | 4586.64 | 7279.02 | 132816 | 77228 |

## Datapoints

## Insights

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



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

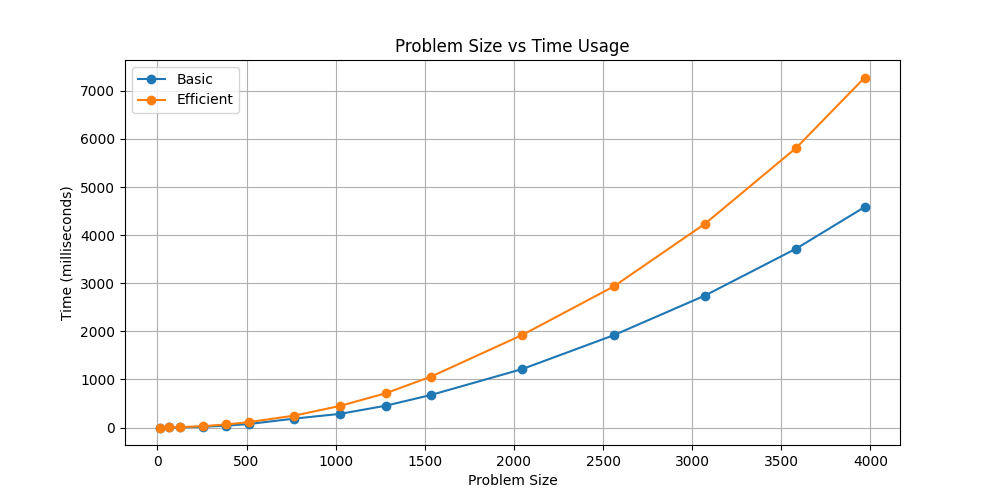
Basic: Polynomial

Efficient: Constant (?)

#### Explanation:

- TODO: Check if we should be getting a constant memory usage for the efficient algorithm

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



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

Basic: Polynomial

Efficient: Polynomial

#### Explanation:

* Given the task of computing the optimal alignment cost between two input strings of X and Y, the *Efficient* algorithm essentially breaks the Y string into two sub-parts YL and YR by choosing a split point in the string Y which minimizes the sum of alignment costs between two equal halves of X and the two unequally broken sub-strings of Y. This can be done using Dynamic programming.
* Suppose the length of X is m and the length of Y is n. Then the above operation could be done in C\*(mn) operations, which is of the same order as the time taken for the *Basic* algorithm.
* Once we have obtained the sub-strings XL, XR, YL, and YR - the *Efficient* algorithm would now proceed with the pairs of (XL, YL) and (XR, YR) and repeat the same steps. In this second level, the algorithm would take C\*(mn)/2. The third level would require C\*(mn)/4 steps and so on…
* We know that the sum of the sequence - 1 + 1/2 + 1/22 + … is a constant and thus the *Efficient* algorithm would still require O(mn) operations.
* Thus, the graph for both the algorithms would be expected to be Polynomial in nature and it is what we observe indeed.
* Furthermore, if we look at the ratio of the time taken by the two algorithms for each value of problem size under consideration, then we can see that indeed the ratio barely changes and stays within a range of [1.5, 2.0], further validating our argument that both the algorithms grow at the same rate.

## 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>