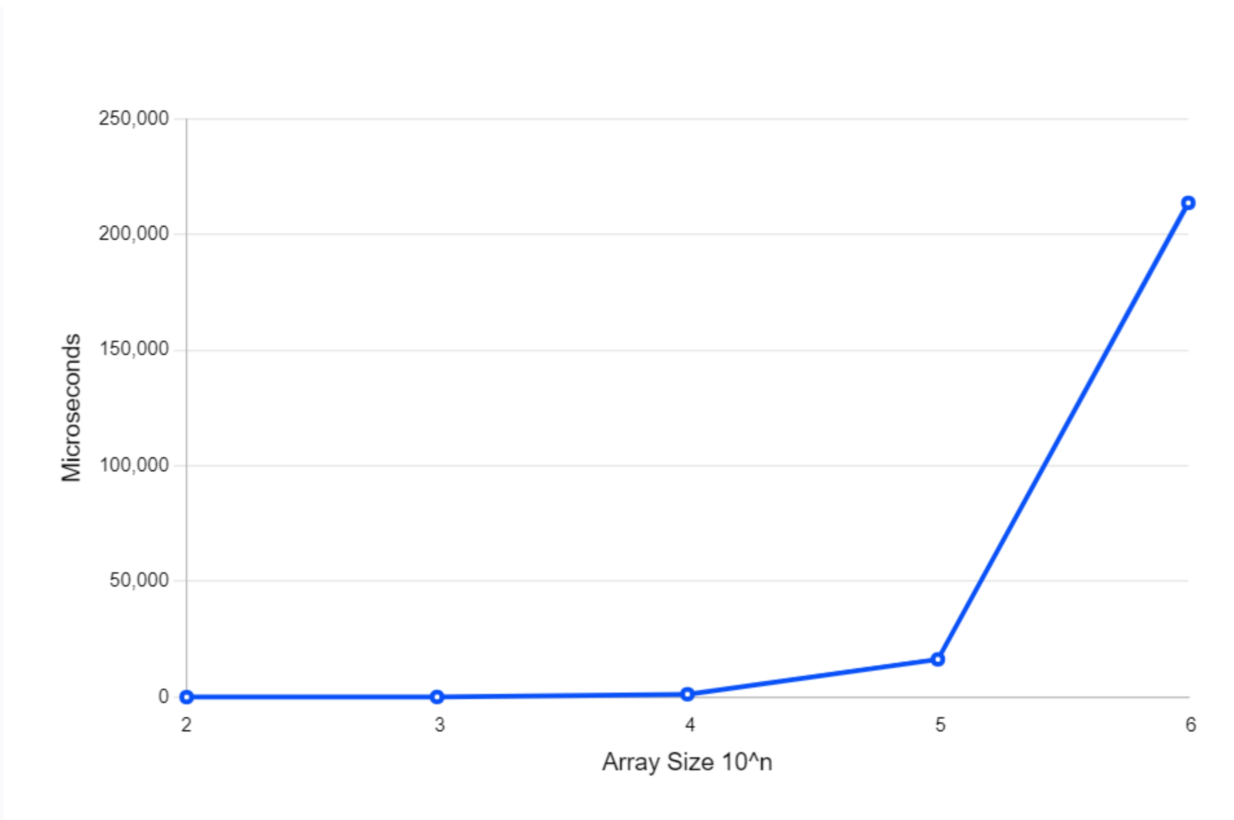
*****NOTE: Updated work for bonus points starts on page 17!**

Random Array
Shell Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	74	18.5
4	10,000	1,248	16.865
5	100,000	16,334	13.088
6	1,000,000	213,673	13.081

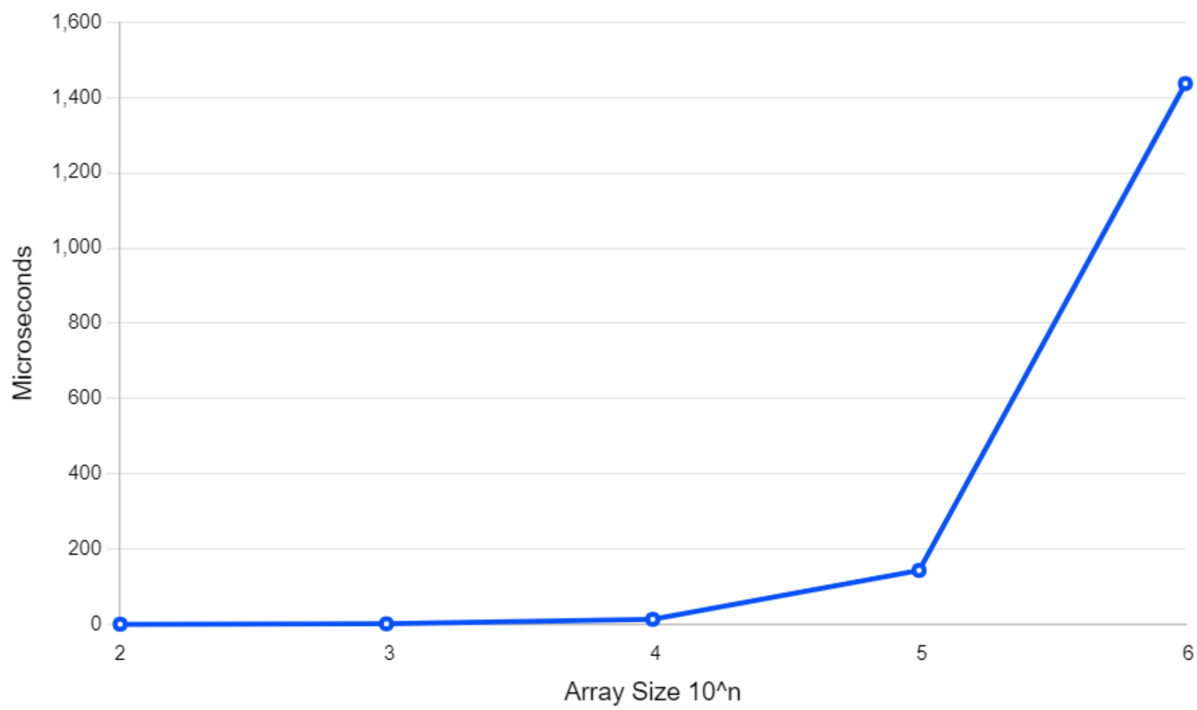


((18.5/16.865)+(16.865/13.088)+(13.088/13.081)) ≈ 1.13
By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the average case of shell sort in this example is

θ(n^1.13).

Insertion Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	13	13
5	100,000	143	11
6	1,000,000	1,437	10.049



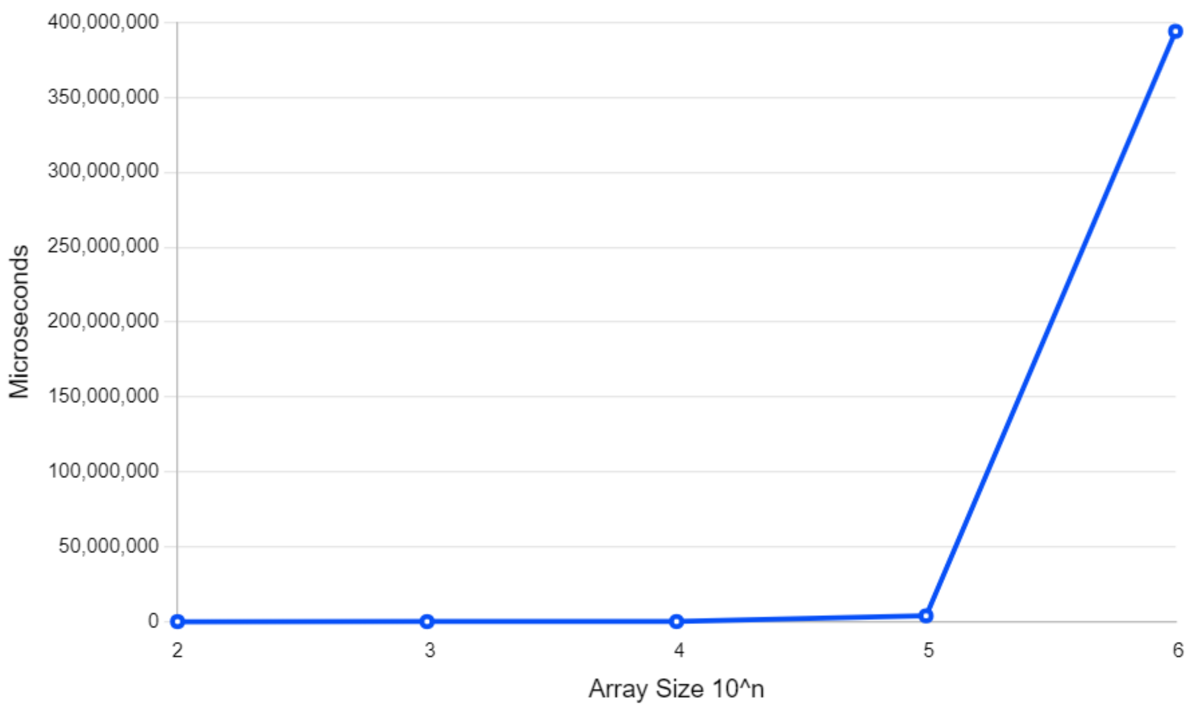
((1/13)+(13/11)+(11/10.049)) ≈ 1.07

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the average case of shell sort in this example is

θ(n^1.07).

Selection Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	387	96.75
4	10,000	40,664	105.075
5	100,000	3,853,388	94. 762
6	1,000,000	394,076,849	102.268

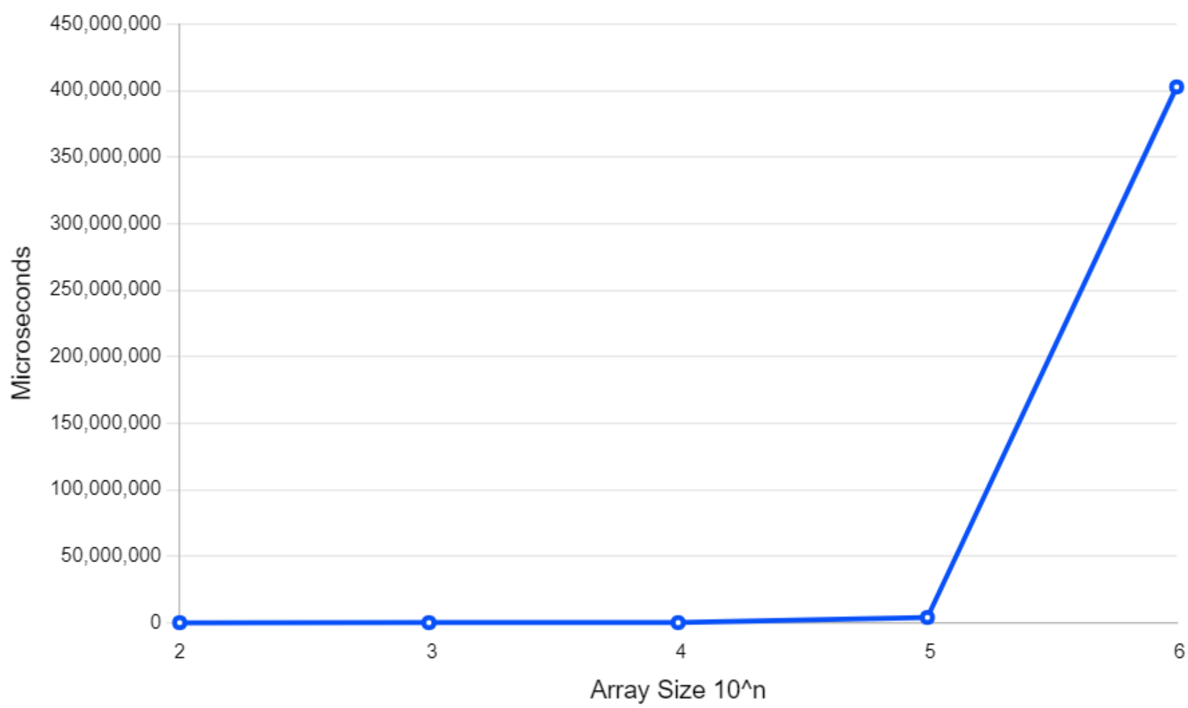


((96.75/105.075)+(105.075/94.762)+(94.762/102.268)) ≈ 1

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the average case of shell sort in this example is θ(n).

Bubble Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	1	-
3	1,000	392	392
4	10,000	39,551	100.895
5	100,000	3,937,929	99.566
6	1,000,000	402,697,164	102.261

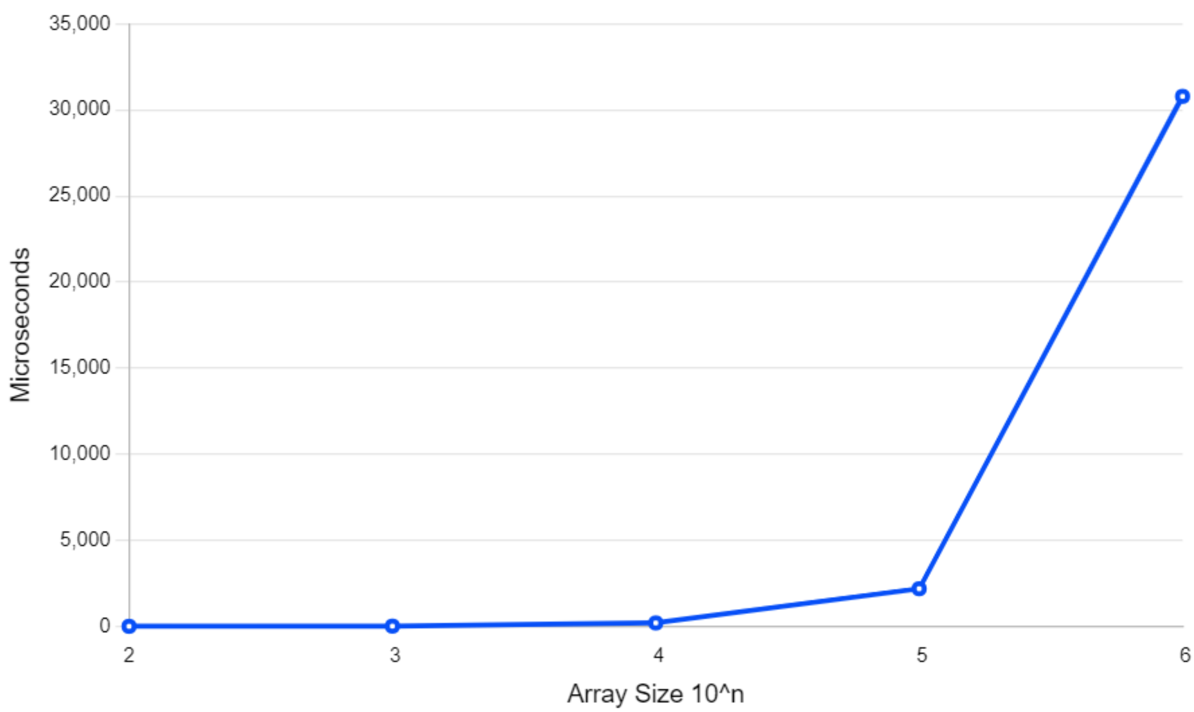


((392/100.895)+(100.895/99.566)+(99.566/102.261)) ≈ 2

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the average case of shell sort in this example is θ(n^2).

Increasing Array
Shell Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	1	-
3	1,000	11	11
4	10,000	196	17.818
5	100,000	2,184	11.143
6	1,000,000	30,794	14.10

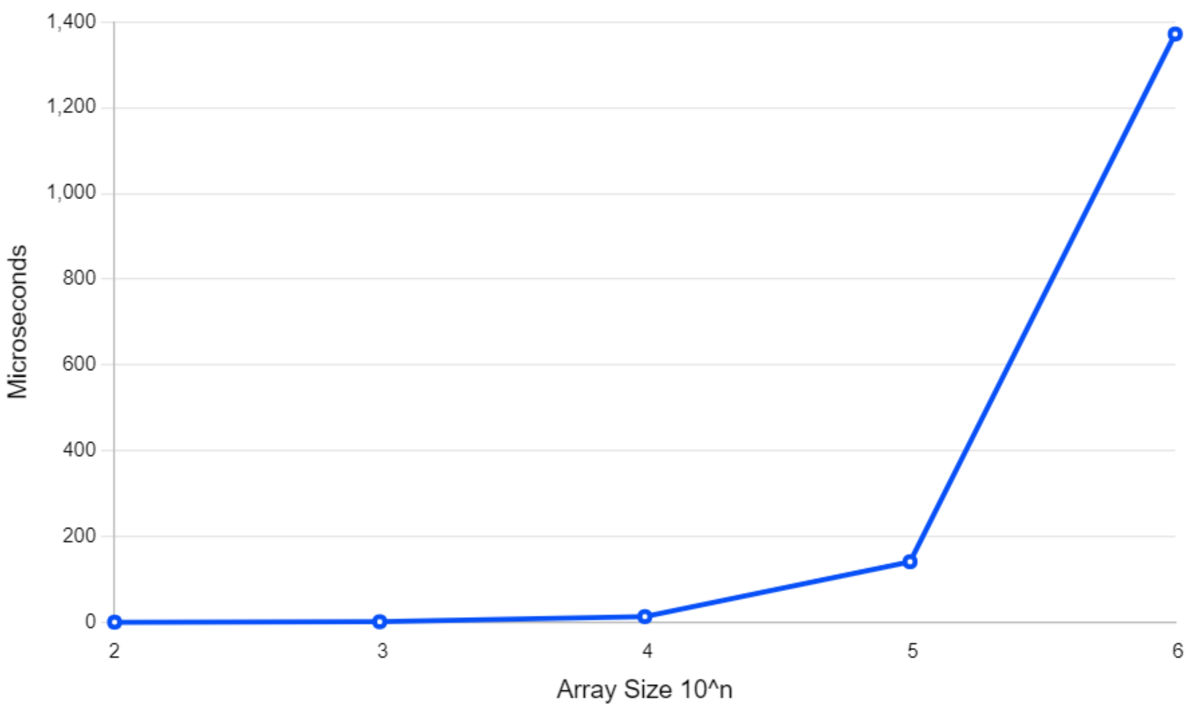


((11/17.818)+(17.818/11.143)+(11.143/14.10)) ≈ 1

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the best case of shell sort in this example is θ(n).

Insertion Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	13	13
5	100,000	141	10.846
6	1,000,000	1,372	9.73

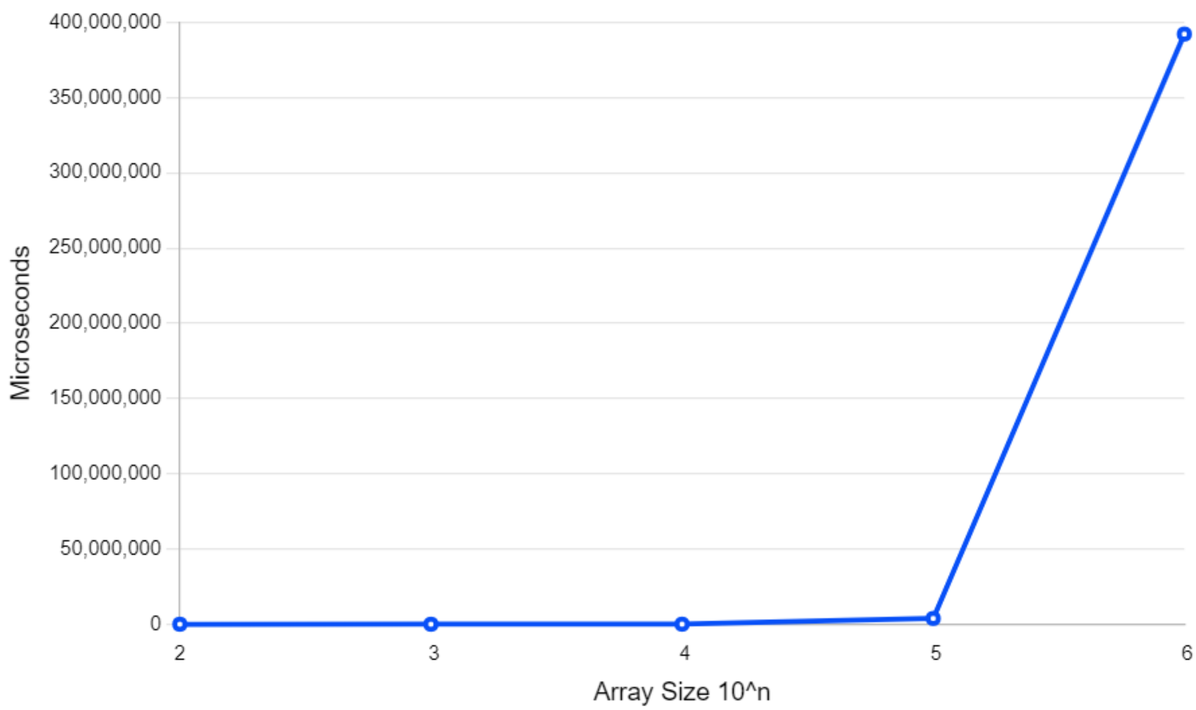


((1/13)+(13/10.846)+(10.846/9.73)) ≈ 0.8

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the best case of shell sort in this example is θ(n^0.8).

Selection Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	403	100.75
4	10,000	40,664	100.903
5	100,000	3,840,530	94.445
6	1,000,000	392,183,811	102.117

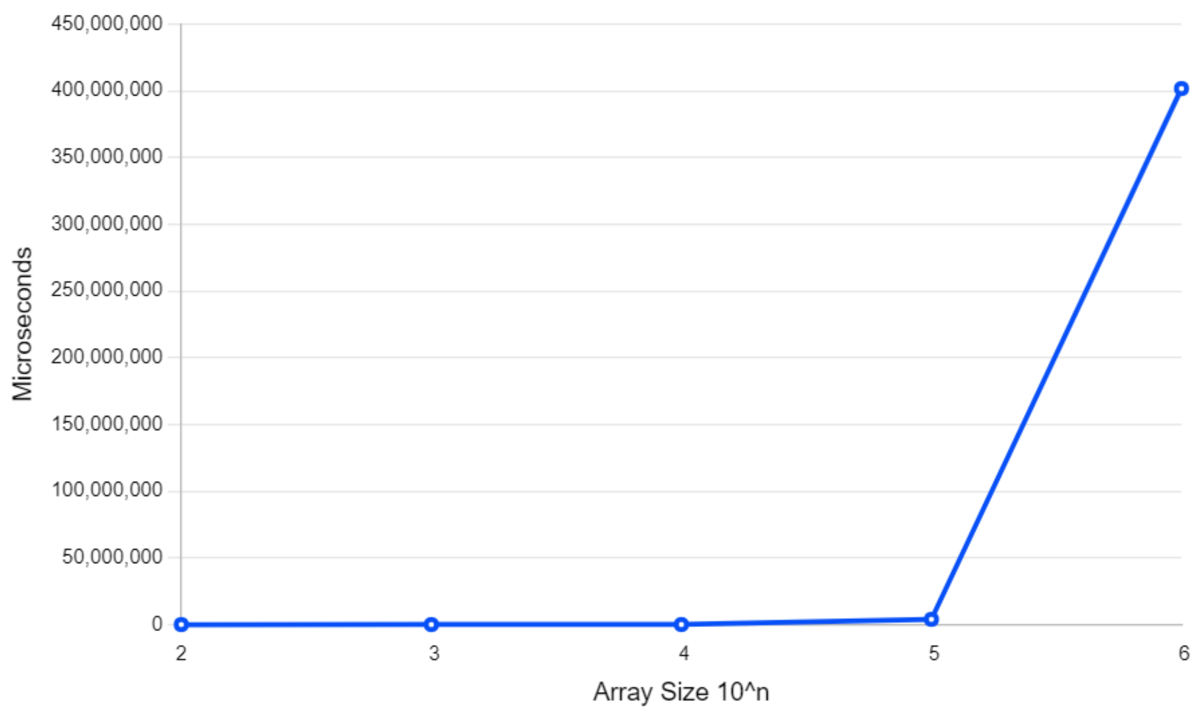


((100.75/100.903)+(100.903/94.445)+(94.445/102.117)) ≈ 1

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the best case of shell sort in this example is θ(n).

Bubble Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	391	97.75
4	10,000	40,401	103.327
5	100,000	3,932,749	97.343
6	1,000,000	401,614,645	102.121

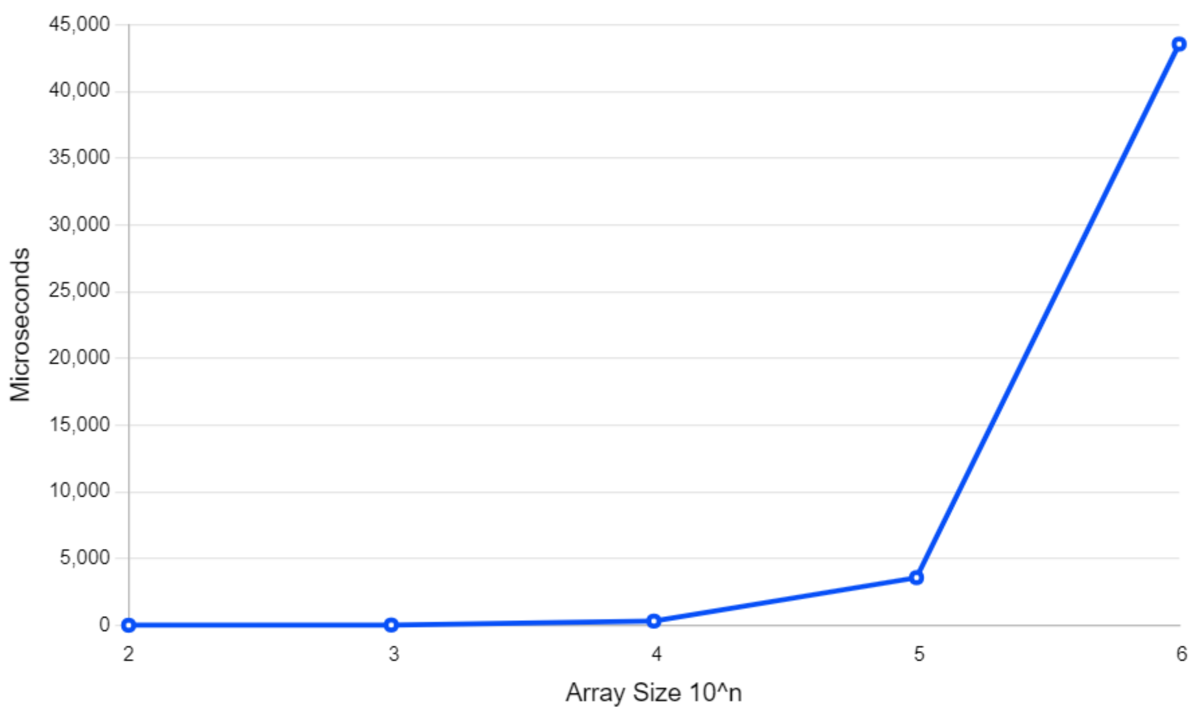


((97.75/103.327)+(103.327/97.343)+(97.343/102.121)) ≈ 1

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the best case of shell sort in this example is θ(n).

Decreasing Array
Shell Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	1	-
3	1,000	19	19
4	10,000	312	16.421
5	100,000	3,569	11
6	1,000,000	43,545	439

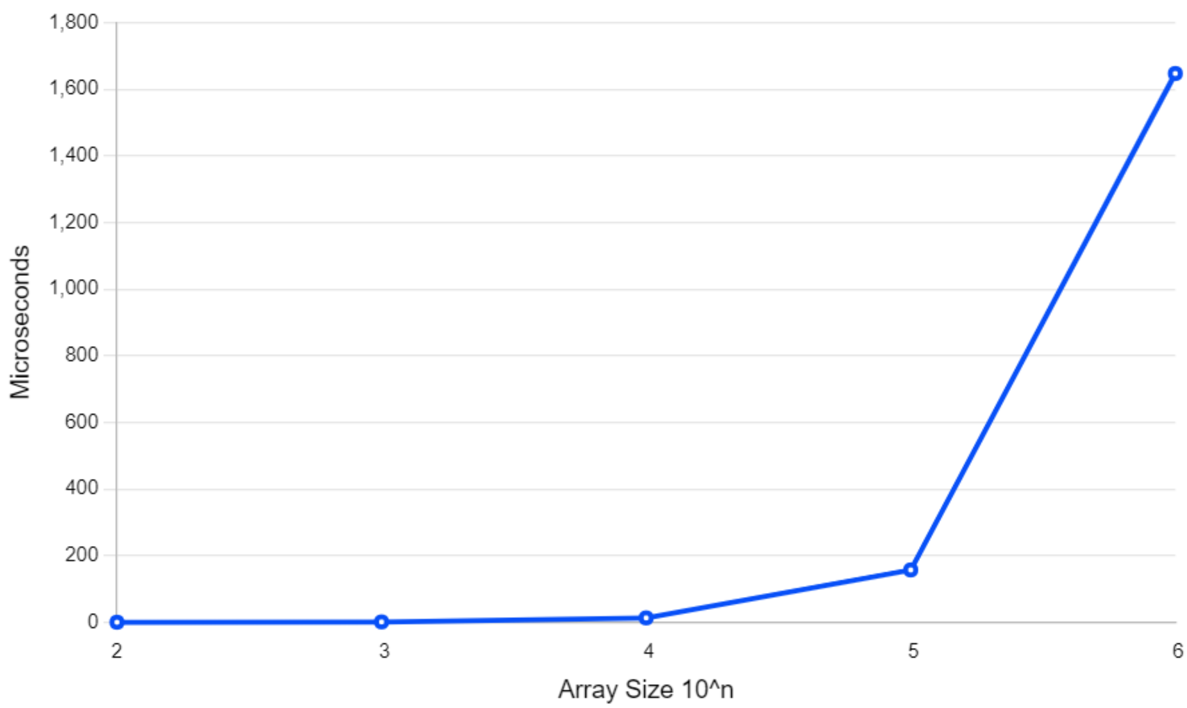


((19/16.421)+(16.421/11)+(11/439)) ≈ 0.9

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the worst case of shell sort in this example is θ(n^0.9).

Insertion Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	13	13
5	100,000	157	12.077
6	1,000,000	1,647	10.49

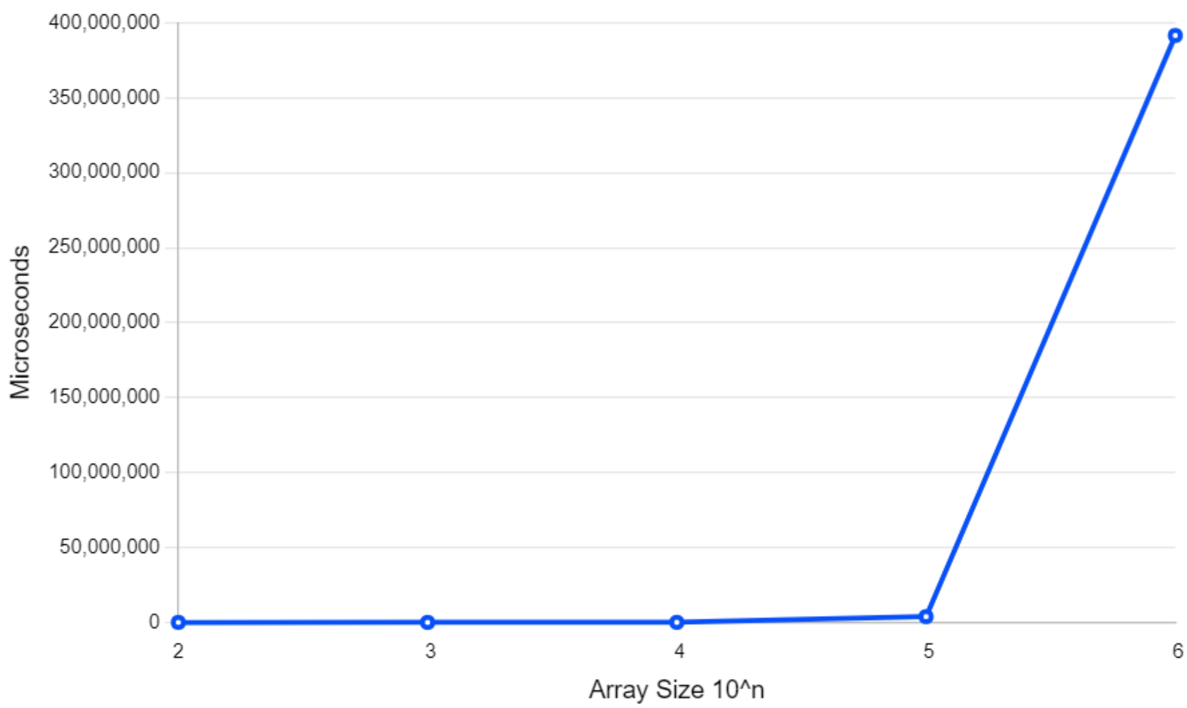


((1/12)+(13/12.077)+(12.077/10.49)) ≈ 1

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the worst case of shell sort in this example is θ(n).

Selection Sort

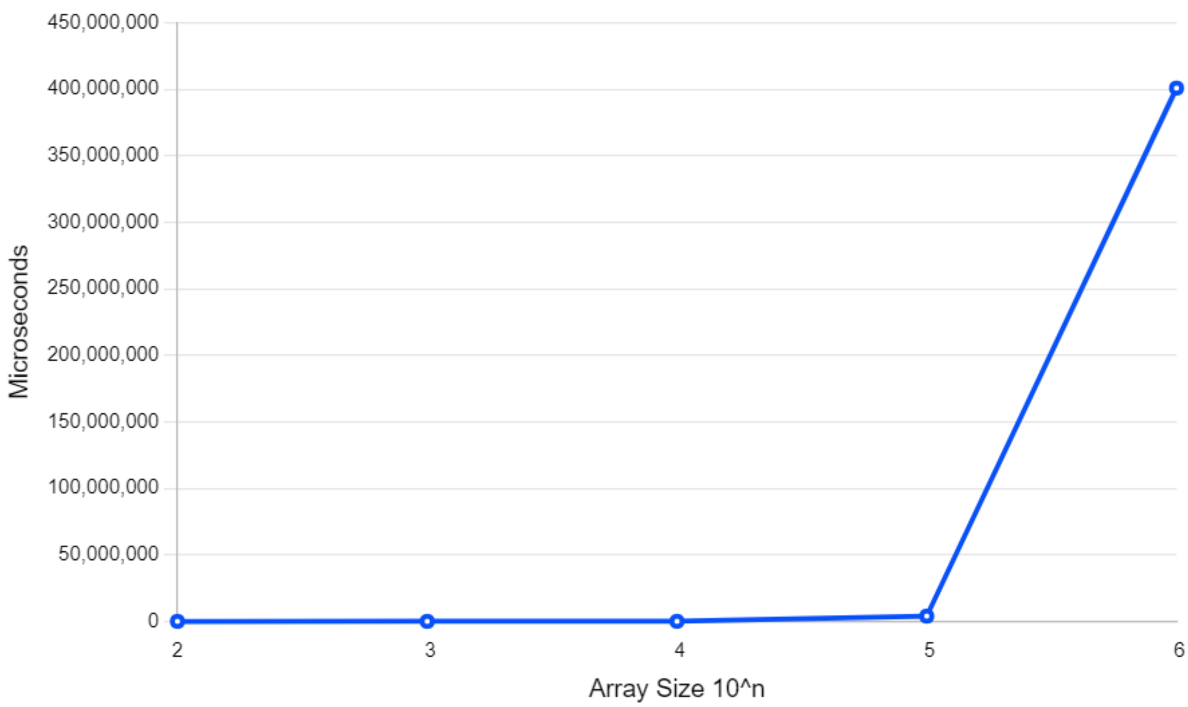
Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	383	95.75
4	10,000	39,054	101.969
5	100,000	3,842,919	98.40
6	1,000,000	391,524,330	101.882



((19/16.421)+(16.421/11)+(11/439)) ≈ 1
By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the worst case of shell sort in this example is θ(n^1).

Bubble Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	392	98
4	10,000	40,606	103.587
5	100,000	3,936,829	96.952
6	1,000,000	400,799,760	101.808



((98/103.587)+(103.587/96.952)+(96.952/101.808)) ≈ 1

By taking the quotient of each adjacent factor increase in time, then averaging them, we find that the time complexity of the worst case of shell sort in this example is θ(n).

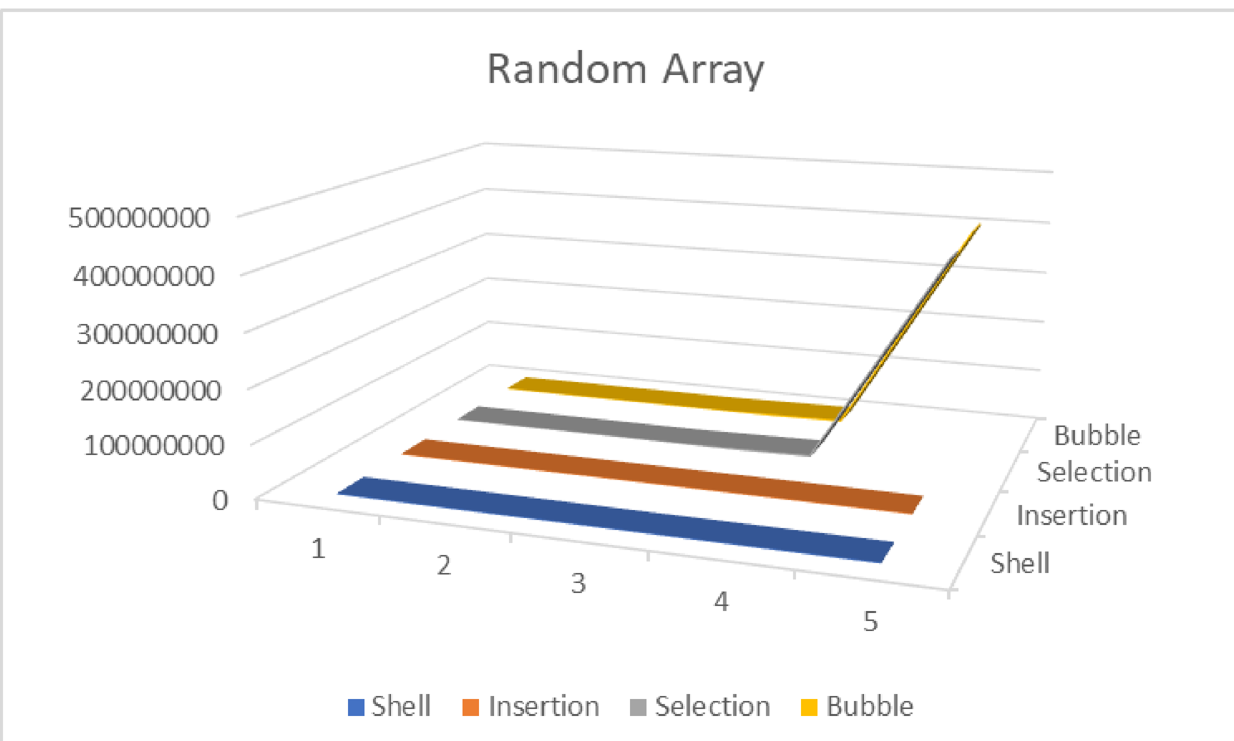
Empirical/Mathematical Complexity

The time it takes to sort the random array represents the average case time efficiency of the sorting algorithms, the time it takes to sort the increasing array represents the best case, and the time it takes to sort the decreasing array represents the worst case.

The following tables and graphs show a comparison between the theoretical time complexity and the actual (approximate) time complexity of each sorting algorithm based on this experiment:

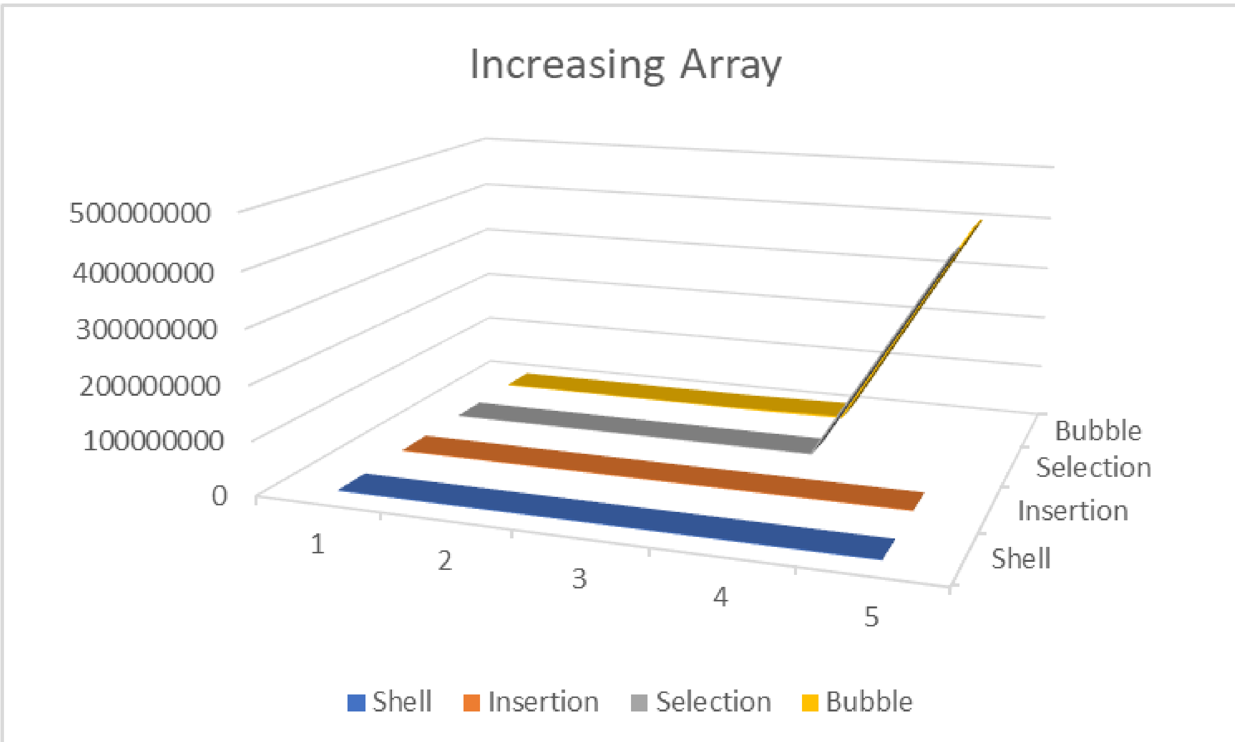
Random (Average Case)

Algorithm	Theoretical Time Complexity	Actual Time Complexity
Shell Sort	θ(n log(n))	θ(n^1.13)
Insertion Sort	θ(n^2)	θ(n^1.07)
Selection Sort	θ(n^2)	θ(n)
Bubble Sort	θ(n^2)	θ(n^2)



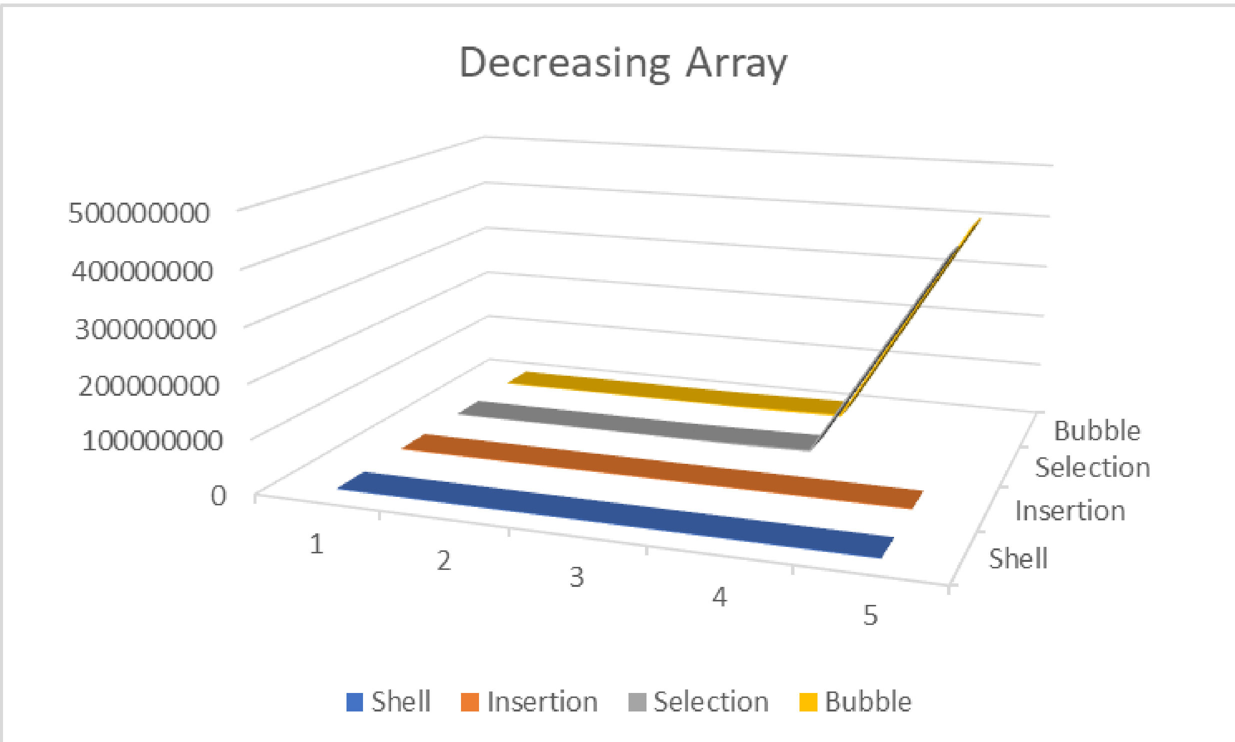
Increasing (Best Case)

Algorithm	Theoretical Time Complexity	Actual Time Complexity
Shell Sort	Ω(n log(n))	Ω(n)
Insertion Sort	Ω(n)	Ω(n^0.8)
Selection Sort	Ω(n^2)	Ω(n)
Bubble Sort	Ω(n)	Ω(n)



Decreasing (Worst Case)

Algorithm	**Theoretical Time Complexity**	**Actual Time Complexity**
Shell Sort	O(n^2)	O(n^0.9)
Insertion Sort	O(n^2)	O(n)
Selection Sort	O(n^2)	O(n)
Bubble Sort	O(n^2)	O(n)



Notes

I do not know if the way I calculated the empirical time complexity is correct. In some cases, it seems to match the mathematical time complexity very well, but in other cases, it seems way off. I suppose the way to figure out if it is correct or not would be to repeat the experiment.

What threw me off was the scale of the line charts I created. Because the size of the arrays range from one hundred to one million, it is difficult to put in one picture, especially because the gaps between 10^2 and 10^3, 10^3 and 10^4, etc. are not the same.

The way the individual charts look make it seem like all of them have the time complexity of 10^2. The way the grouped charts look make it seem like shell and insertion sort are always linear and selection and bubble sort are always quadratic, but when compared to a Big-O Time Complexity Chart from my CSC 1310 notes, and taking into account mentally the width of the graphs (as opposed to the misleading visuals),it made more sense. I had to realize how wide each graph was really getting, and that they were not as close to the Y-axis as they seem to be.

Updated work for bonus points starts here!

Code

Issue: Passing the already sorted arrays into the next sorting function.

Updates:

1. New Function - Lines 201 through 208
2. New Temporary Array Definitions - Lines 236 and 237
3. For Loops to Revert Arrays Back to Their Unsorted States
4. New Delete [] Statements - Lines 288 and 289

Report

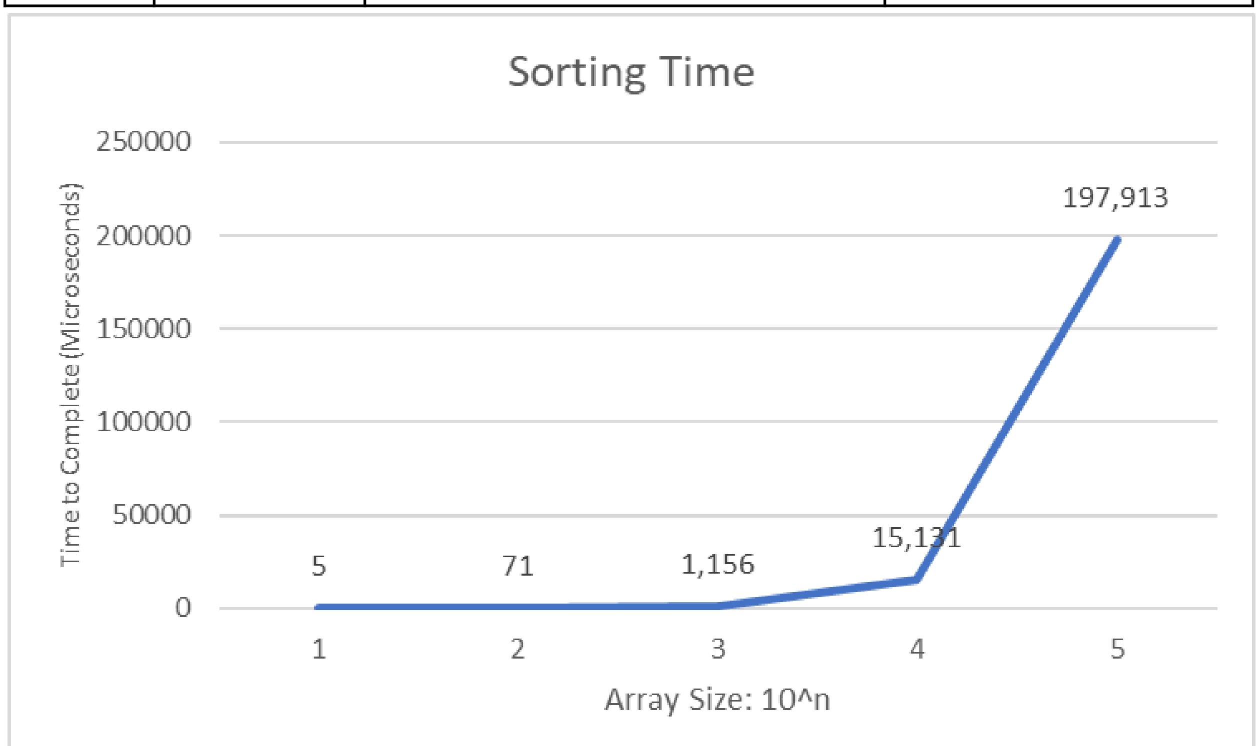
Issue: Averaged the random, increasing, and decreasing array sorting times instead of analyzing them all individually.

Updates:

Random Array

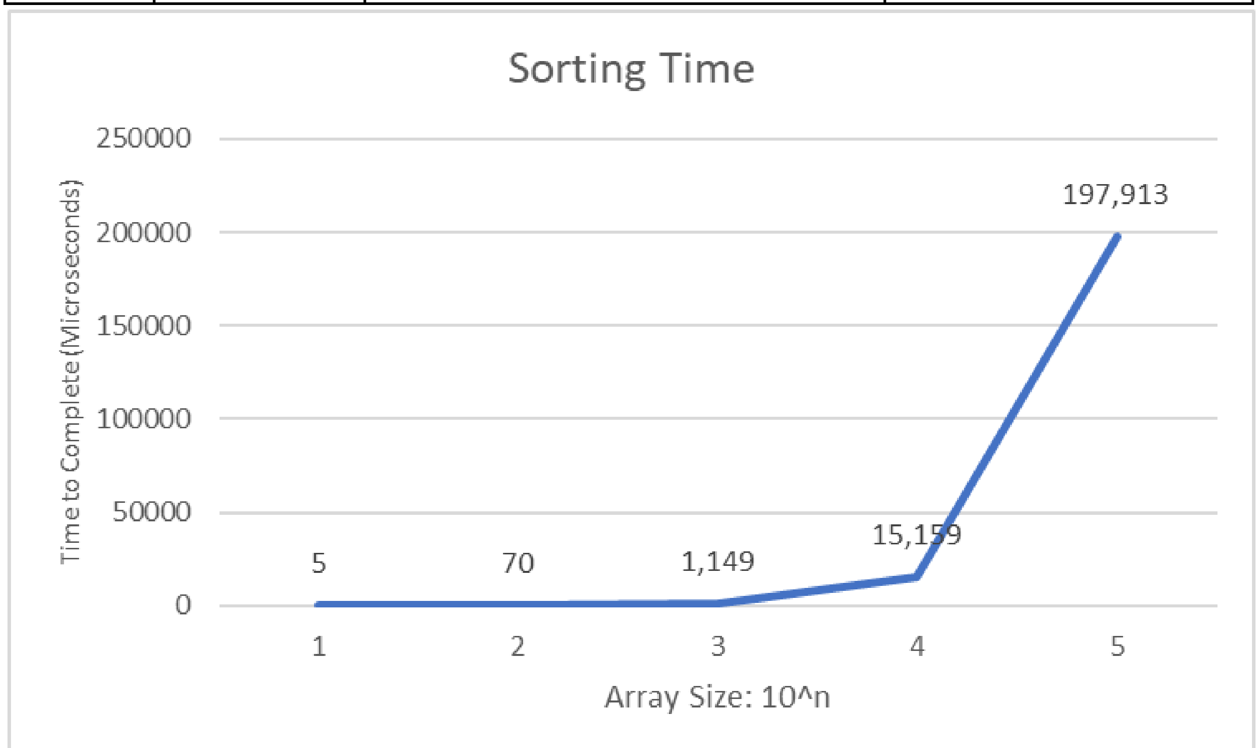
Shell Sort 1

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	5	-
3	1,000	71	14.2
4	10,000	1,156	16.282
5	100,000	15,131	13.089
6	1,000,000	197,913	13.079



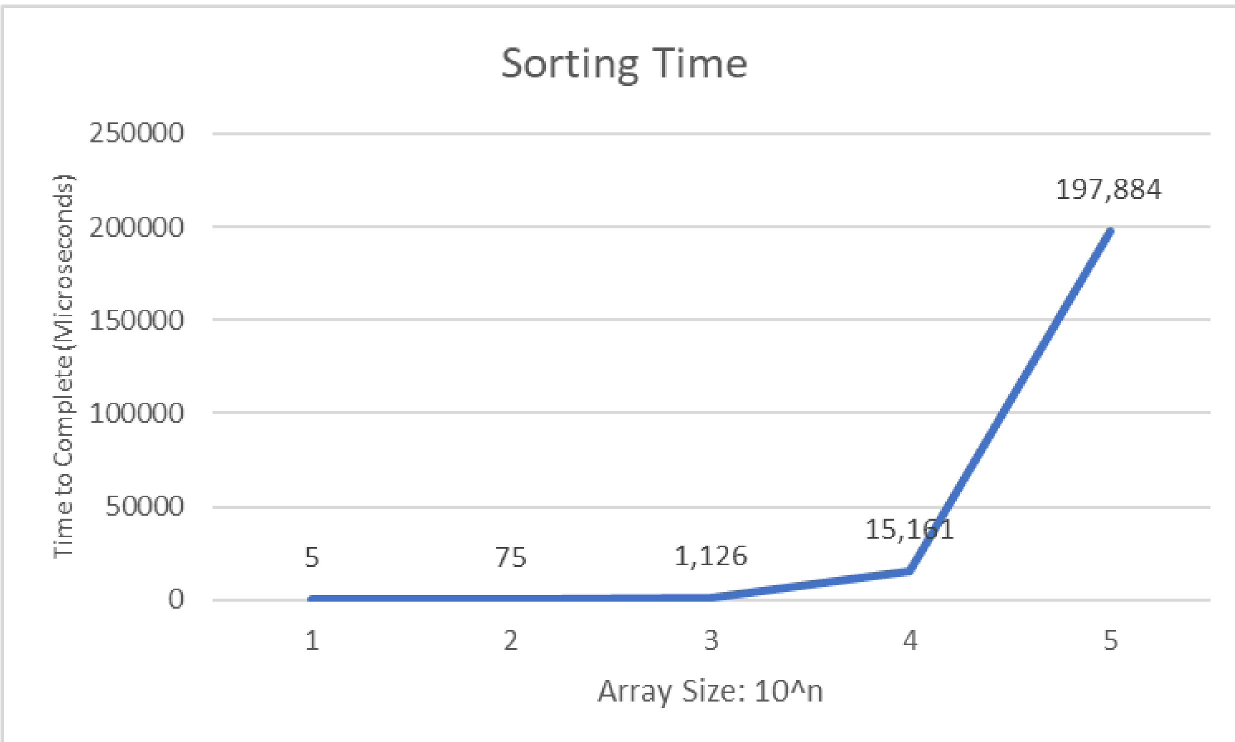
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	5	-
3	1,000	70	14
4	10,000	1,149	16.414
5	100,000	15,159	13.193
6	1,000,000	197,913	13.056



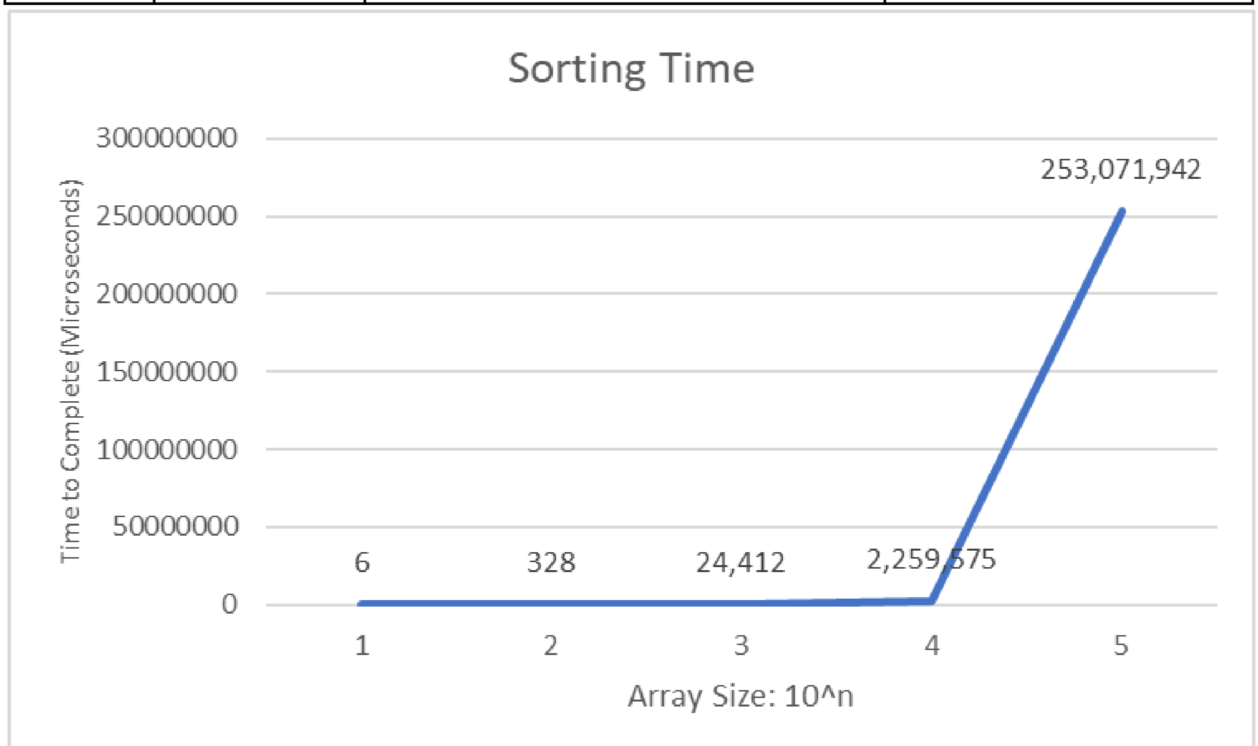
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	5	-
3	1,000	75	15
4	10,000	1,126	15.013
5	100,000	15,161	13.464
6	1,000,000	197,884	13.052



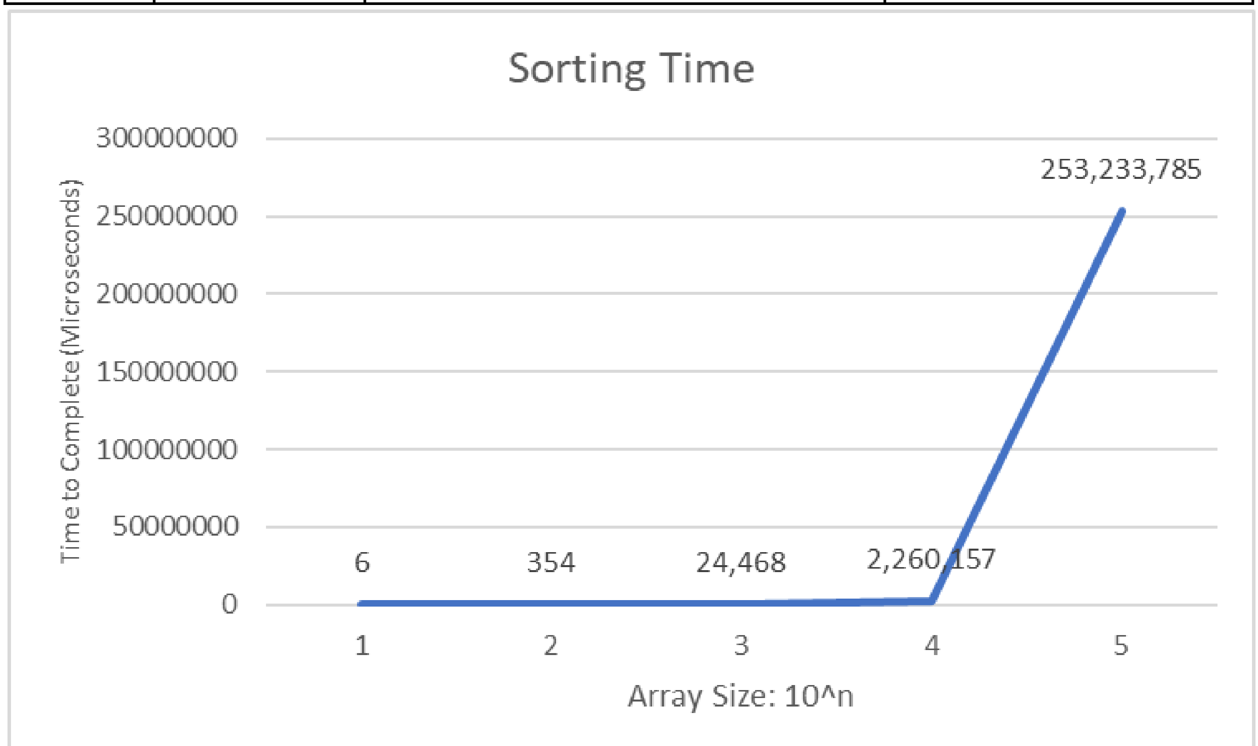
Insertion Sort 1

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	6	-
3	1,000	328	54.667
4	10,000	24,412	74.427
5	100,000	2,259,575	92.56
6	1,000,000	253,071,942	111.999



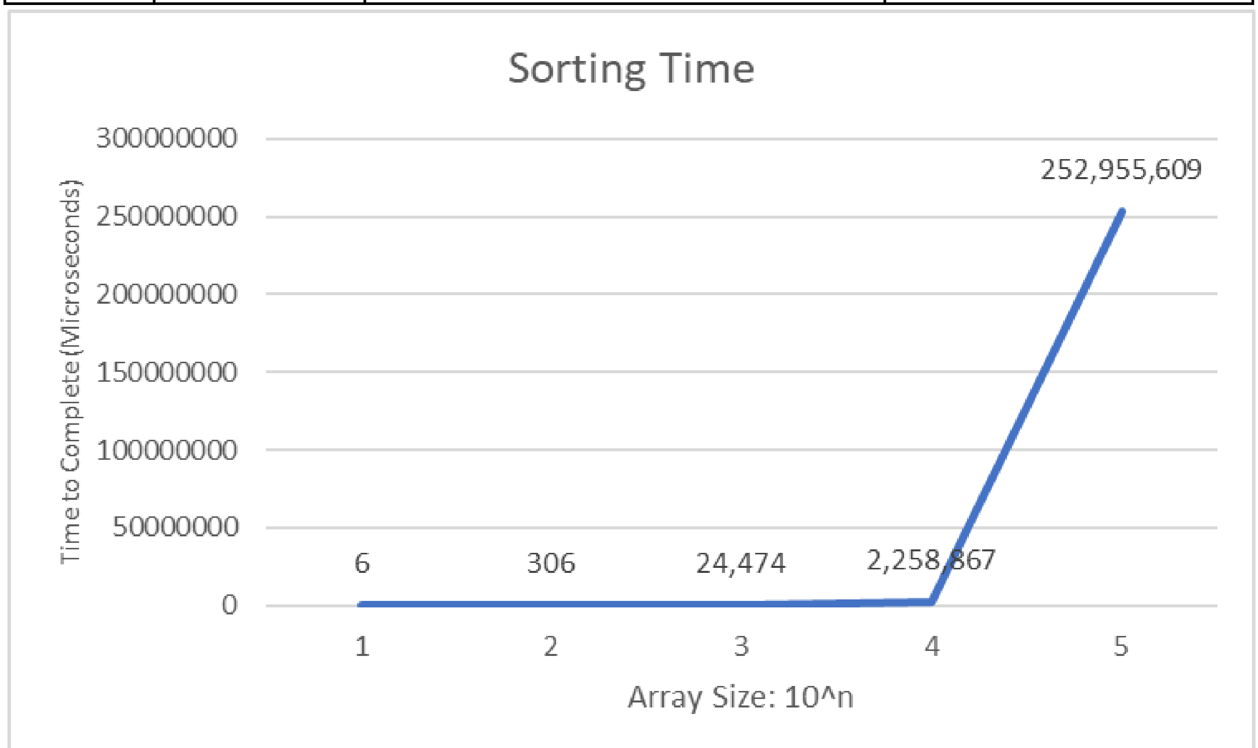
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	6	-
3	1,000	354	59
4	10,000	24,468	69.119
5	100,000	2,260,157	92.372
6	1,000,000	253,233,785	112.043



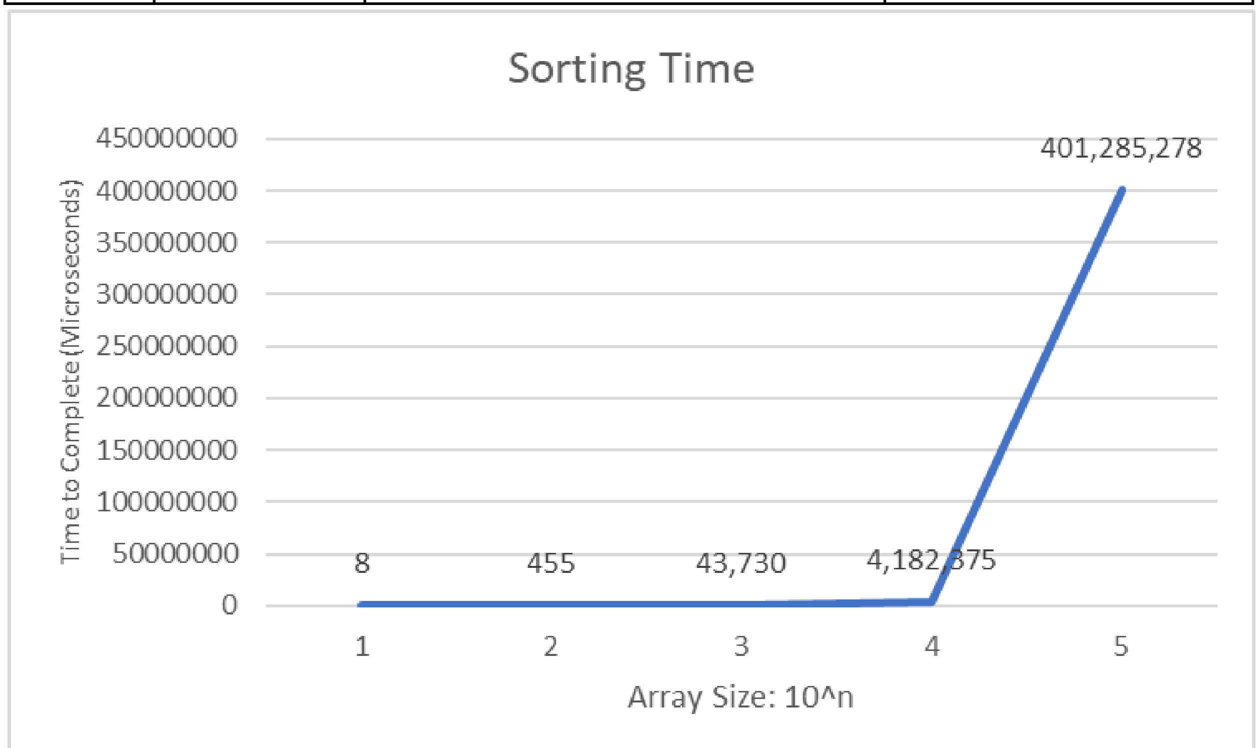
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	6	-
3	1,000	306	51
4	10,000	24,474	79.980
5	100,000	2,258,867	92.297
6	1,000,000	252,955,609	111.983



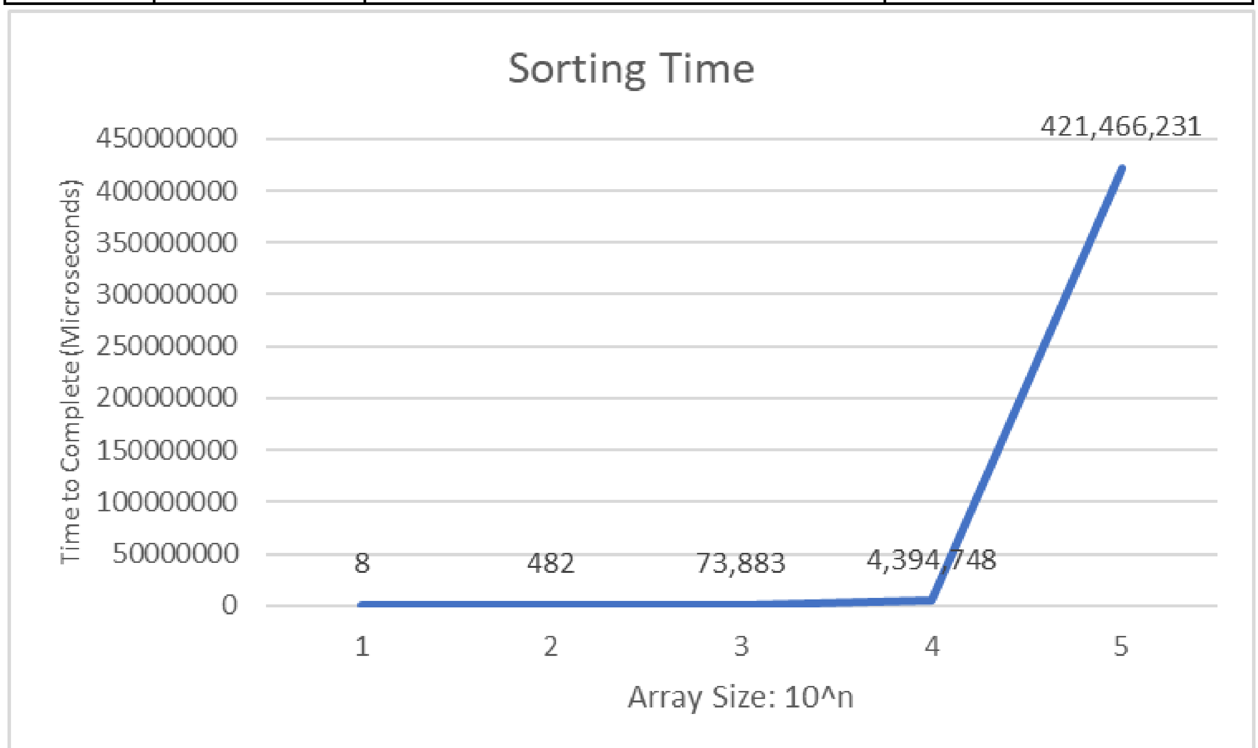
Selection Sort 1

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	8	-
3	1,000	455	56.875
4	10,000	43,730	96.11
5	100,000	4,182,375	95.641
6	1,000,000	401,285,278	95.947



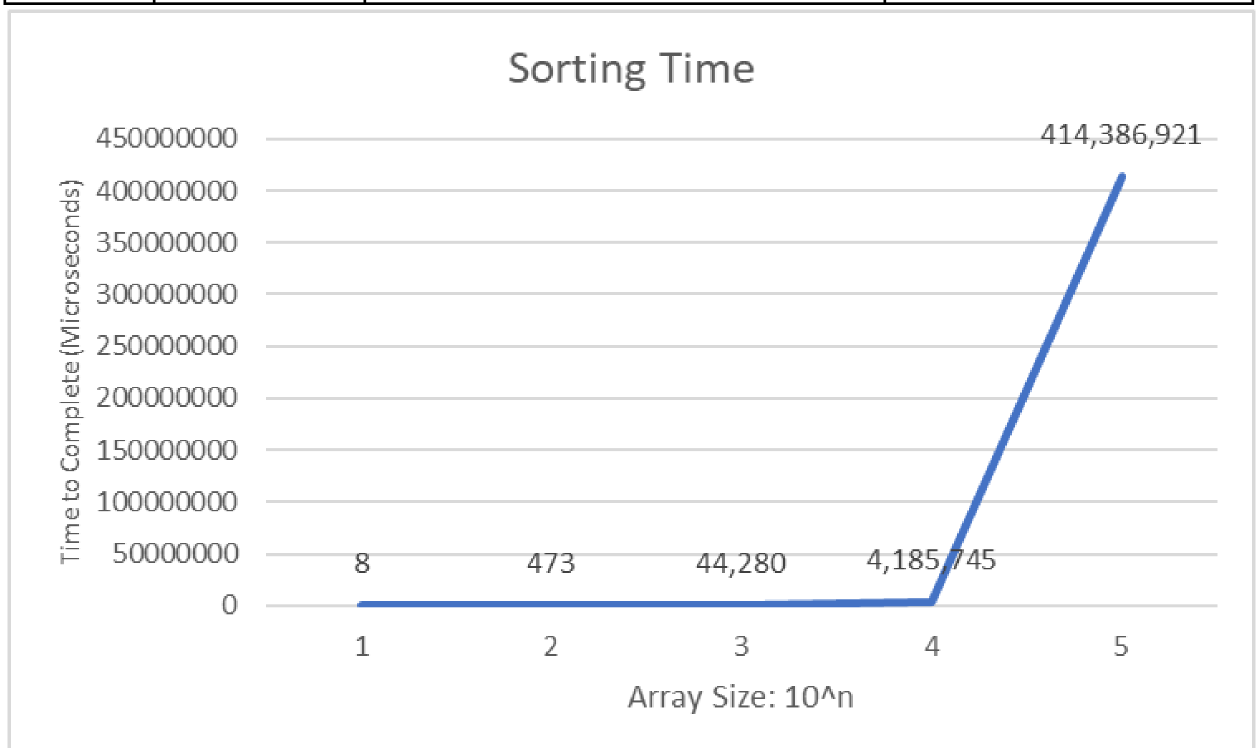
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	8	-
3	1,000	482	60.25
4	10,000	73,883	153.284
5	100,000	4,394,748	59.483
6	1,000,000	421,466,231	95.902



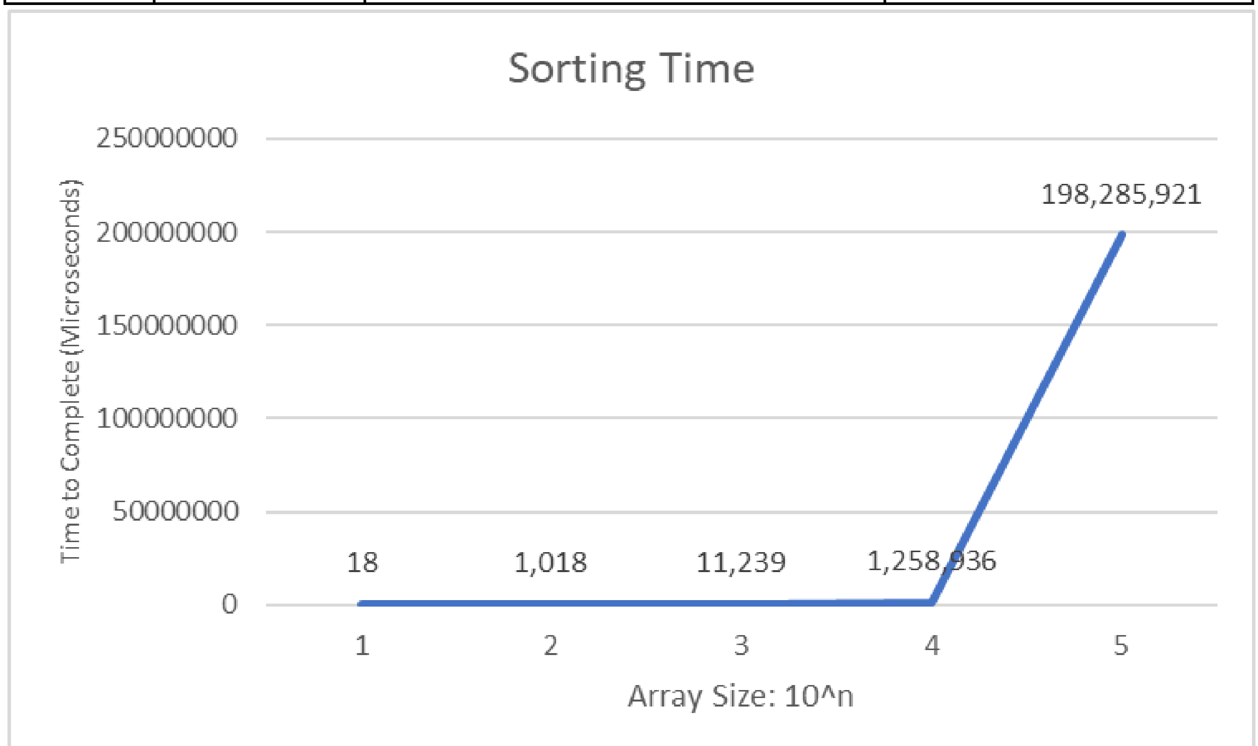
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	8	-
3	1,000	473	59.125
4	10,000	44,280	93.615
5	100,000	4,185,745	94.529
6	1,000,000	414,386,921	98.999



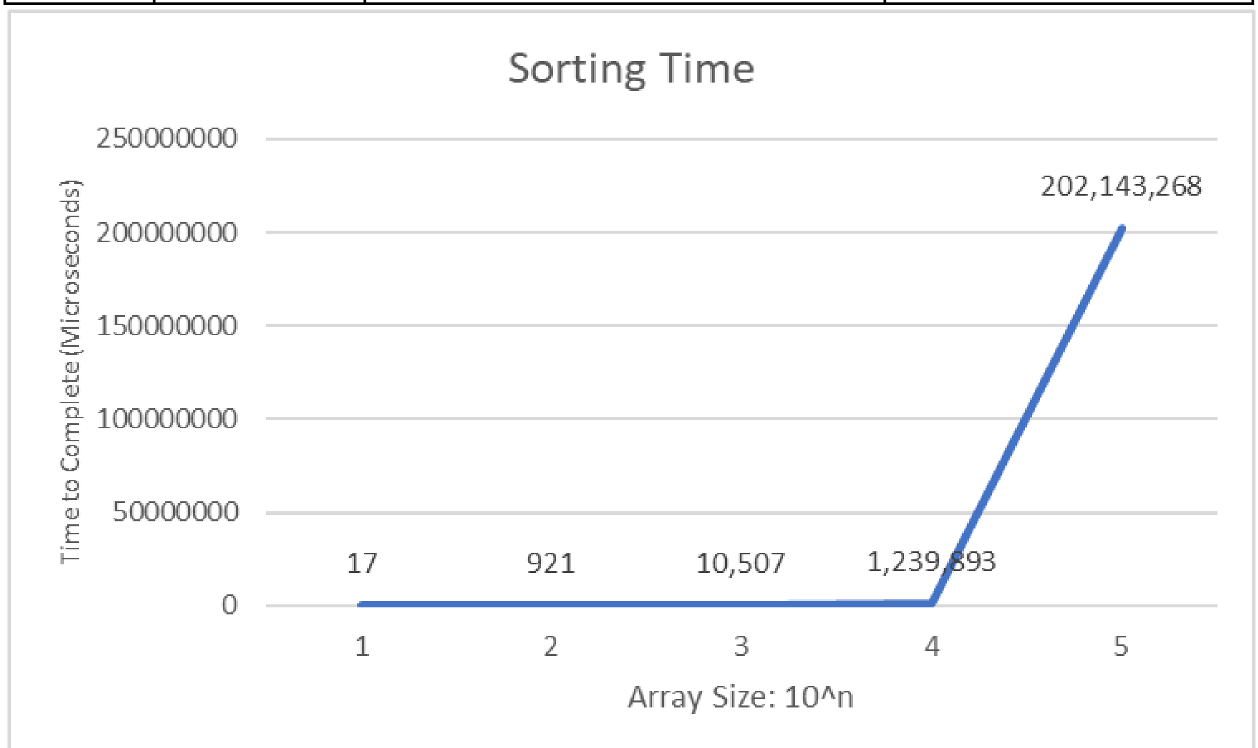
Bubble Sort 1

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	18	-
3	1,000	1,018	56.556
4	10,000	11,239	11.040
5	100,000	1,258,936	112.015
6	1,000,000	198,285,921	157.503



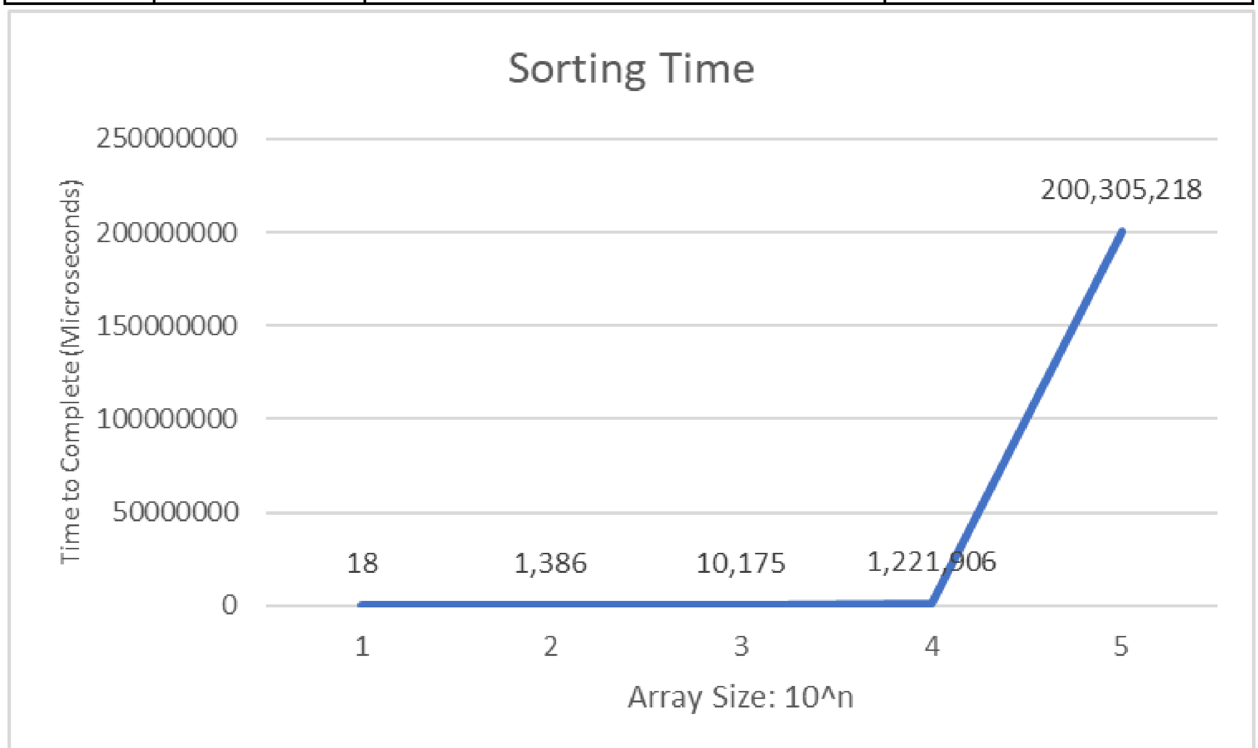
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	17	-
3	1,000	921	54.176
4	10,000	10,507	11.408
5	100,000	1,239,893	118.006
6	1,000,000	202,143,268	163.033



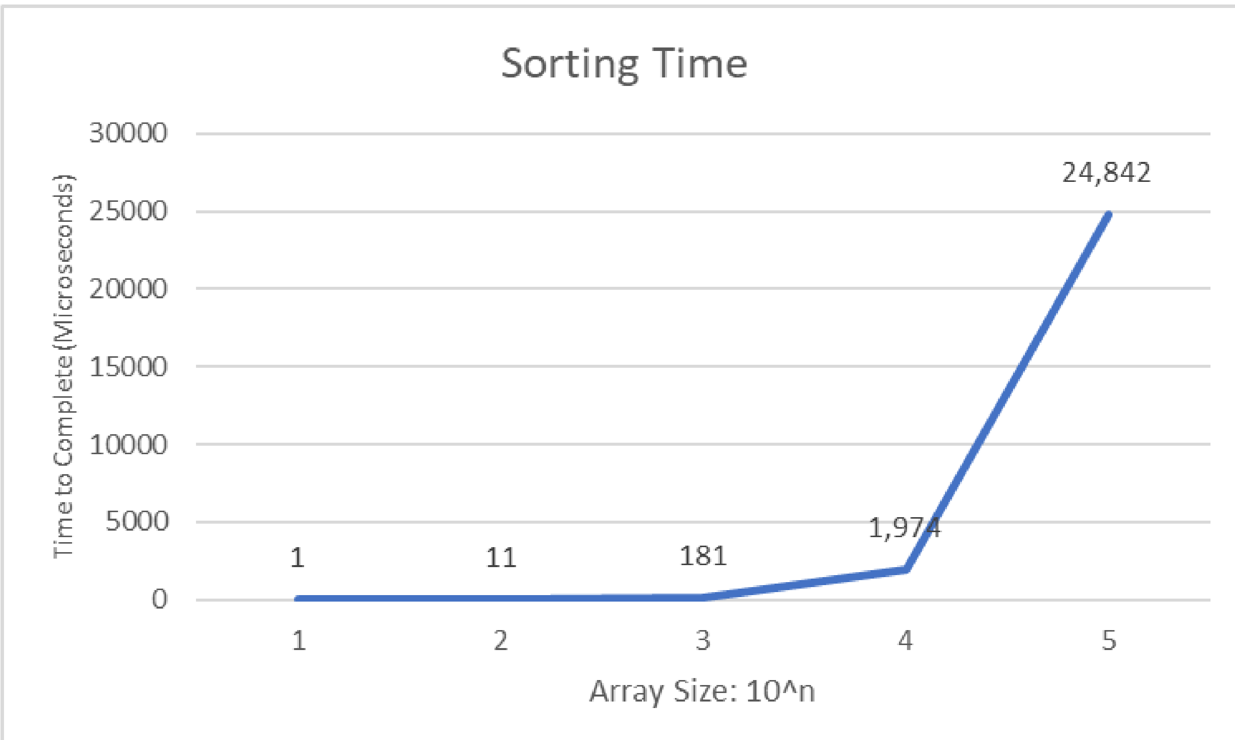
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	18	-
3	1,000	1,386	77
4	10,000	10,175	7.341
5	100,000	1,221,906	120.089
6	1,000,000	200,305,218	163.929



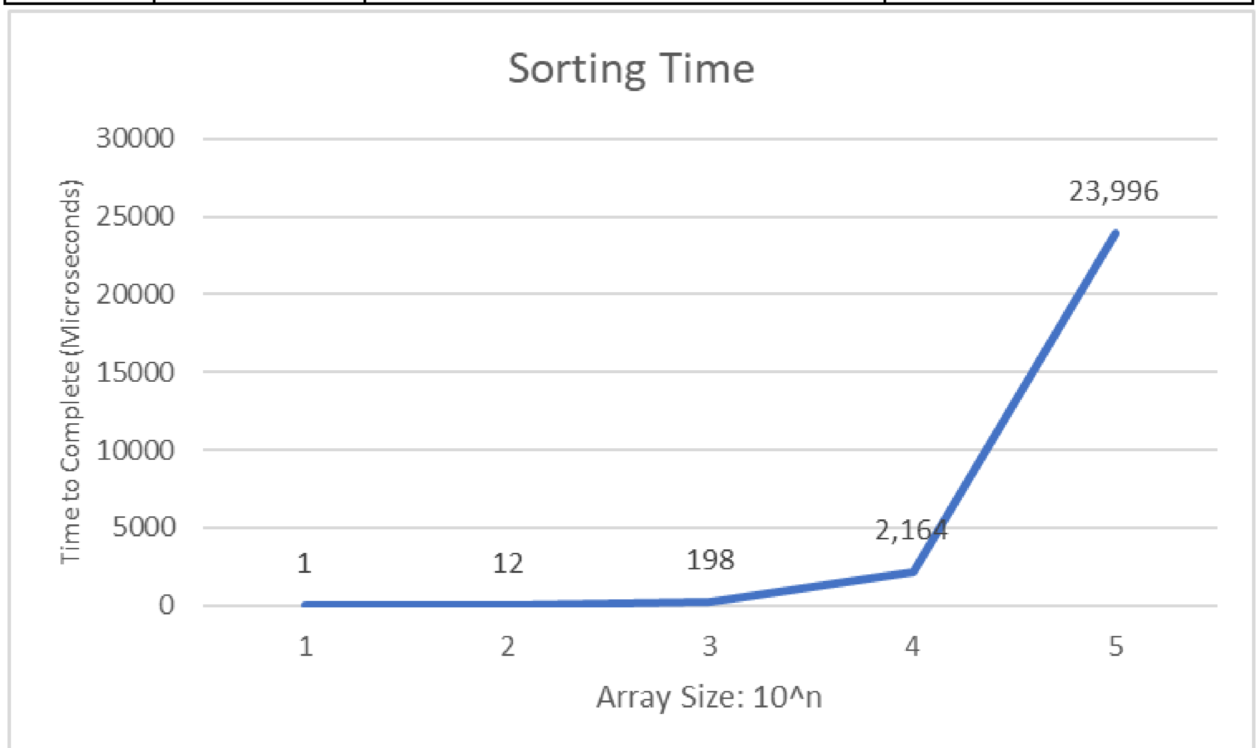
Increasing Array
Shell Sort 1

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	1	-
3	1,000	11	11
4	10,000	181	16.45
5	100,000	1,974	10.906
6	1,000,000	24,842	12.585



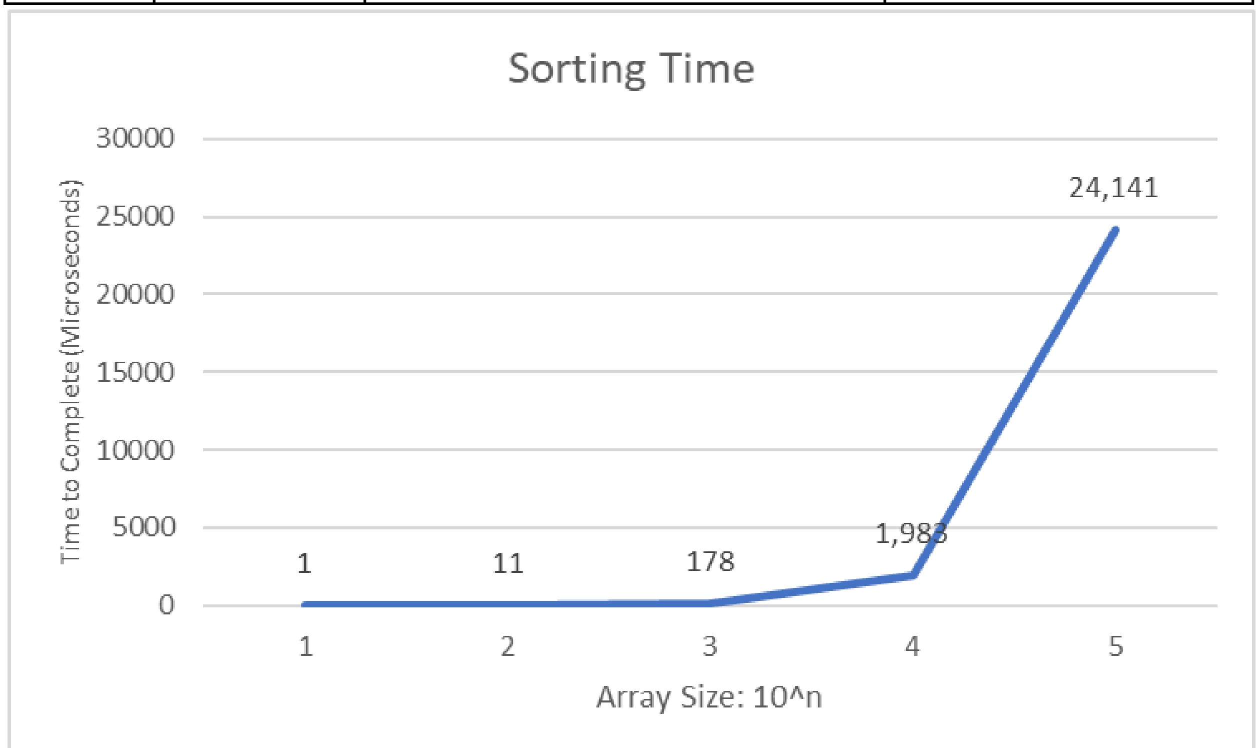
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	1	-
3	1,000	12	12
4	10,000	198	16.5
5	100,000	2,164	10.929
6	1,000,000	23,996	11.089



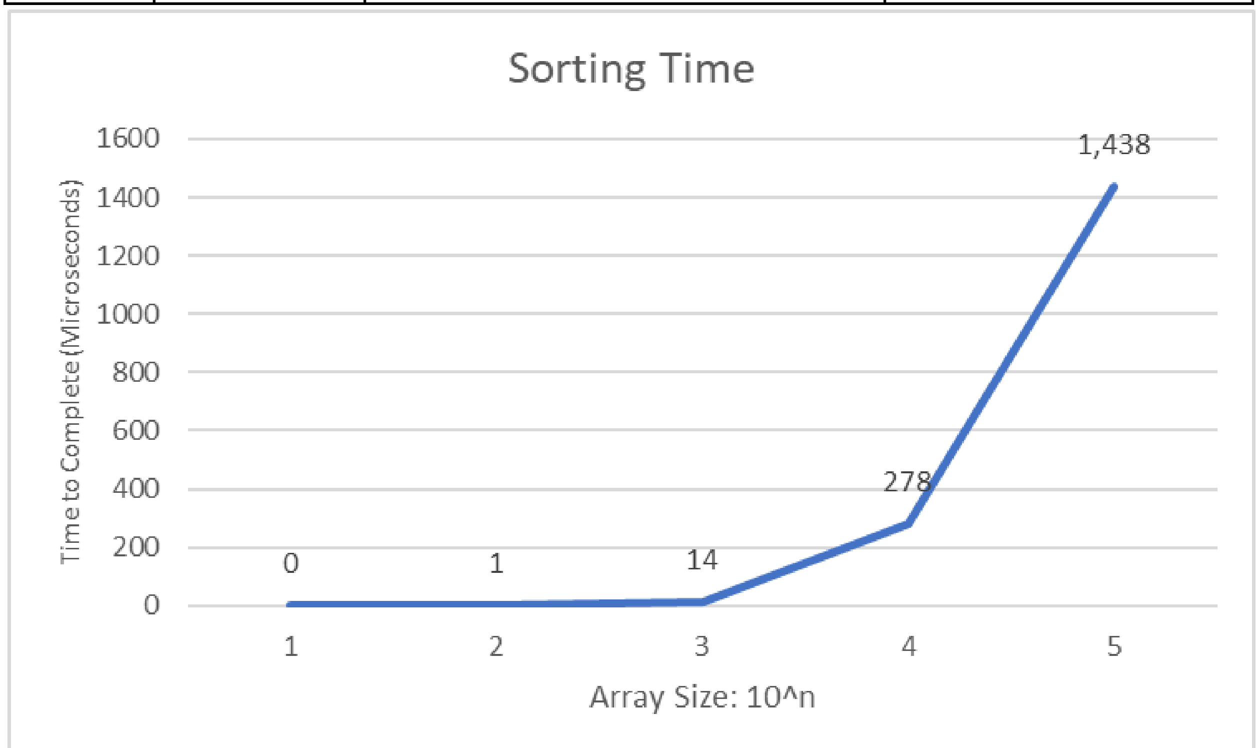
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	1	-
3	1,000	11	11
4	10,000	178	16.182
5	100,000	1,983	11.140
6	1,000,000	24,141	12.174



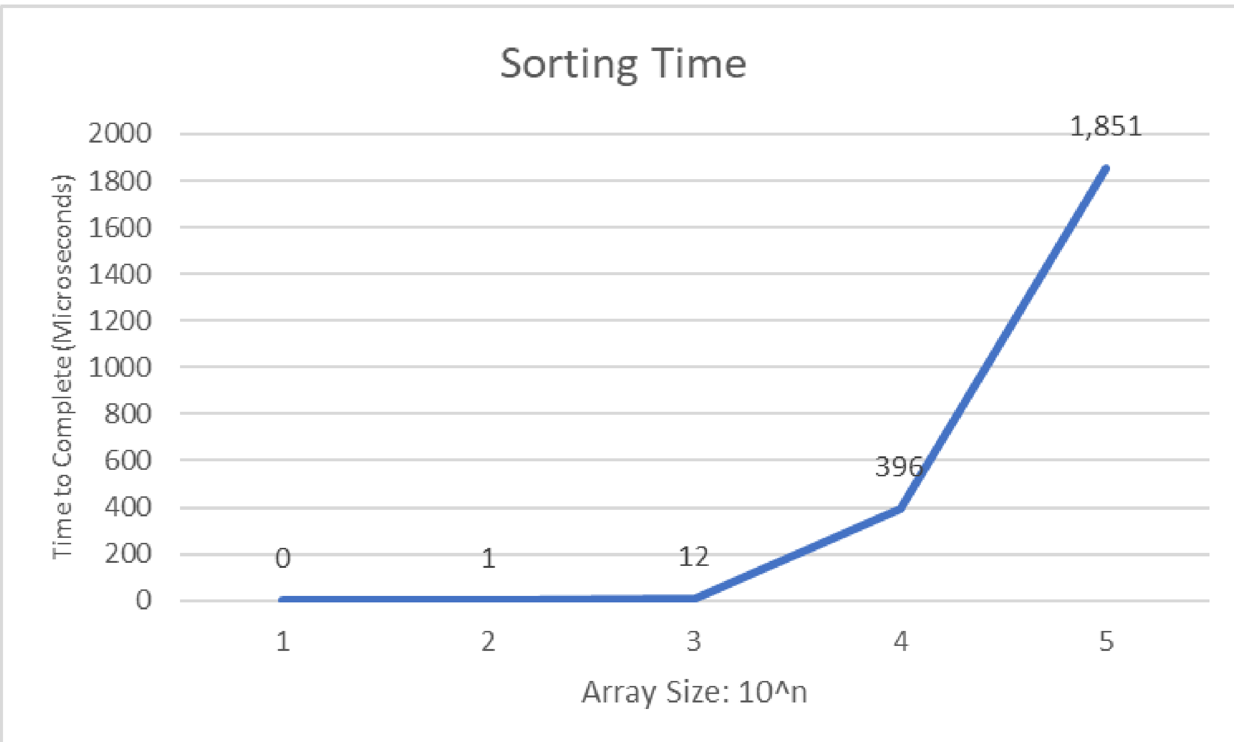
Insertion Sort 1

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	14	14
5	100,000	278	19.857
6	1,000,000	1,438	5.173



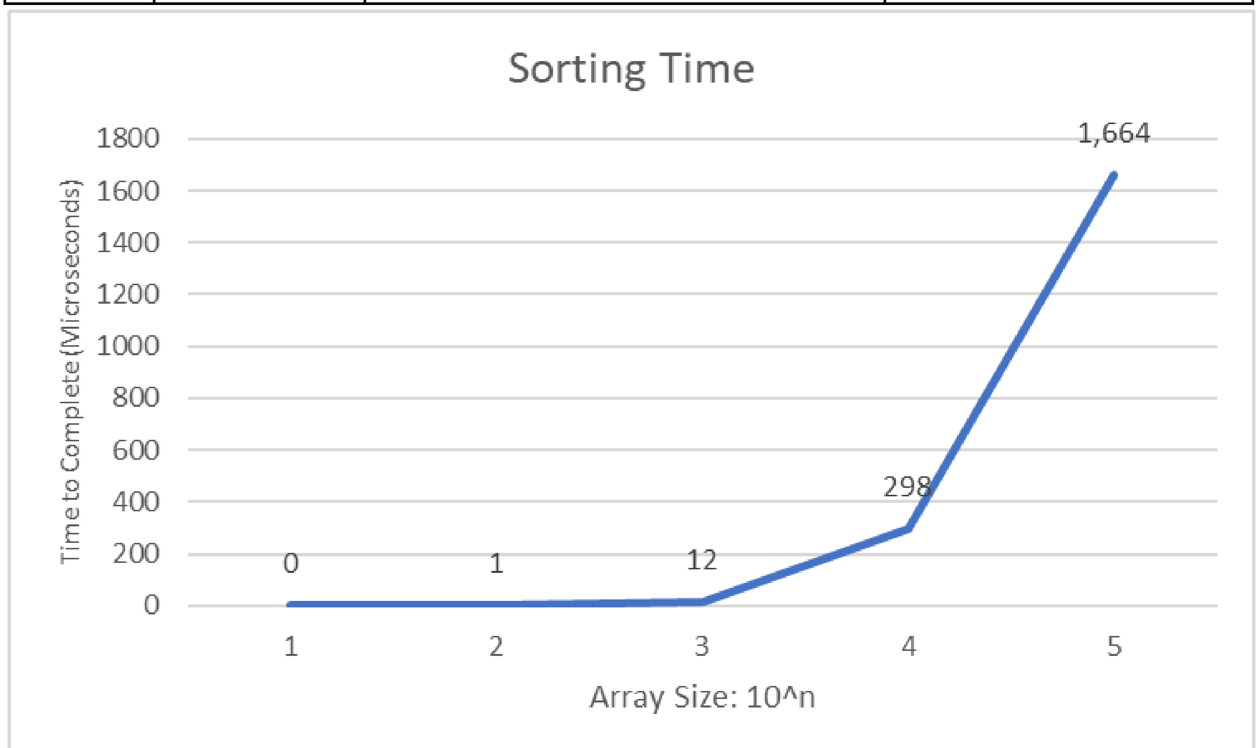
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	12	12
5	100,000	396	33
6	1,000,000	1,851	4.598



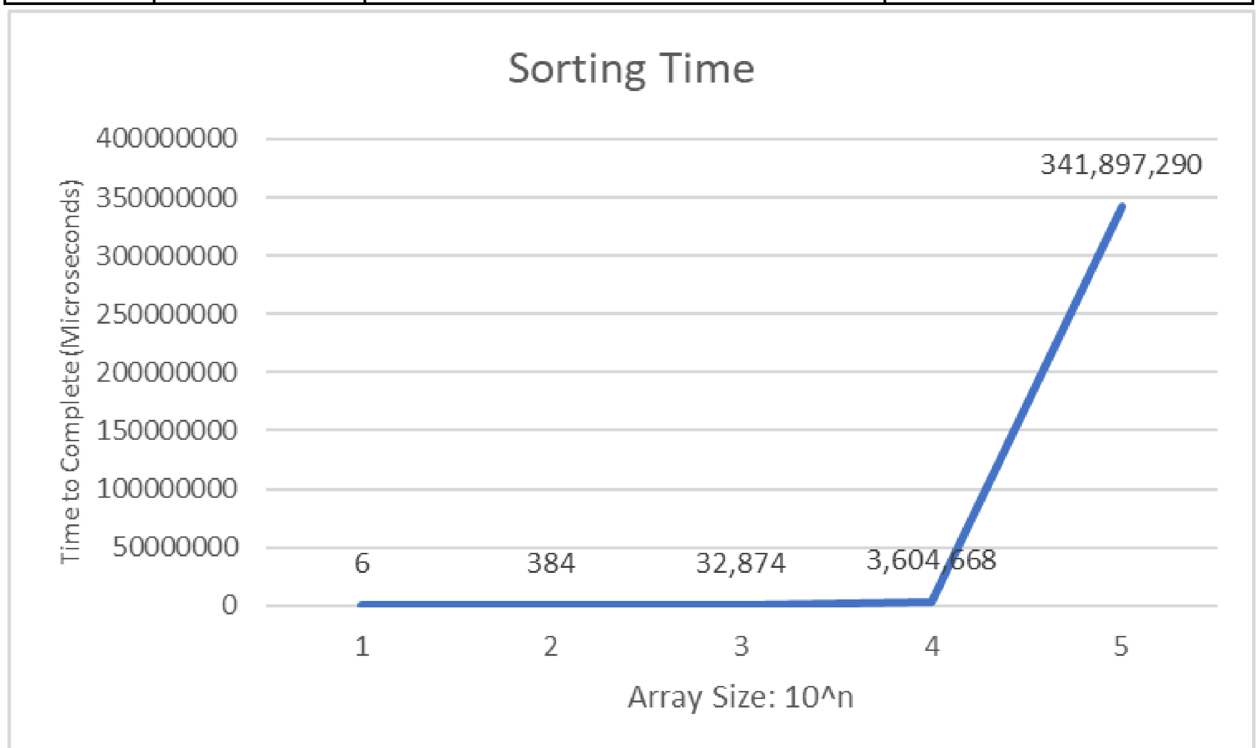
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	12	12
5	100,000	298	24.833
6	1,000,000	1,664	5.584



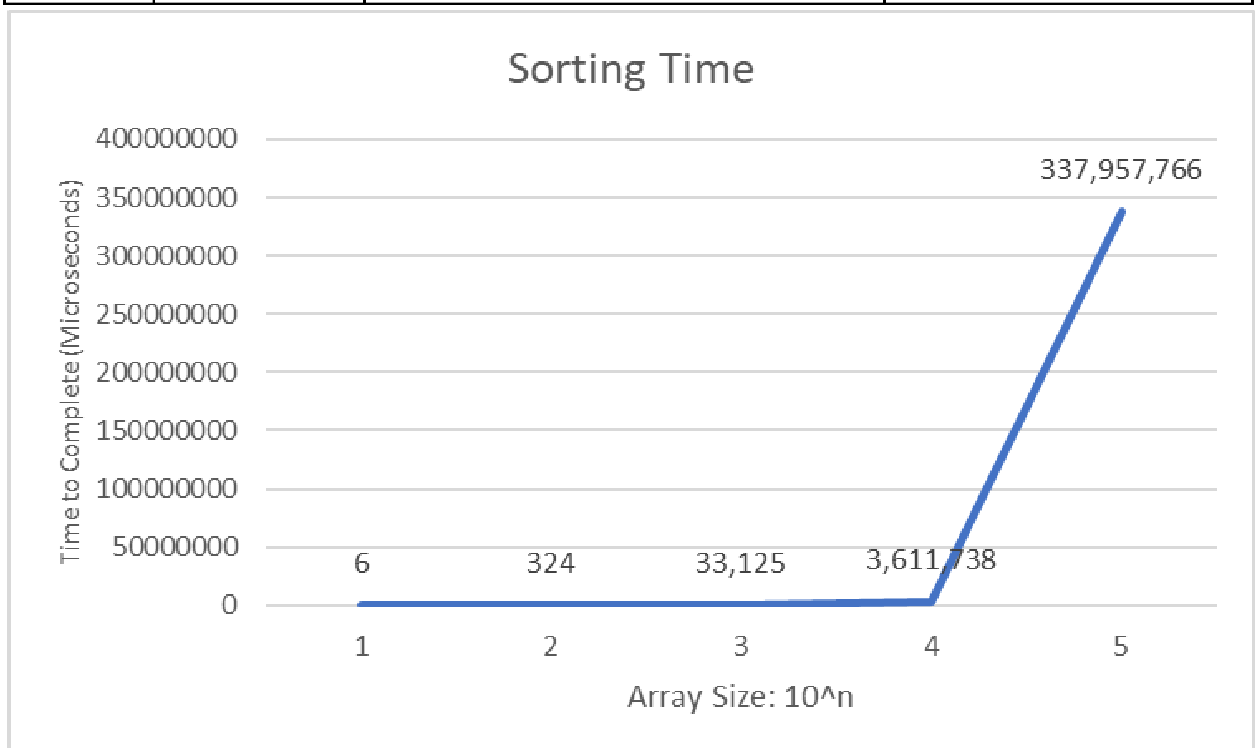
Selection Sort 1

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	6	-
3	1,000	384	64
4	10,000	32,874	85.609
5	100,000	3,604,668	1,022.226
6	1,000,000	341,897,290	10.174



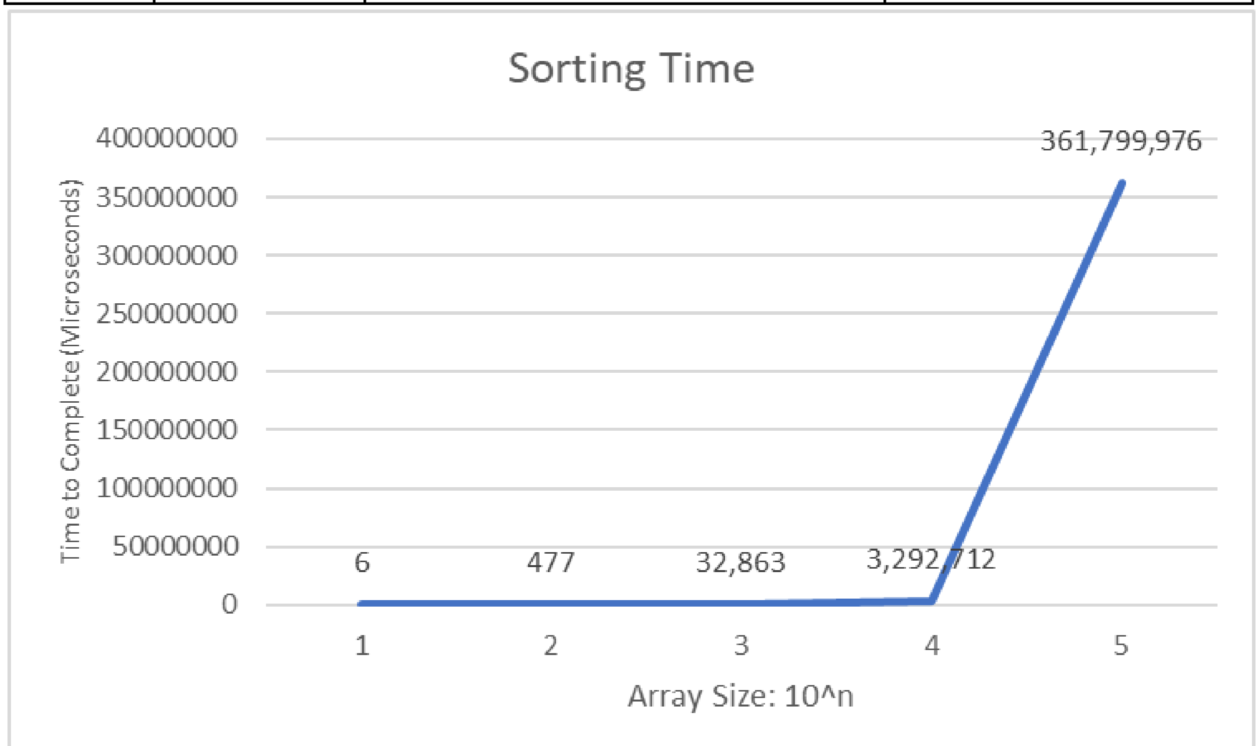
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	6	-
3	1,000	324	54
4	10,000	33,125	102.238
5	100,000	3,611,738	109.034
6	1,000,000	337,957,766	93.572



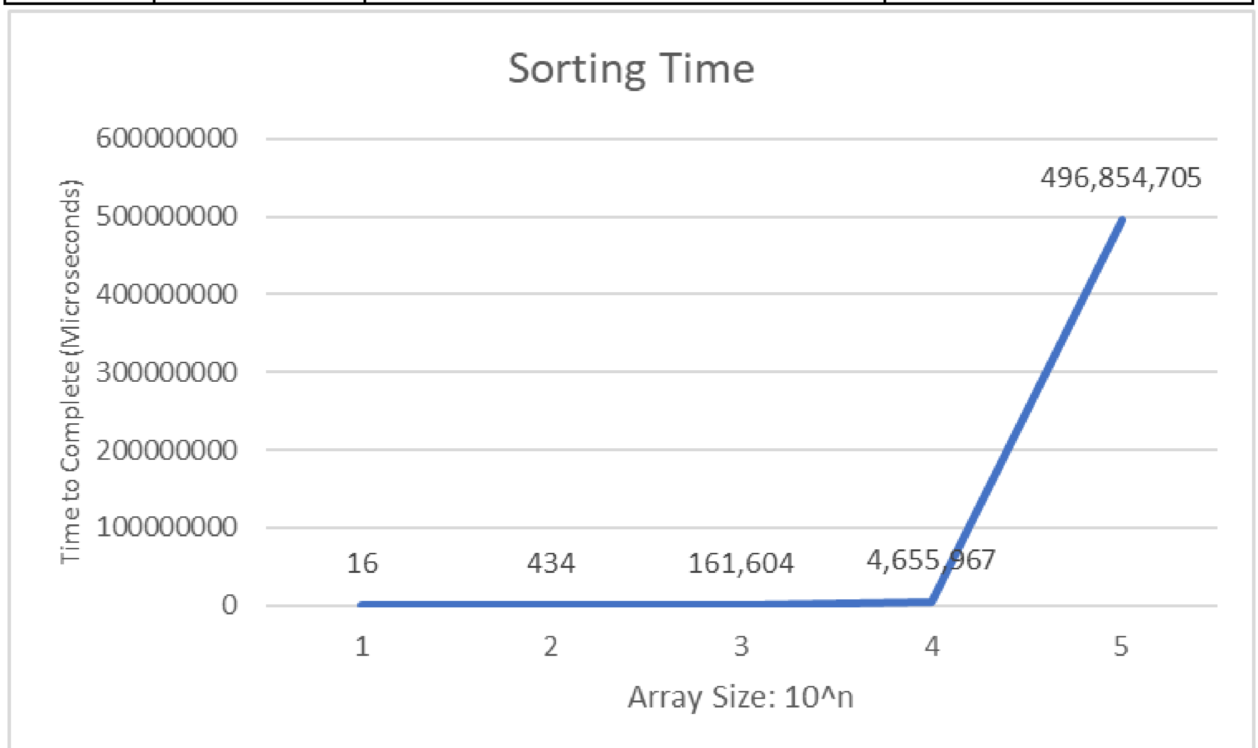
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	6	-
3	1,000	477	79.5
4	10,000	32,863	68.895
5	100,000	3,292,712	100.195
6	1,000,000	361,799,976	109.879



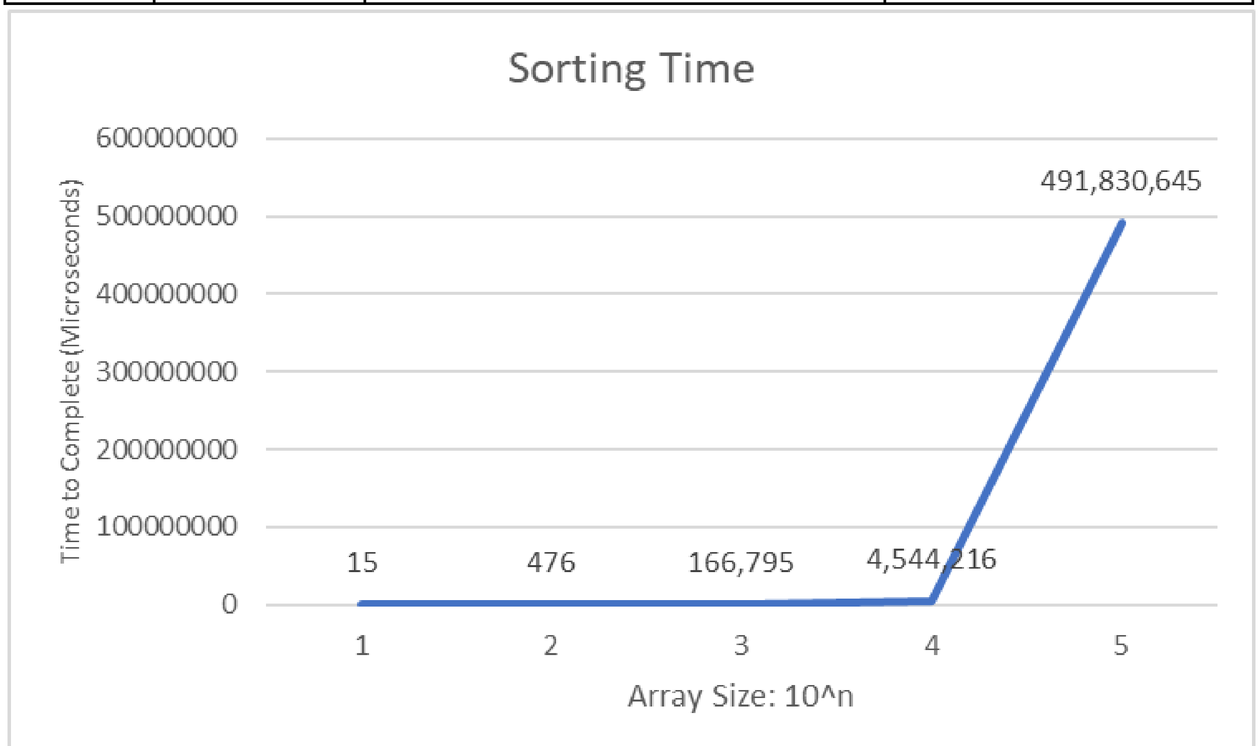
Bubble Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	16	-
3	1,000	434	27.125
4	10,000	161,604	372.359
5	100,000	4,655,967	28.811
6	1,000,000	496,854,705	106.714



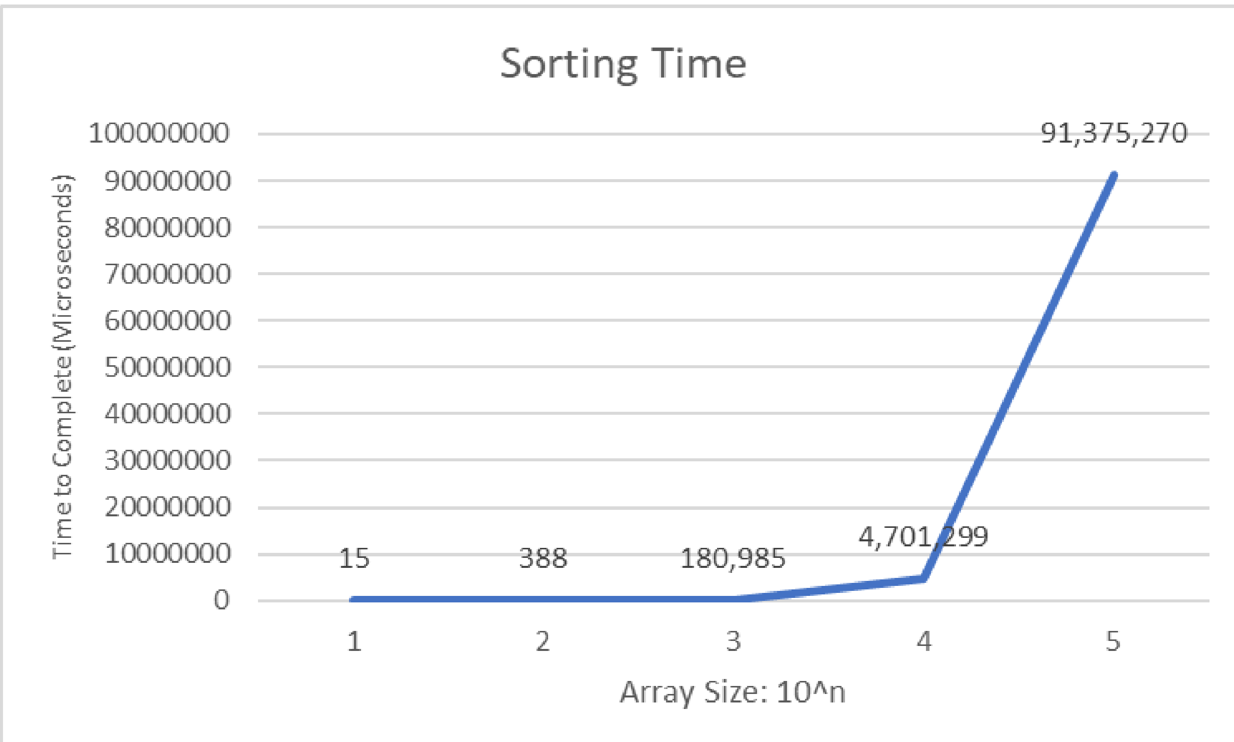
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	15	-
3	1,000	476	31.733
4	10,000	166,795	350.410
5	100,000	4,544,216	27.244
6	1,000,000	491,830,645	108.232



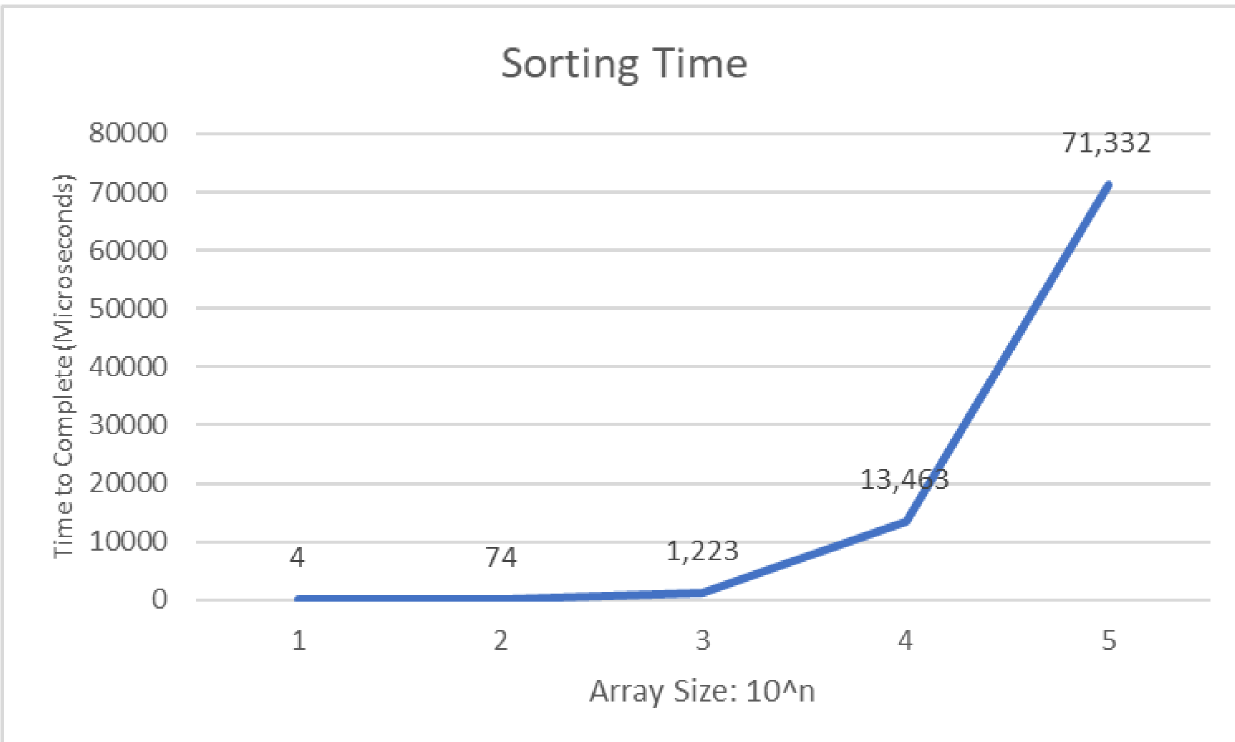
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	15	-
3	1,000	388	25.867
4	10,000	180,985	466.456
5	100,000	4,701,299	25.976
6	1,000,000	91,375,270	19.436



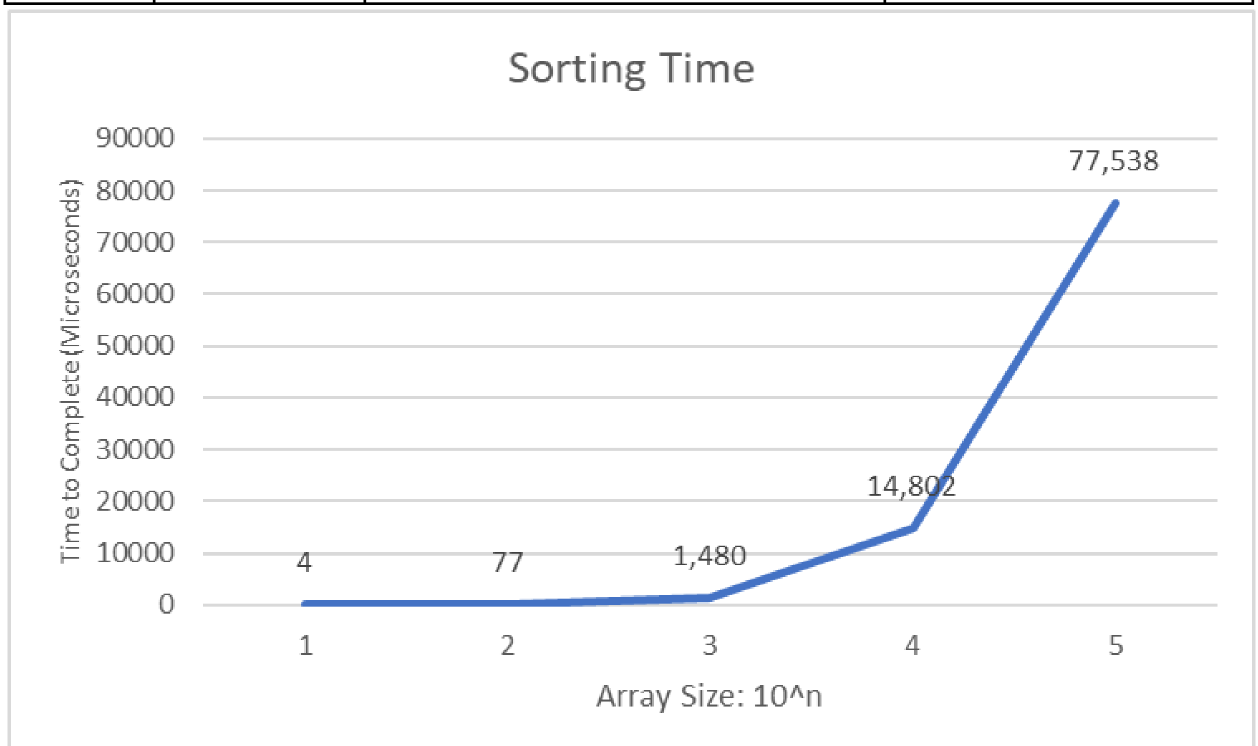
Decreasing Array
Shell Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	74	18.5
4	10,000	1,223	16.527
5	100,000	13,463	11.008
6	1,000,000	71,332	5.298



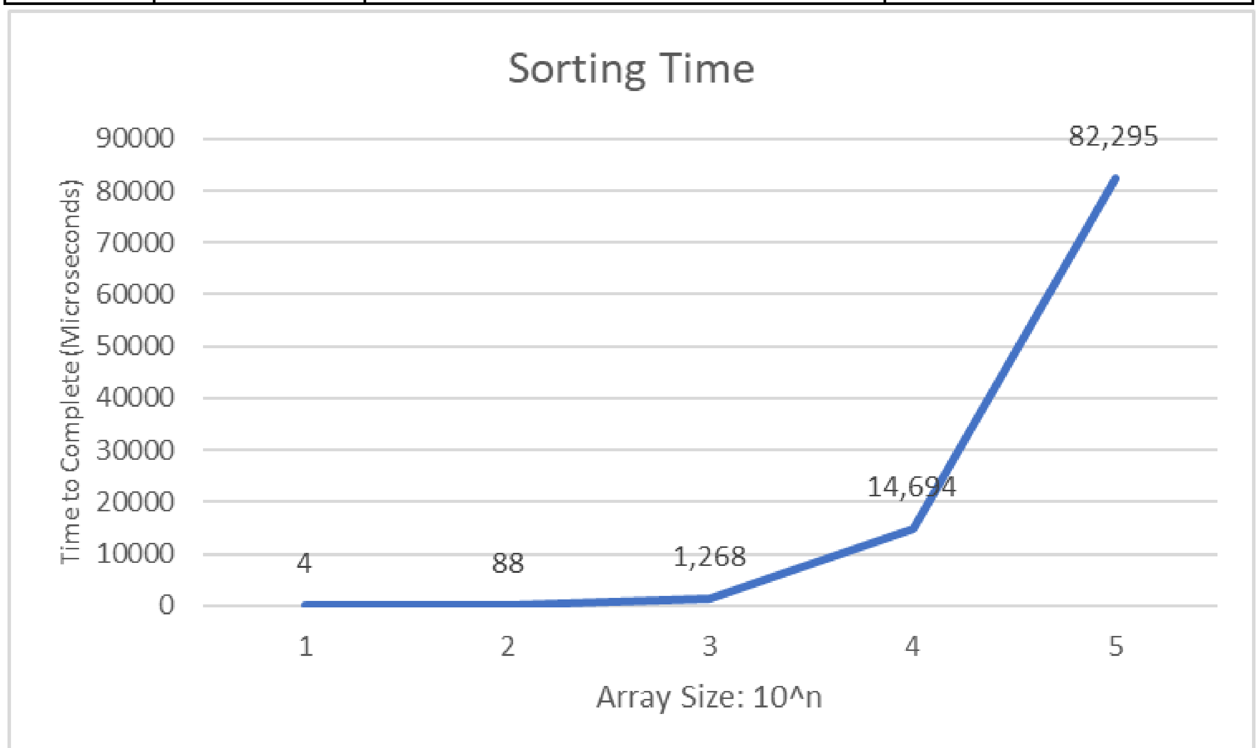
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	77	19.250
4	10,000	1,480	19.221
5	100,000	14,802	10.001
6	1,000,000	77,538	5.238



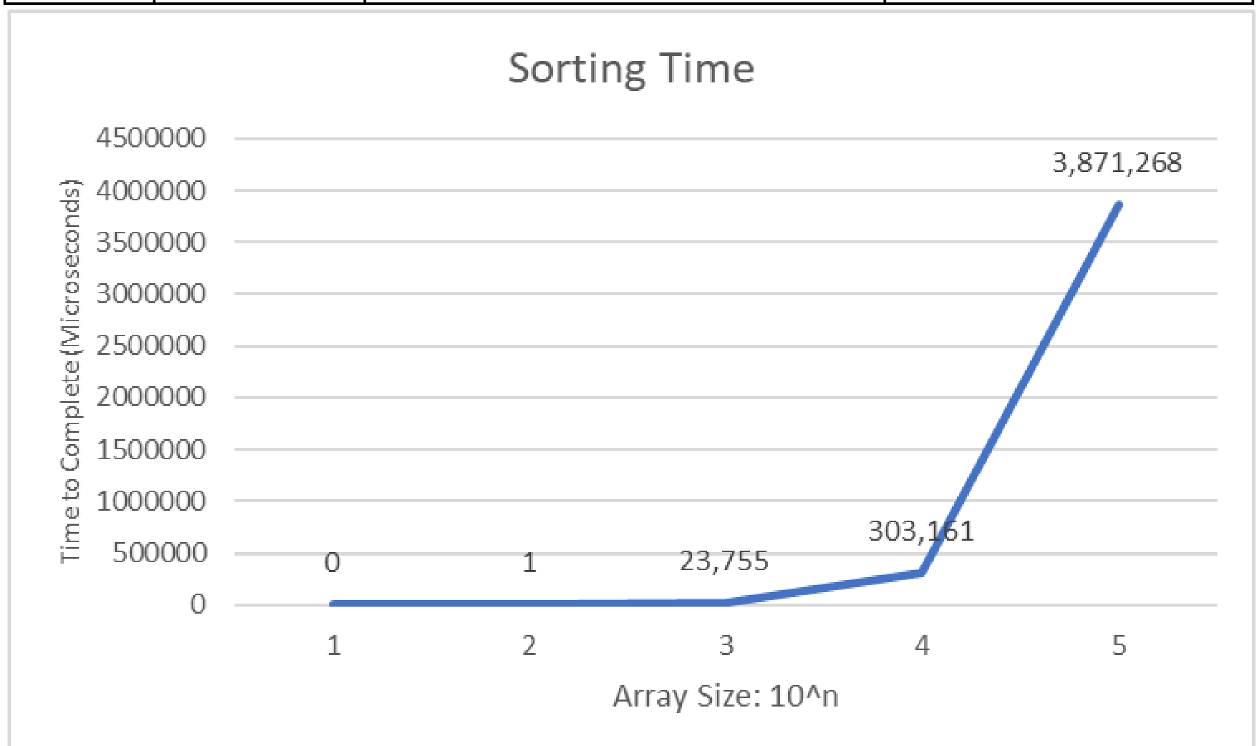
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	4	-
3	1,000	88	22
4	10,000	1,268	14.409
5	100,000	14,694	11.588
6	1,000,000	82,295	5.601



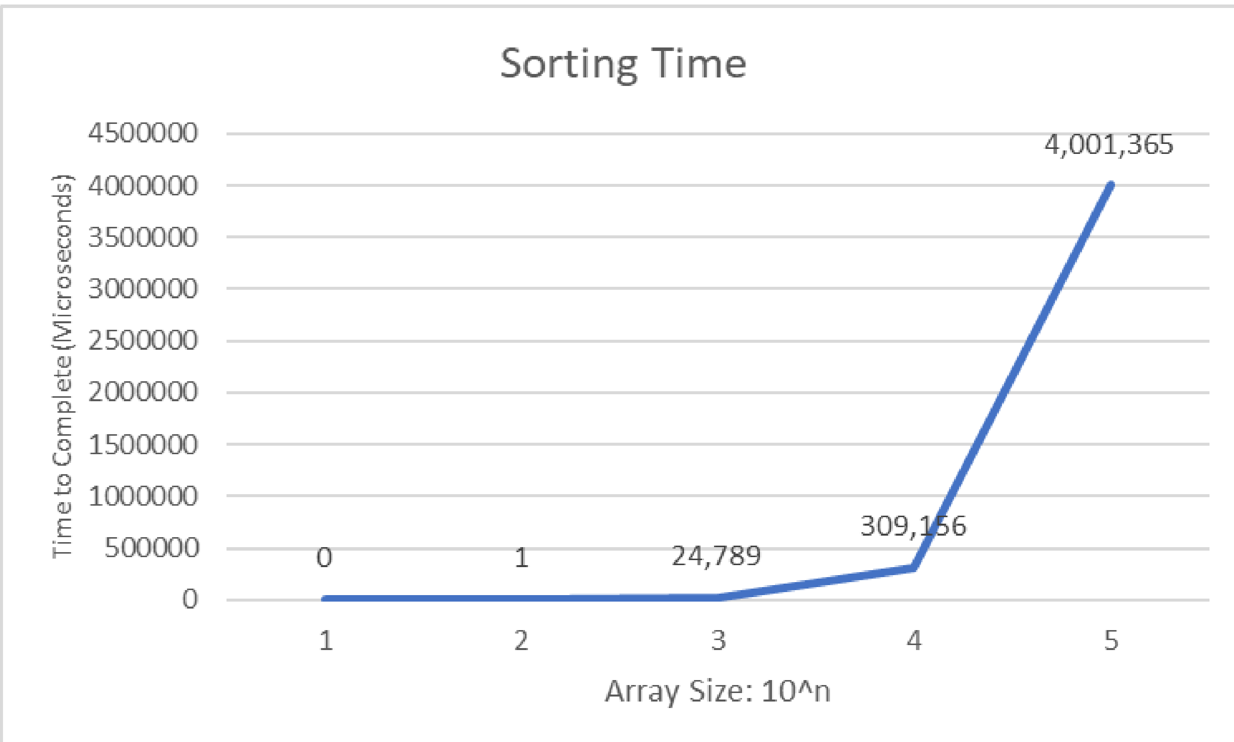
Insertion Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	23,755	23,755
5	100,000	303,161	12.762
6	1,000,000	3,871,268	12.770



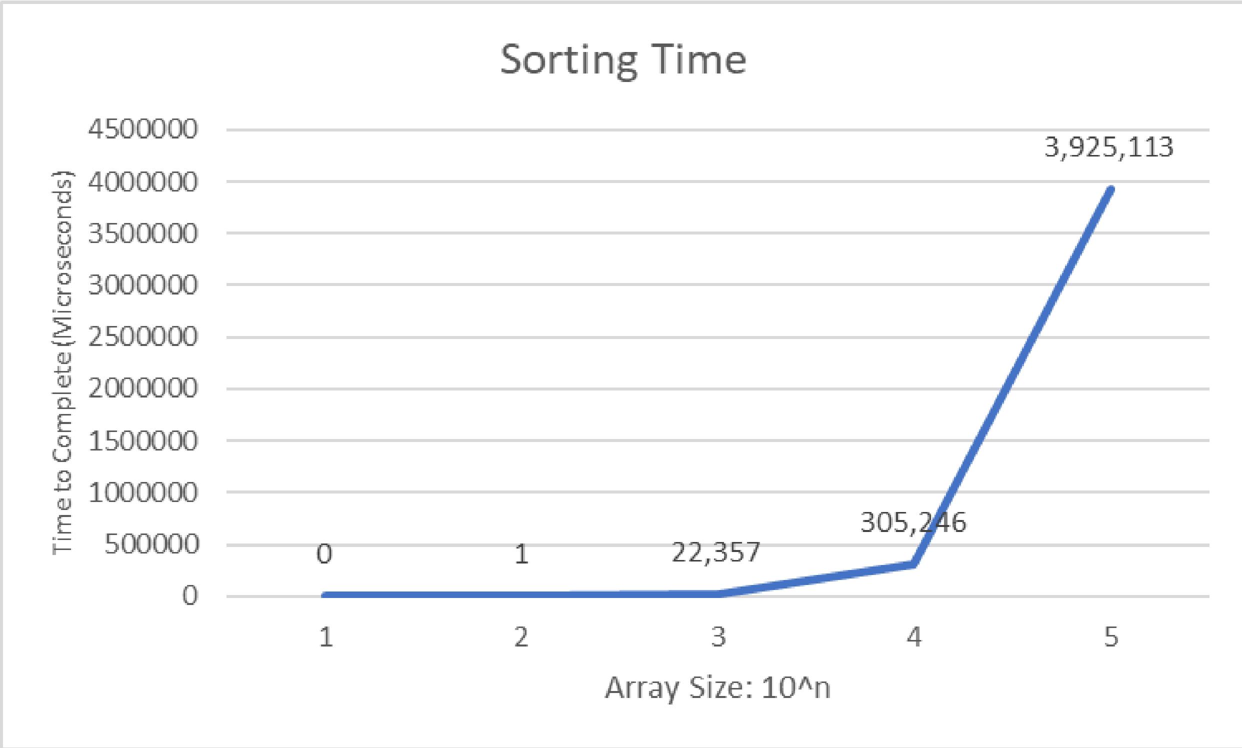
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	24,789	24,789
5	100,000	309,156	12.471
6	1,000,000	4,001,365	12.943



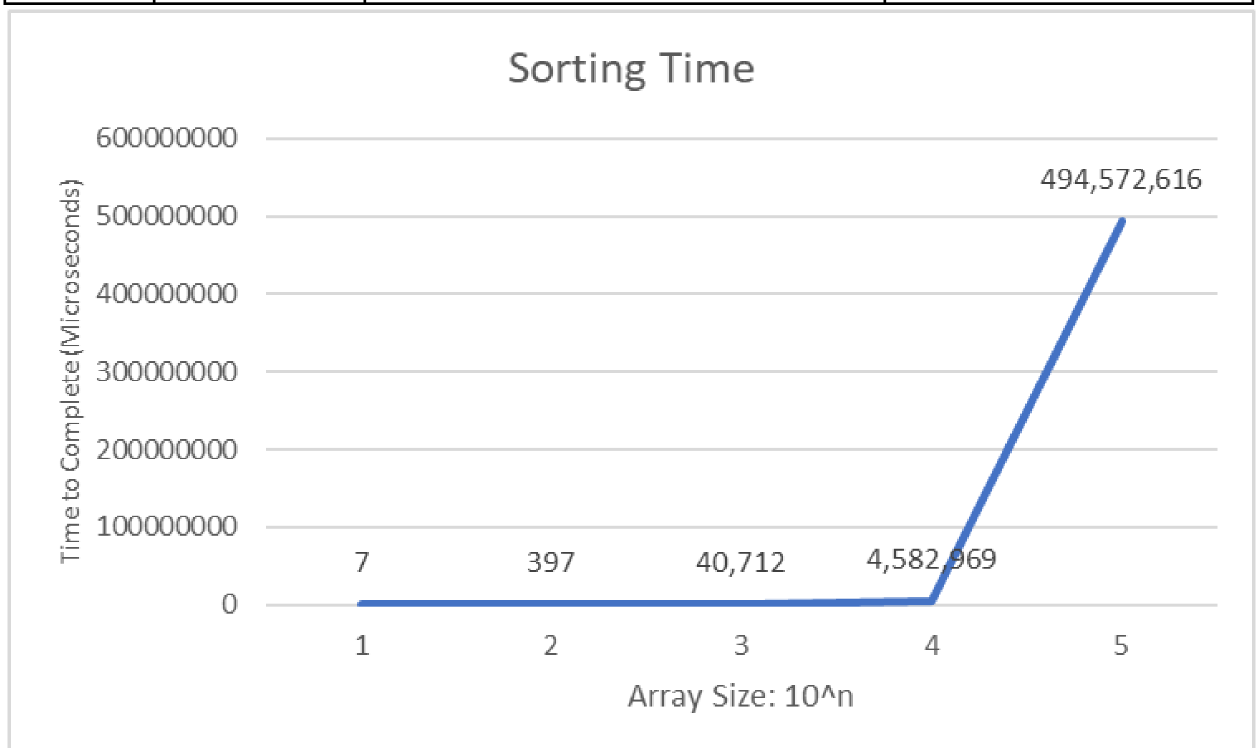
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	0	-
3	1,000	1	1
4	10,000	22,357	22,357
5	100,000	305,246	13.653
6	1,000,000	3,925,113	12.859



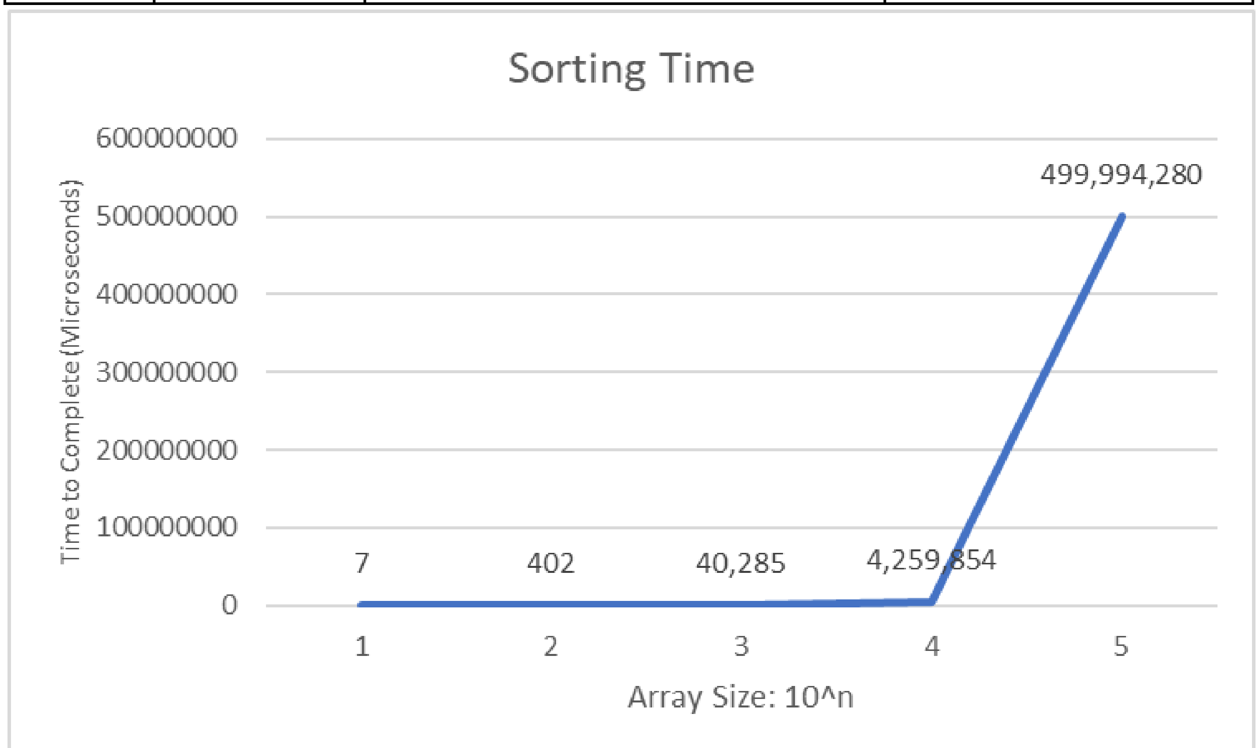
Selection Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	7	-
3	1,000	397	56.714
4	10,000	40,712	102.549
5	100,000	4,582,969	112.570
6	1,000,000	494,572,616	107.915



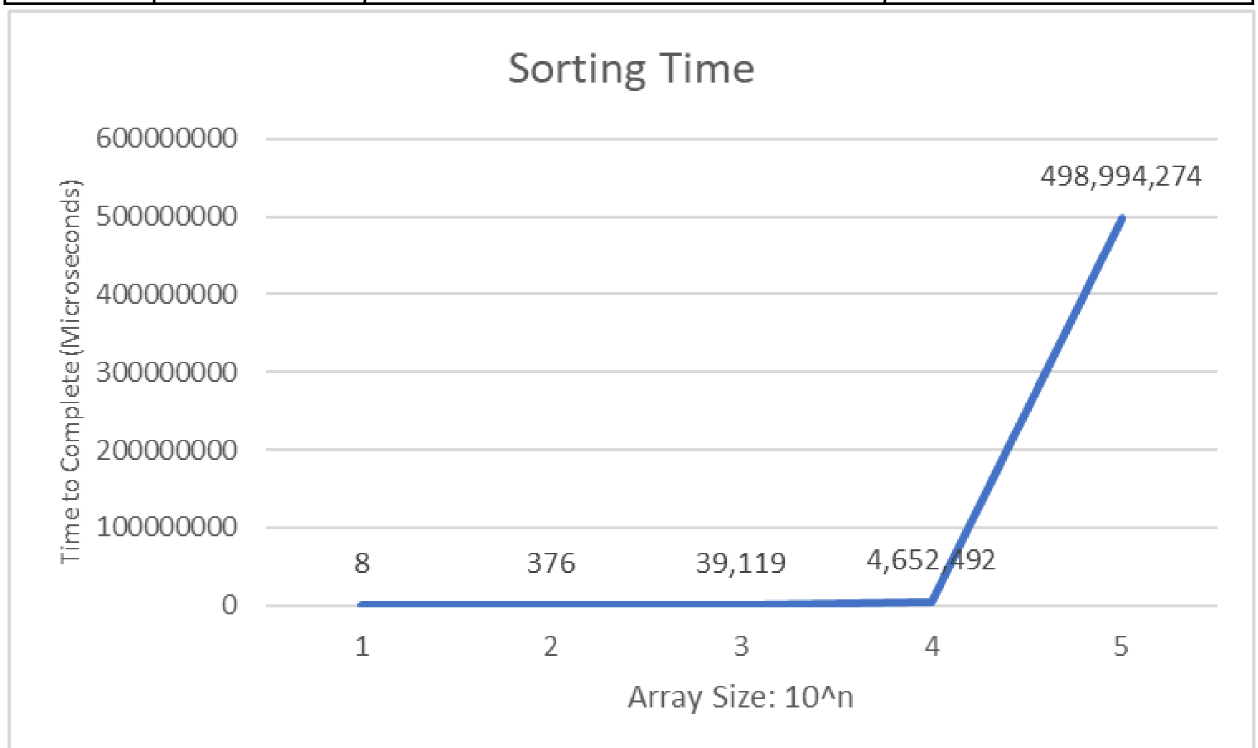
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	7	-
3	1,000	402	57.429
4	10,000	40,285	100.211
5	100,000	4,259,854	105.743
6	1,000,000	499,994,280	117.374



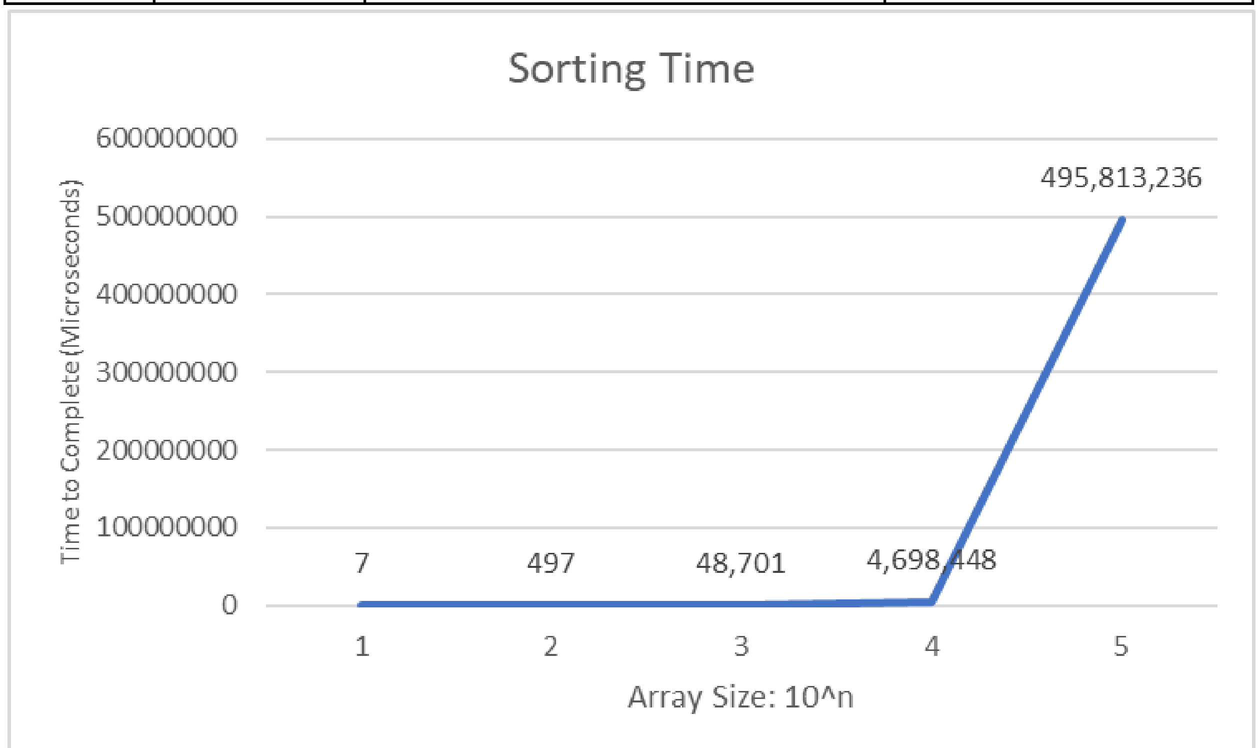
3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	8	-
3	1,000	376	47
4	10,000	39,119	104.040
5	100,000	4,652,492	118.931
6	1,000,000	498,994,274	107.253



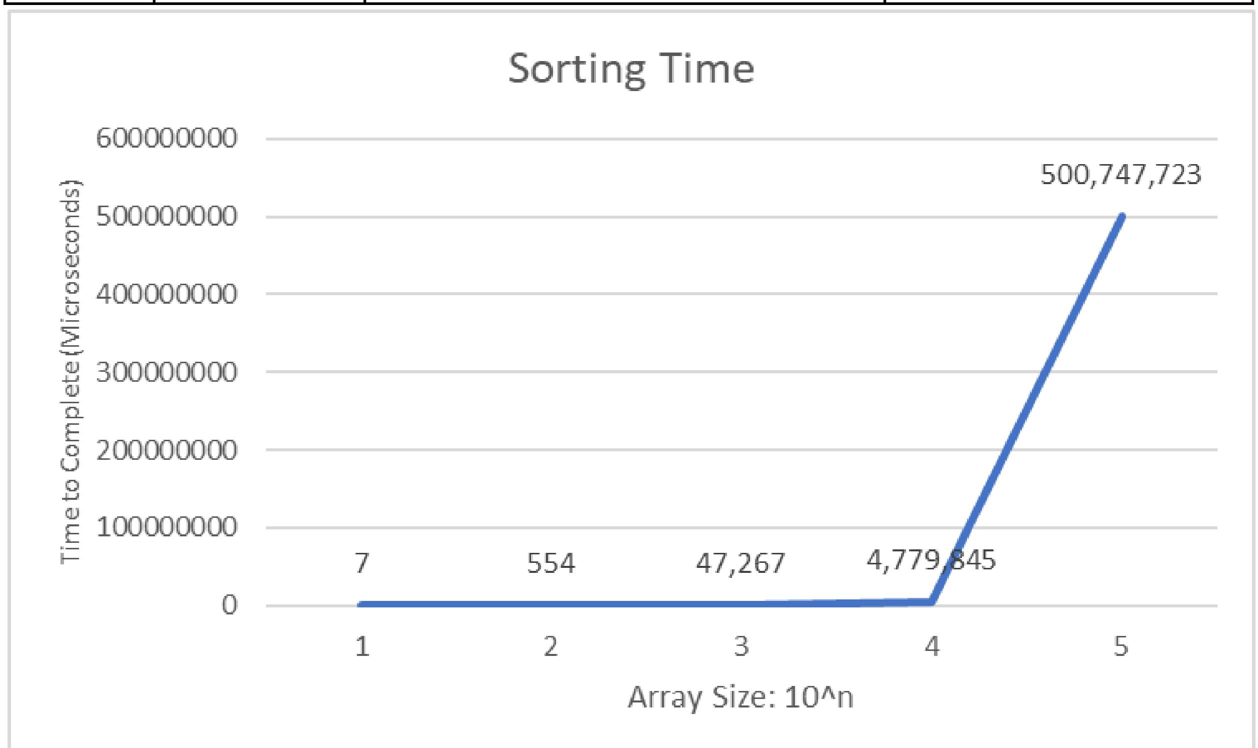
Bubble Sort

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	7	-
3	1,000	497	71
4	10,000	48,701	97.990
5	100,000	4,698,448	96.475
6	1,000,000	495,813,236	105.527



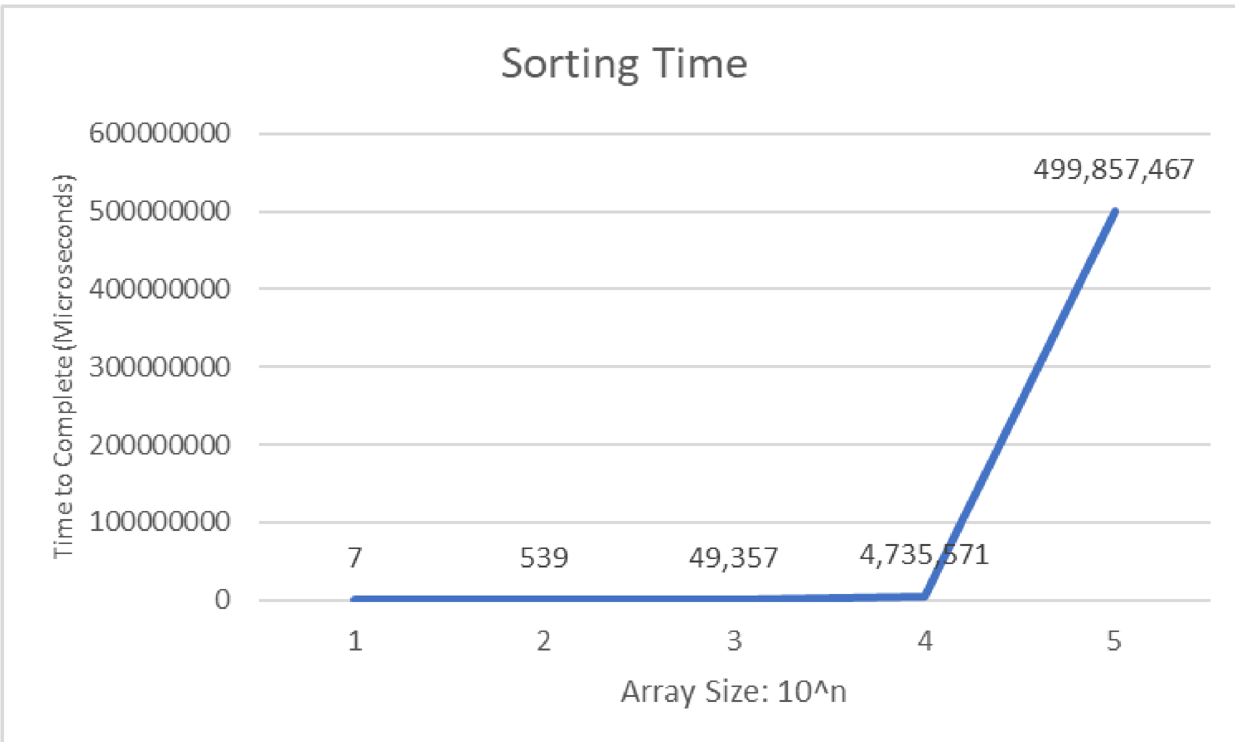
2

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	7	-
3	1,000	554	79.143
4	10,000	47,267	85.319
5	100,000	4,779,845	101.124
6	1,000,000	500,747,723	104.762



3

Size of n	Size of Array	Time to Complete (microseconds)	Factor Increase in Time
2	100	7	-
3	1,000	539	77
4	10,000	49,357	91.571
5	100,000	4,735,571	95.945
6	1,000,000	499,857,467	105.55379



Issue: No complex case analyzed.

Updates: take one array type and one size and see which algorithm did the best/worst; expectations and surprises)

Random Array of size 1,000 | Time Measured in Microseconds

	Trial 1	Trial 2	Trial 3	Average
Shell Sort	71 μs	70 μs	75 μs	72 μs
Insertion Sort	328 μs	354 μs	306 μs	329.33 μs
Selection Sort	455 μs	482 μs	473 μs	470 μs
Bubble Sort	1,018 μs	921 μs	1,386 μs	1108.33 μs

Shell Sort did the best by a landslide, which is no surprise given the average time complexity of these four algorithms:

Algorithm	**Theoretical Time Complexity**
Shell Sort	$\theta(n \log(n))$
Insertion Sort	$\theta(n^2)$
Selection Sort	$\theta(n^2)$
Bubble Sort	$\theta(n^2)$

In class, I've really focused on when it is better or worse to use certain techniques (divide and conquer, transform and conquer, etc.) to design and analyze algorithms. Considering that the insertion sort and selection sort algorithms do not divide to conquer, rather divide to organize, and bubble sort doesn't divide at all, it shows that the divide and conquer technique was the way to go. That is not always the case, however.

It is no surprise at all that Bubble Sort was the slowest considering how many times it iterates over the array and how many comparisons it does. It also doesn't surprise me that insertion sort and selection sort were fairly similar in runtime

GitHub Link: https://github.com/alyssakitchen/DesignOfAlgorithms