SUMMARY

USC ID/s:

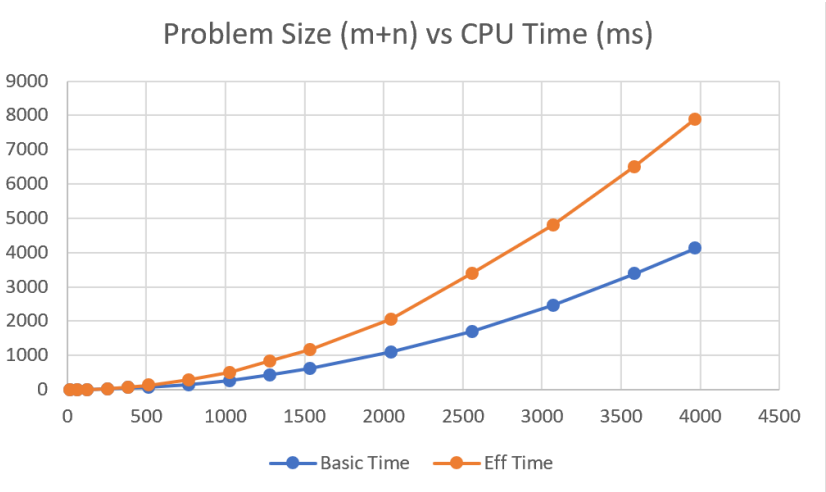
7661563090
2233295549

M+N	Time in MS (Basic)	Time in MS (Efficient)	Memory in KB (Basic)	Memory in KB (Efficient)
16	0.24414063	0.494241714	10248	10360
64	1.32203102	2.685070038	10244	10344
128	4.30631638	8.888721466	10460	10424
256	16.6618824	33.13994408	10956	10420
384	36.4830494	74.08571243	11788	10492
512	67.8589344	129.7528744	12936	10400
768	150.214911	291.0022736	15804	10468
1024	267.032146	511.3811493	17056	10524
1280	420.334101	830.5909634	18528	10644
1536	618.516922	1177.844048	19700	10628
2048	1090.56783	2060.583115	23504	10616
2560	1686.54728	3401.557207	28380	10736
3072	2451.60294	4807.706118	34372	10912
3584	3374.41087	6499.275923	41088	10820
3968	4115.34119	7894.522905	38660	10872

Datapoints

Insights

Graph1 – Memory vs Problem Size (M+N)



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

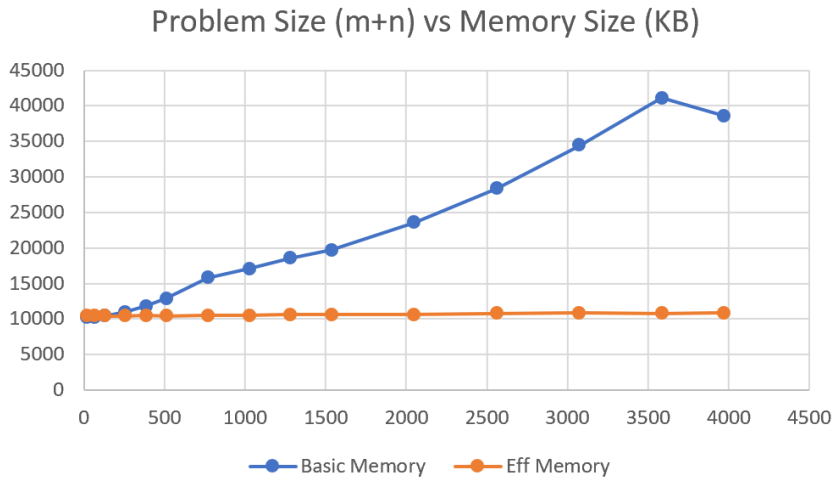
Basic: Polynomial

Efficient: Polynomial

Explanation:

The relation for both the Basic Function and the Memory Efficient Function in the Problem Size vs CPU Time graph can be described as being polynomial. Since with low problem size, the time for both Basic and Efficient functions are quite low, and as the problem size increases, both the Basic and Efficient CPU time also increase, albeit with Basic being much more time efficient than the Memory Efficient function.

Graph2 – Time vs Problem Size (M+N)



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

Basic: Linear

Efficient: Logarithmic (Basically Constant)

Explanation:

The relation between the basic and efficient functions for the Problem Size vs Memory Size graph, on the other hand, is linear and logarithmic respectively. We can conclude that while the Basic Function runs faster than the Efficient Function linearly, the Efficient Function makes up for the longer runtime with almost near constant memory usage.

Contribution

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

(7661563090) Jacob Jun: basic_3.py, shell files

(2233295549) Joseph Yoon: efficient_3.py, Data and Analysis