Khoi Trinh

DSA 5005

Project 6 Report

Dec 3 2022

1. Introduction

The purpose of this report is to showcase a comparison between 2 sorting algorithms: bubble sort and shell sort. While both can produce a completely sorted array of numbers, their speed differs greatly. Moreover, if time is a constraint for a sorting algorithm; as in; if the algorithm is only allowed to do D number of comparisons, which one would outperform the other?

The 2 metrics that are used to compare these algorithms are: the number of inversions; and the Chebyshev's distance. Below is a brief explanation of these metrics: (these were taken from the project specifications)

Inversion: An inversion in a given random set of numbers σ is a pair $(\sigma(i),\sigma(j))$ such that $i < j$ and $\sigma(i) > \sigma(j)$.
Where i and j are index values and $\sigma(i)$ and $\sigma(j)$ are the numbers at those positions.

Chebyshev's distance: Given random set of numbers $\sigma 1$, and the same set of numbers completely sorted $\sigma 2$, the Chebyshev distance (dl) between $\sigma 1$ and $\sigma 2$ is

$$dl_{\infty}(\sigma 1,\sigma 2) = |\sigma 1 - \sigma 2|_{\infty} = \max |\sigma 1(i) - \sigma 2(i)|.$$

2. Experiment Setup and Results

For this experiment, the algorithms are ran on a number of different arrays of randomly generated numbers. These arrays are 1000, 5000, 1000, 20000, and 30000 randomly generated elements. The number of comparisons D for each algorithm are as follows: n, 2n, 4n, 10n, 50n, 100n, 200n, 1000n, 2000n, 4000n, n^2 with n being the number of elements in the array. For example; for n = 5000; D will be 5000, 10000, 20000, 50000, 250000, 500000, 1000000, 5000000, 10000000, 20000000, and 2500000.

For the following tables; the rows are:

Bub In: number of inversions for bubble sort

Bub dist: Chebyshev's Distance for bubble sort

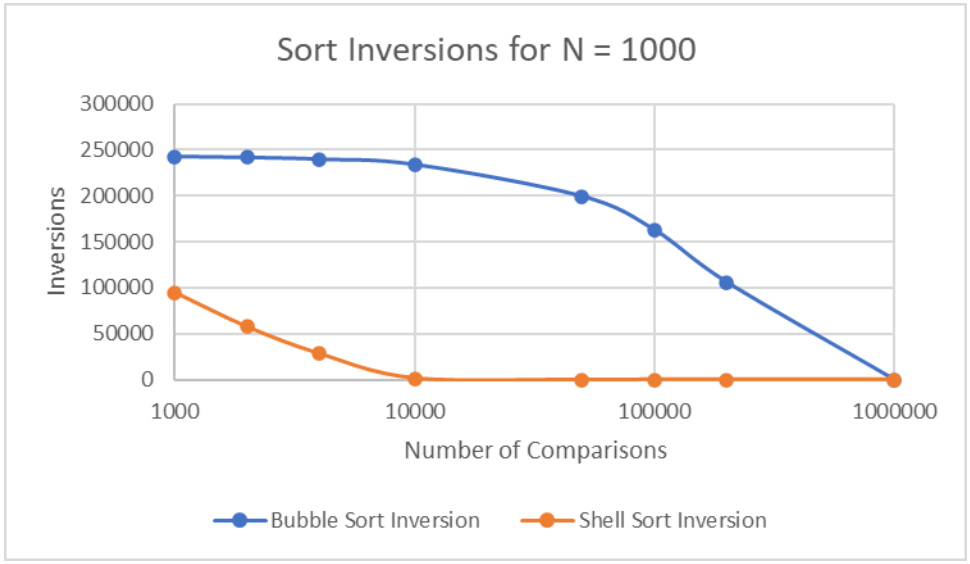
Shell in: number of inversions for shell sort

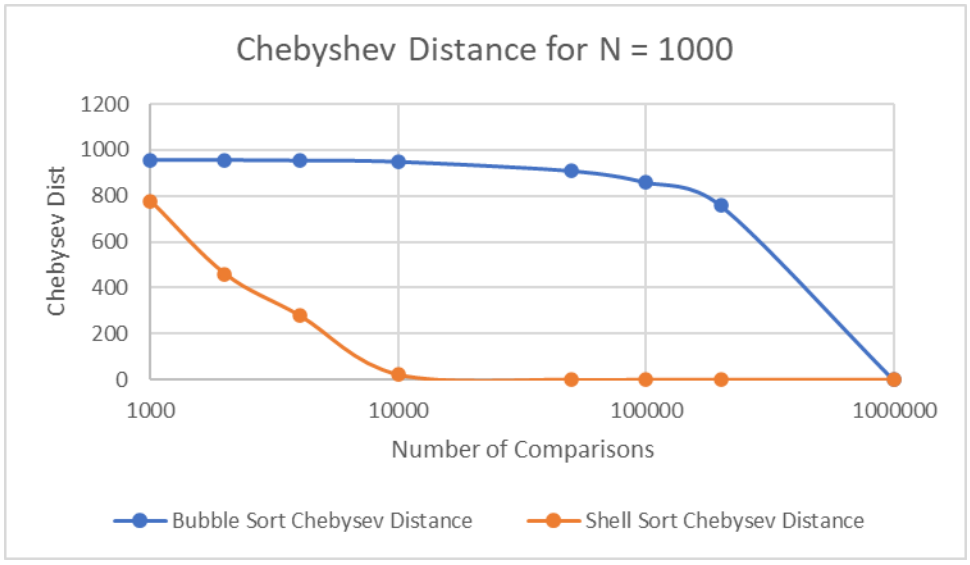
Shell dist: Chebyshev's Distance for shell sort

First, for 1000 elements; the results are as follows; note that due to $1000n = n^2$ for this case; the table are smaller.

D	1000	2000	4000	10000	50000	100000	200000	1000000
Bub In	242578	241590	239634	233899	199344	162701	106049	0
Bub dist	957	956	954	948	908	858	758	0
Shell in	94777	57417	28223	1513	0	0	0	0
Shell dist	778	460	279	22	0	0	0	0

Graphically; note that the x-axis is log scale base 10

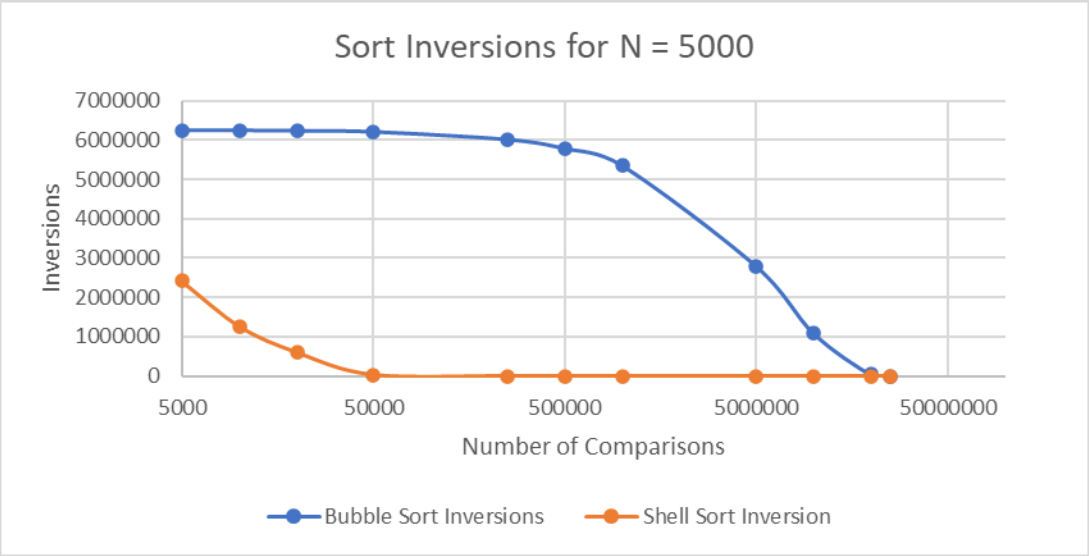


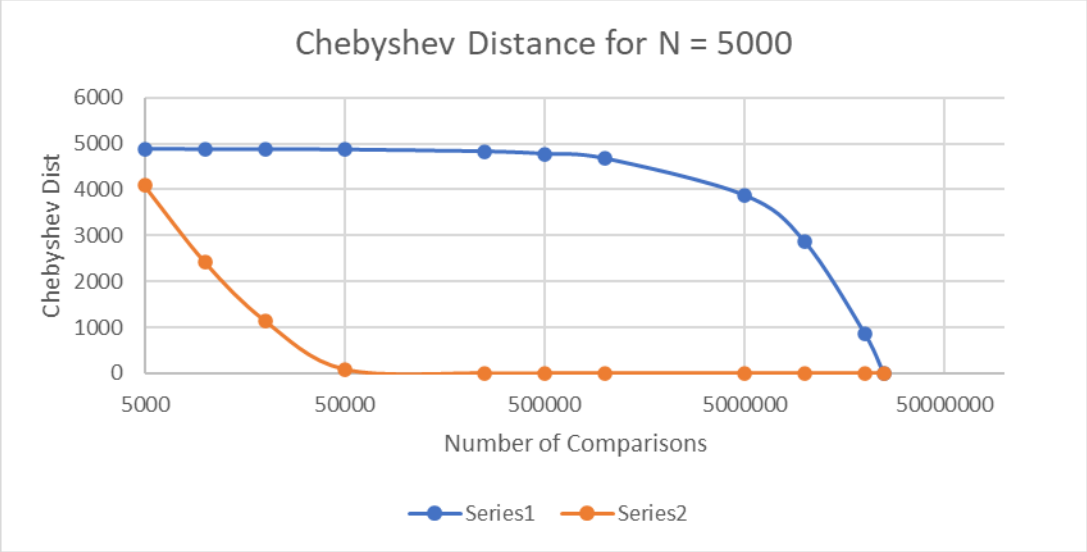


Next, for n = 5000.

5000											
D	5000	10000	20000	50000	250000	500000	1000000	5000000	10000000	20000000	25000000
Bub In	6244258	6239273	6229325	6199665	6007513	5777380	5342594	2795847	1092970	35696	0
Bub dist	4882	4873	4871	4865	4825	4775	4675	3875	2875	874	0
Shell in	2412411	1258170	596621	28629	0	0	0	0	0	0	0
Shell dist	4083	2421	1146	86	0	0	0	0	0	0	0

Graphically:

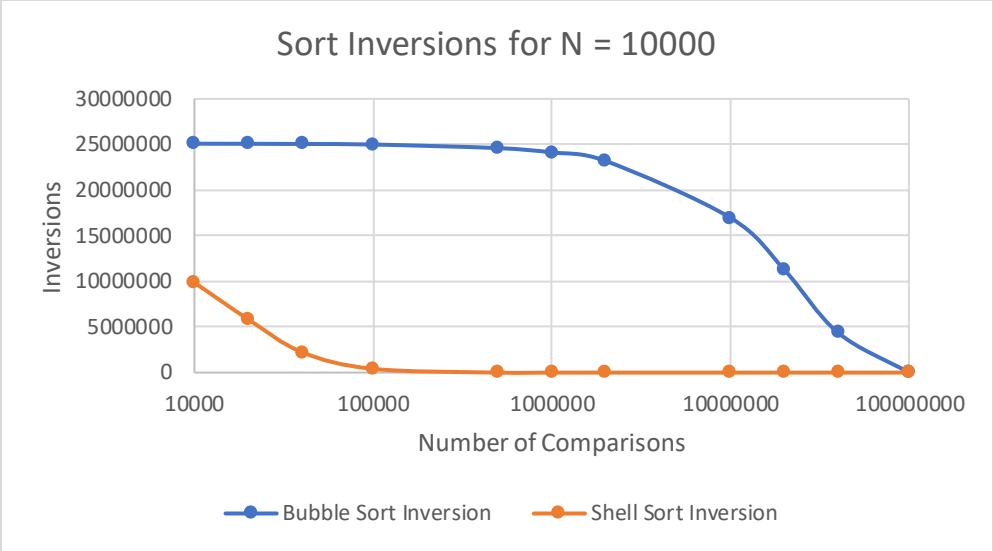


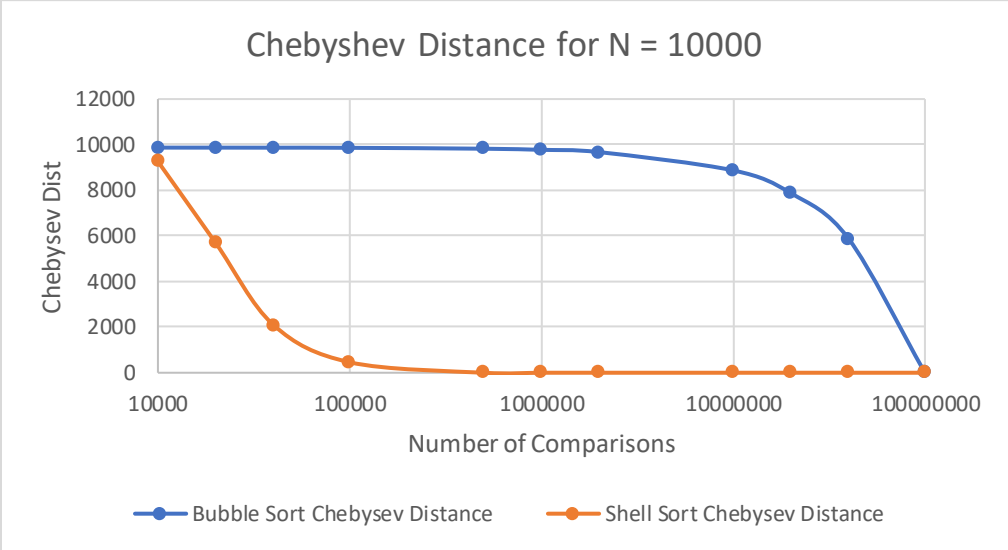


Next, for n = 10000.

D	10000	20000	40000	100000	500000	1000000	2000000	10000000	20000000	40000000	100000000
Bub In	25087773	25077792	25057846	24998192	24606801	24129847	23206382	16976107	11306346	4389104	0
Bub dist	9872	9871	9869	9863	9823	9773	9673	8873	7873	5873	0
Shell in	9863522	5845464	2239242	410971	0	0	0	0	0	0	0
Shell dist	9263	5694	2098	460	0	0	0	0	0	0	0

Graphically:

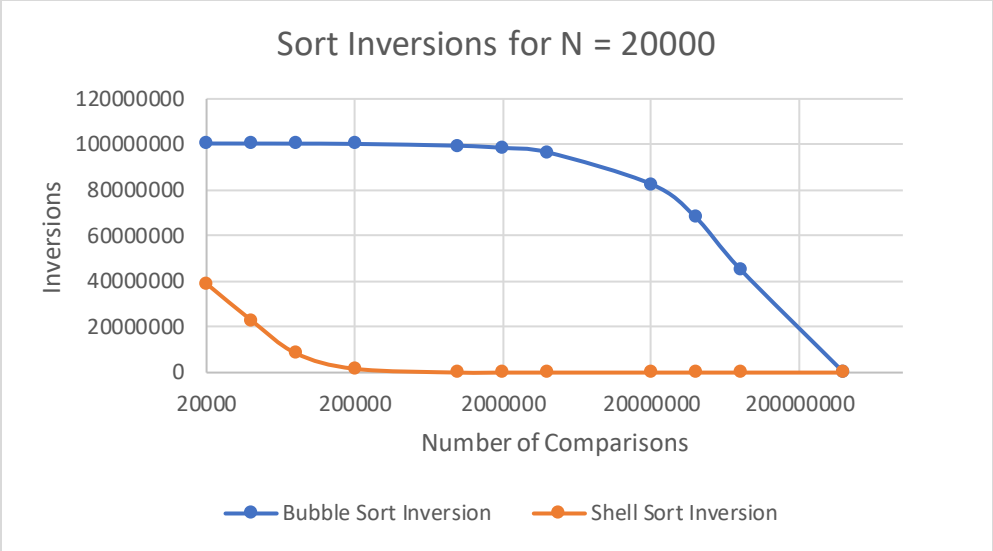


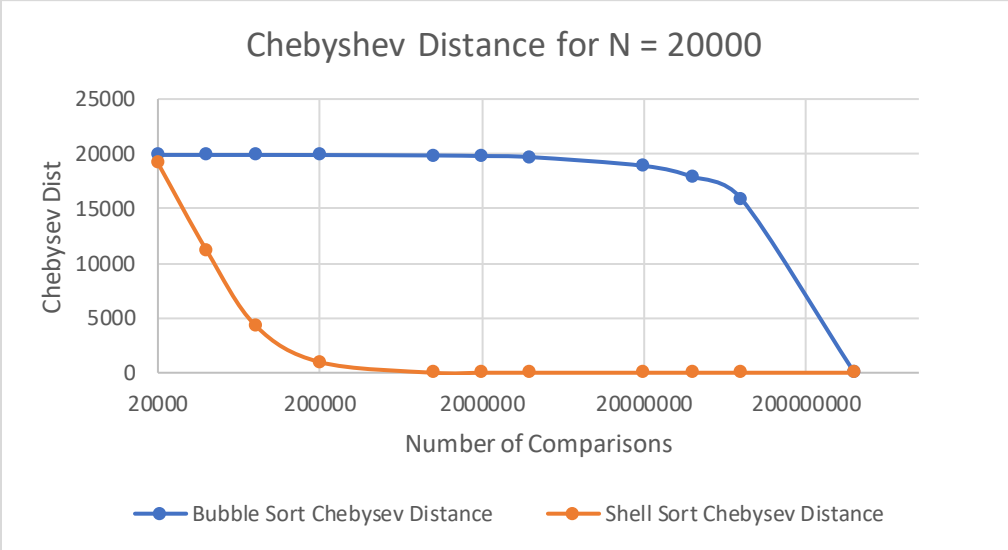


Next, n = 20000.

D	20000	40000	80000	200000	1000000	2000000	4000000	20000000	40000000	80000000	400000000
Bub In	100439708	100419725	100379780	100260109	99469273	98494928	96583445	82672015	68066000	45303228	0
Bub dist	19897	19896	19894	19888	19848	19798	19698	18898	17898	15898	0
Shell in	38965040	22966703	8485760	1655772	0	0	0	0	0	0	0
Shell dist	19149	11165	4320	973	0	0	0	0	0	0	0

Graphically:

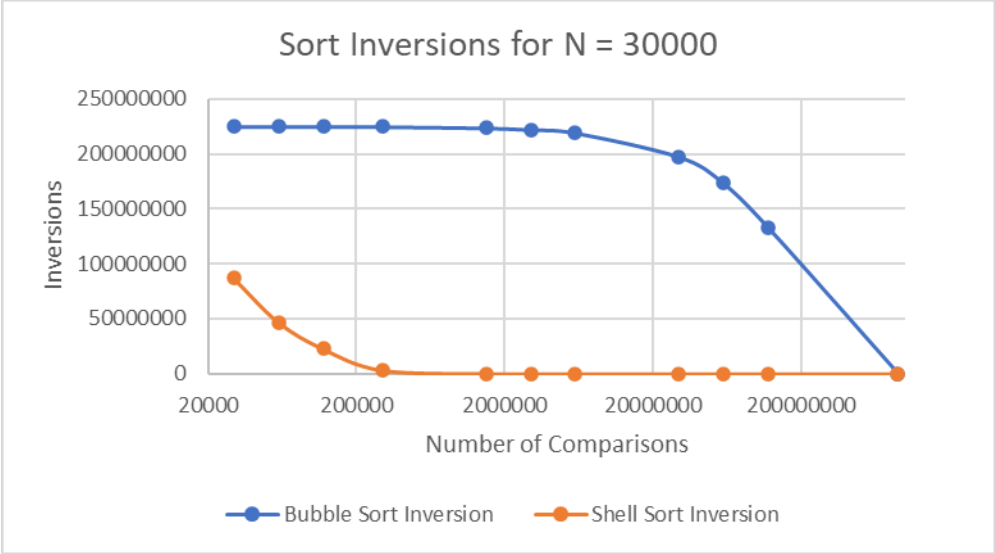


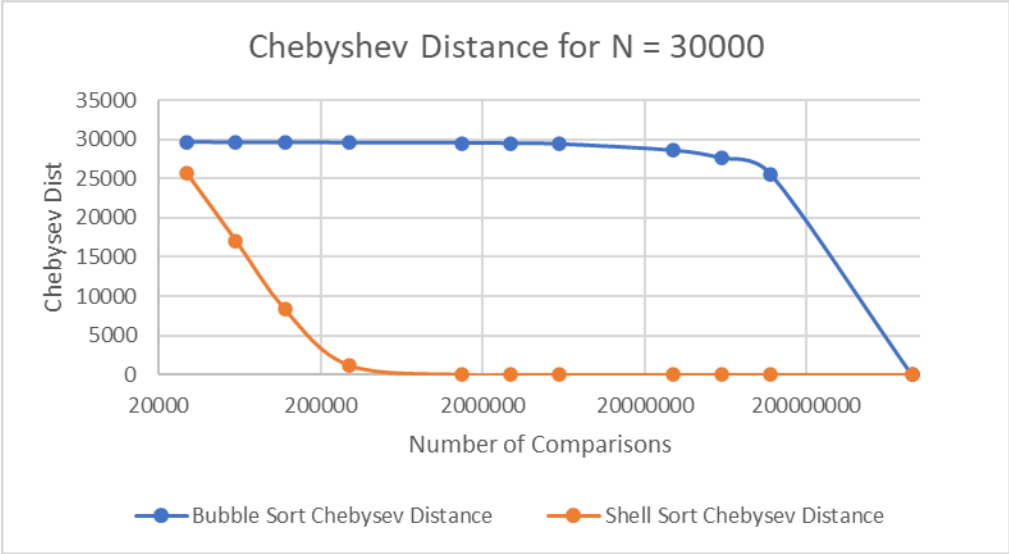


Finally, n = 30000

D	30000	60000	120000	300000	1500000	3000000	6000000	30000000	60000000	120000000	900000000
Bub In	224946663	224916682	224856738	224677094	223486629	222013679	219109218	197388456	173388684	133035075	0
Bub dist	29669	29595	29593	29587	29547	29497	29397	28597	27597	25597	0
Shell in	86704952	46858440	22599302	3227824	0	0	0	0	0	0	0
Shell dist	25659	17100	8365	1196	0	0	0	0	0	0	0

Graphically:





3. Analysis and Conclusion

From the graphs and tabulated results; a clear pattern can be seen. For shell sort, the number of inversions and Chebyshev Distance reaches 0 when D = 50n; meaning that after 50n comparisons, shell sort algorithm will completely sort an array. On the contrary, bubble sort inversions and Chebyshev Distance don't reach 0 until $D = n^2$; which is much longer than shell sort. In conclusion, given the same number of comparison, shell sort would be a better choice for a sorting algorithm.