

SUMMARY

USC ID/s:

Joohan Lee: 7711577119

Hee Ji Park: 4090715830

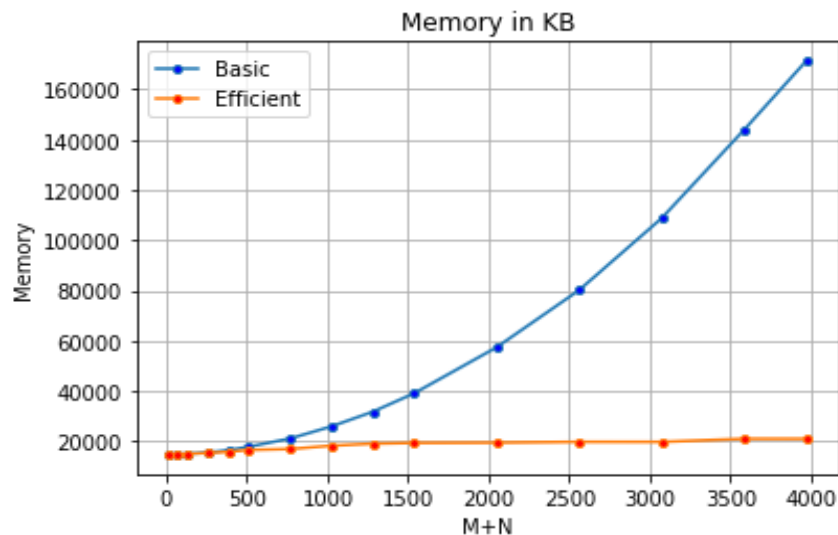
Claire Yuen: 3510645665

Datapoints :

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.0	0.0	14776	14880
64	1.6655921936035156	6.744384765625	14852	14932
128	2.902984619140625	3.1816959381103516	14976	14988
256	10.470867156982422	17.392635345458984	15488	15316
384	16.231060028076172	32.98830986022949	16416	15784
512	27.696847915649414	101.36103630065918	17816	16524
768	75.57296752929688	146.44169807434082	21068	16892
1024	116.4557933807373	294.4614887237549	25880	18212
1280	164.57080841064453	364.3190860748291	31708	19036
1536	263.80038261413574	560.8949661254883	39036	19340
2048	454.88762855529785	954.3516635894775	57356	19436
2560	723.3548164367676	1899.2176055908203	80104	19748
3072	1045.4788208007812	3207.106113433838	108720	19732
3584	1601.8283367156982	6367.202043533325	143952	20968
3968	2021.9225883483887	12326.189756393433	171256	20960

Insights

Graph1 – Memory vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

Efficient: Linear

Explanation:

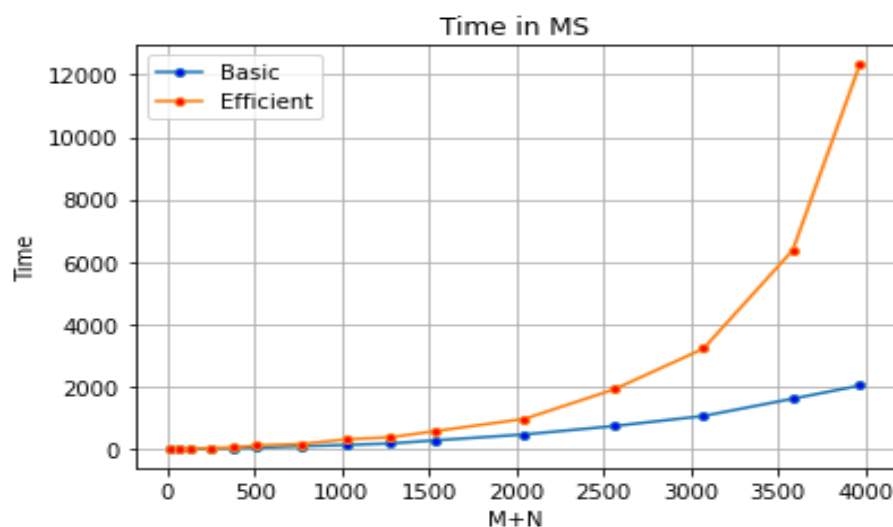
1. The space complexity of Basic algorithm: $O(mn)$

For the basic algorithm, as problem size goes up, the used memory grows rapidly. We used a whole two-dimensional array to store values of the solution. Thus, as the problem size grows up, the used memory for the two-dimensional array grows up as well.

2. The space complexity of Efficient algorithm: $O(m+n)$

On the other hand, we can bring the space requirement down to linear $O(m+n)$. We applied recursive calls sequentially and reused the working space from one call to the next. Thus, since we only worked on one recursive call at a time, the total space is $O(m+n)$. As a result, the above line of efficient algorithm graph increases linearly closer to the x-axis than the basic one.

Graph2 – Time vs Problem Size (M+N)



Nature of the Graph (Logarithmic/ Linear/ Exponential)

Basic: Polynomial

Efficient: Polynomial

Explanation:

1. The time complexity of Basic algorithm: $O(mn)$

We iterated through the whole characters of two generated strings, so it takes $O(m*n)$ time. Thus, as you can see, the above graph has the shape of $y = k*m*n$ which is polynomial. As the problem size (M+N) is increased, the total time is increased as well.

2. The time complexity of Efficient algorithm: $O(mn)$

Likewise, we iterated through the whole characters of two generated strings, so it also takes $O(m*n)$ time. However, we need more workload to use the Divide and conquer algorithm. For example, we should split the problem into subproblems and combine them again. Therefore, the graph of the memory-efficient algorithm increases more steeply than the basic one.

Contribution

Joohan Lee: 7711577119 <Equal Contribution>

Hee Ji Park: 4090715830 <Equal Contribution>

Claire Yuen: 3510645665 <Equal Contribution>