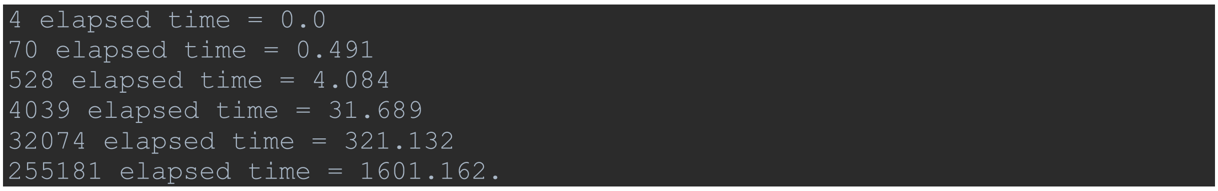
Friday 5 February 2021

COMP 20290 – Algorithms Lab 1

Empirical Analysis.

Results from testing ThreeSumA.

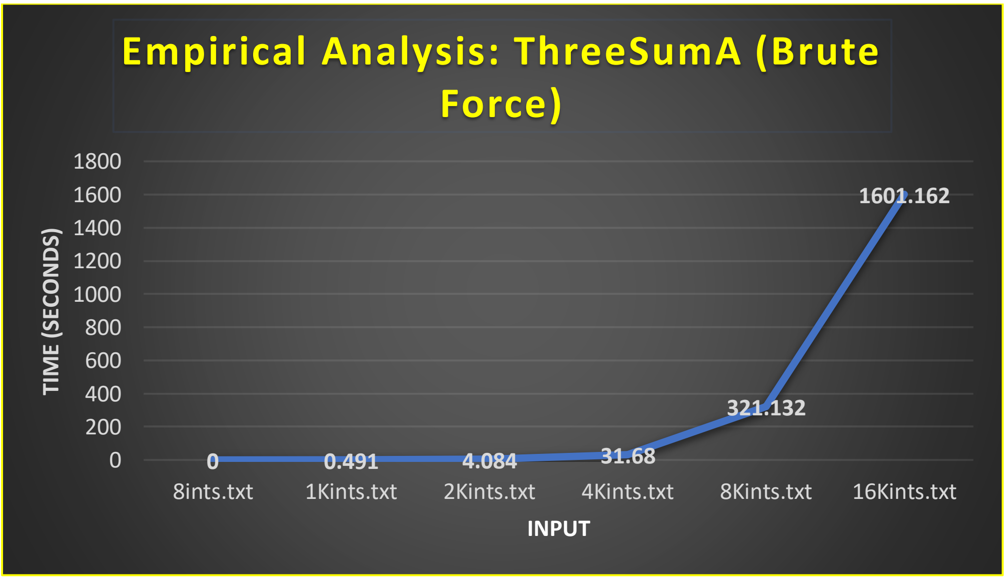


```
4 elapsed time = 0.0
70 elapsed time = 0.491
528 elapsed time = 4.084
4039 elapsed time = 31.689
32074 elapsed time = 321.132
255181 elapsed time = 1601.162.
```

Results inputted into table to be used to graph data.

Algorithm	Input	Time (seconds)	# of inputs
ThreeSumA	8ints.txt	0	4
	1Kints.txt	0.491	70
	2Kints.txt	4.084	528
	4Kints.txt	31.68	4039
	8Kints.txt	321.132	32074
	16Kints.txt	1601.162	255181

Running time data from ThreeSumA graphed.



Friday 5 February 2021

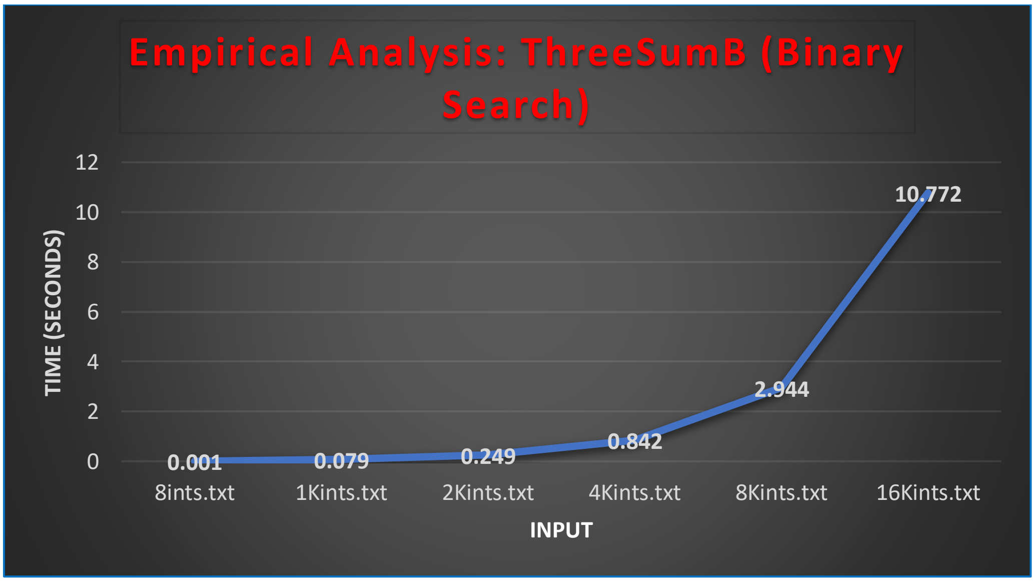
Result from testing ThreeSumB.

```
4 elapsed time = 0.001
70 elapsed time = 0.079
528 elapsed time = 0.249
4039 elapsed time = 0.842
32074 elapsed time = 2.944
255181 elapsed time = 10.772.
```

Results inputted into table to be used to graph data.

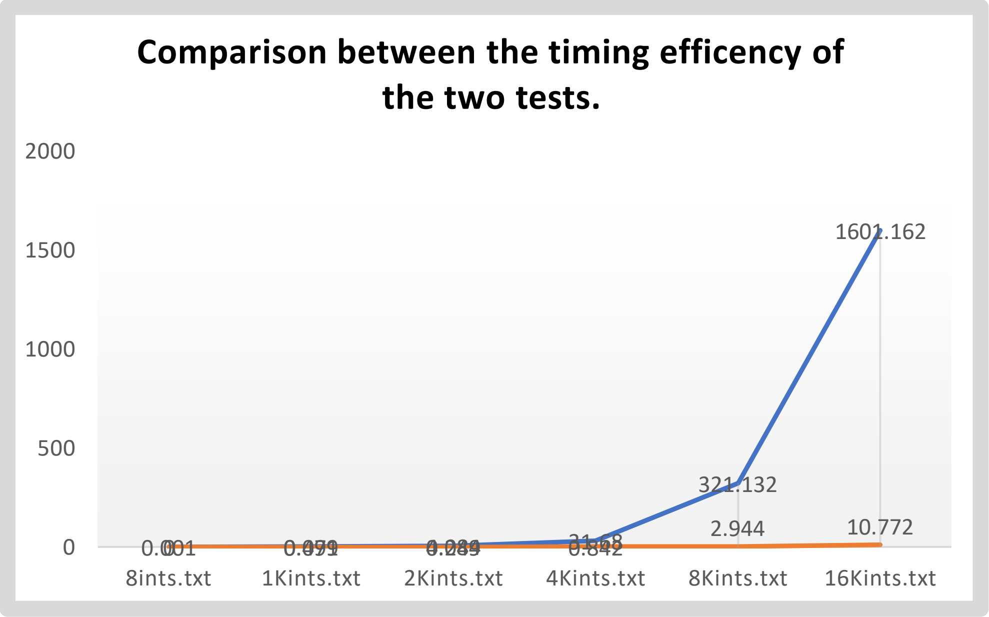
Algorithm	Input	Time	# of inputs
ThreeSumB	8ints.txt	0.001	4
	1Kints.txt	0.079	70
	2Kints.txt	0.249	528
	4Kints.txt	0.842	4039
	8Kints.txt	2.944	32074
	16Kints.txt	10.772	255181

Running time data from ThreeSumB graphed.



Friday 5 February 2021

Comparing Data from both tests.



Questions

Q1. Which algorithm performs better (i.e., take less time in relation to the size of the input)?

According to the comparison of time efficiency between the two algorithms I can confidently conclude that the algorithm utilised in ThreeSumB (Binary Search) performed better.

Q2. Why do you think this is the case?

The algorithm used in ThreeSumB outperformed its counter part due to the fact a Binary search approach was implemented in constrast to the Brute Force, array of integers, approach we can see in ThreeSumA.

These choices made in ThreeSumB cut down run time exponentially.