HACETTEPE UNIVERSITY DEPARTMENT OF

COMPUTER ENGINEERING

BBM 204 ASSIGNMENT 1

Name Surname – Number

Subject : Analysis of Algorithms

Programming Language : Java

1. Record Execution Time for each Algorithm

Since Bitonic Sort can sorting at the powers of two, the tables were created at the powers of 2.

If array size is not equal to the power of two, bitonic sort does not work properly as in the example image

Exp image:

```
public static void main(String[] args) {
BitonicSort ob = new BitonicSort();
    int up = 1;
    int[] arr = {8, 4, 1, 56, 3,  23, -7, 0,-999};
    ob.sort(arr, arr.length,up);

    System.out.println("sorted array");
    for (int j : arr) System.out.print(j + " ");
}
```

```
sorted array
-7 0 1 3 4 8 23 56 -999
```

The times recorded in the whole table are in seconds.

1.1) Random Array (Average)

Algorithms/n	32	64	128	256	512	1024
Comb Sort	0.0	0.0	0.0	0.0	0.0	0.001
Gnome Sort	0.0	0.0	0.0	0.001	0.002	0.003
Shaker Sort	0.0	0.0	0.0	0.001	0.002	0.001
Stooge Sort	0.0	0.001	0.001	0.005	0.042	0.209
Bitonic Sort	0.0	0.001	0.0	0.0	0.0	0.0

Algorithms/n	2048	4096	8192	16384	32768
Comb Sort	0.001	0.001	0.001	0.001	0.002
Gnome Sort	0.003	0.013	0.044	0.174	0.76
Shaker Sort	0.002	0.007	0.046	0.199	1.059
Stooge Sort	1.869	16.782	151.04	272.284	2469.751
Bitonic Sort	0.0	0.001	0.002	0.003	0.007

1.2) All Array Member is 0

Algorithms/n	32	64	128	256	512	1024
Comb Sort	0.0	0.0	0.0	0.0	0.0	0.00
Gnome Sort	0.0	0.0	0.0	0.0	0.0	0.00
Shaker Sort	0.0	0.0	0.0	0.0	0.0	0.00
Stooge Sort	0.0	0.001	0.001	0.006	0.049	0.216
Bitonic Sort	0.0	0.0	0.0	0.0	0.0	0.0

Algorithms/n	2048	4096	8192	16384	32768
Comb Sort	0.0	0.0	0.0	0.0	0.001
Gnome Sort	0.0	0.0	0.0	0.0	0.001
Shaker Sort	0.0	0.0	0.0	0.0	0.0
Stooge Sort	1.86	16.706	150.688	275.677	2479.137
Bitonic Sort	0.0	0.0	0.001	0.002	0.003

1.3) Sorting Descending Array

Algorithms/n	32	64	128	256	512	1024
Comb Sort	0.0	0.0	0.0	0.0	0.0	0.0
Gnome Sort	0.0	0.0	0.001	0.002	0.002	0.001
Shaker Sort	0.0	0.0	0.0	0.001	0.001	0.001
Stooge Sort	0.0	0.001	0.001	0.005	0.046	0.211
Bitonic Sort	0.0	0.0	0.0	0.0	0.0	0.0

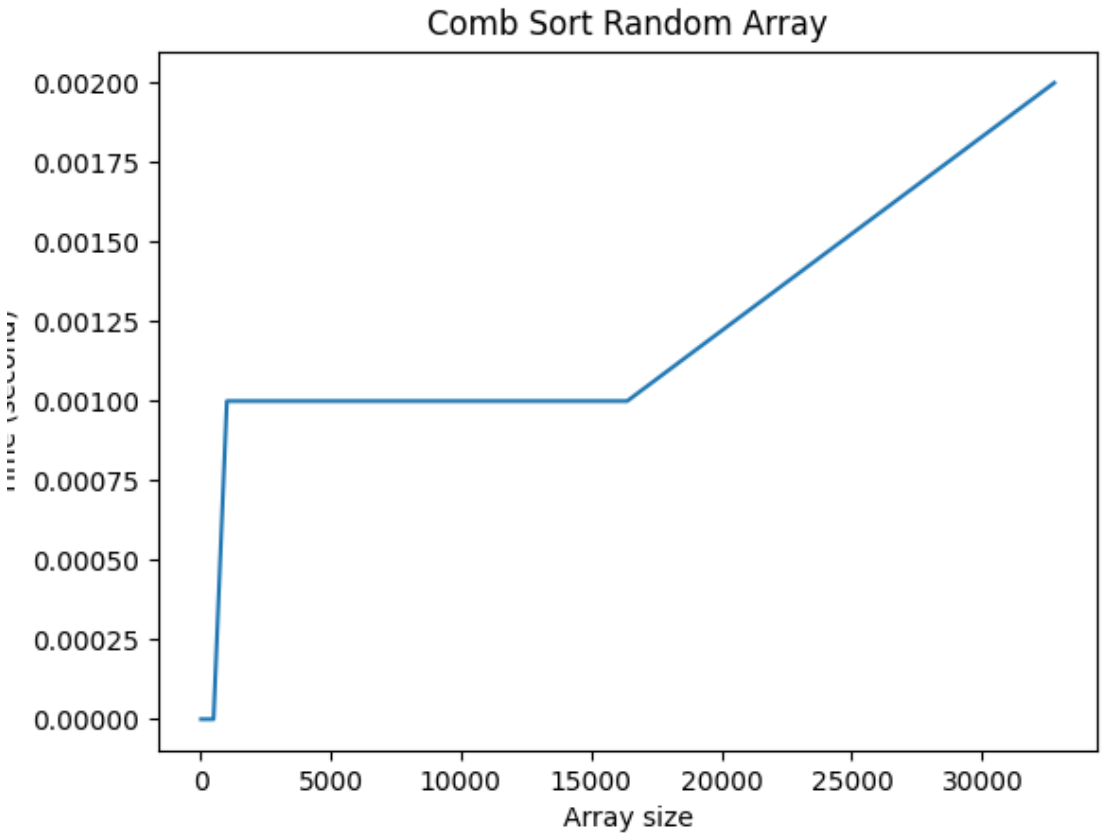
Algorithms/n	2048	4096	8192	16384	32768
Comb Sort	0.0	0.0	0.001	0.001	0.002
Gnome Sort	0.005	0.022	0.088	0.346	1.398
Shaker Sort	0.002	0.007	0.027	0.105	0.422
Stooge Sort	1.872	16.817	151.692	275.246	2480.975
Bitonic Sort	0.0	0.0	0.001	0.002	0.004

1.4) Sorting Ascending Array

Algorithms/n	32	64	128	256	512	1024
Comb Sort	0.0	0.0	0.0	0.0	0.0	0.00
Gnome Sort	0.0	0.0	0.0	0.0	0.0	0.00
Shaker Sort	0.0	0.0	0.0	0.0	0.0	0.00
Stooge Sort	0.0	0.001	0.001	0.004	0.042	0.209
Bitonic Sort	0.0	0.0	0.0	0.0	0.0	0.0

Algorithms/n	2048	4096	8192	16384	32768
Comb Sort	0.0	0.0	0.0	0.001	0.001
Gnome Sort	0.0	0.0	0.0	0.0	0.001
Shaker Sort	0.0	0.0	0.0	0.0	0.0
Stooge Sort	1.867	16.742	150.896	276.118	2499.968
Bitonic Sort	0.0	0.0	0.001	0.002	0.003

2. Comb Sort Algorithm



Best case time complexity: Θ(n log n)

Average case time complexity: Ω(n*n/2^p), p is a number of increment

Worst case time complexity: O(n*n)

Space complexity: Θ(1) auxillary space

Comb Sort is an in-place sorting algorithm

Comb Sort doesn't require additional space for sorting the lists.

Comb Sort easy to implement sorting algorithm

Combsort with different endings changes to a more efficient sort when the data is almost sorted (when the gap is small).

Comb sort is not a stable sorting algorithm as it does not sort the repeated elements in the same order as they appear in the input.

Comb Sort has no recursive function calls

3. Gnome Sort Algorithm

Best case time complexity: Θ(n)

Average case time complexity: Θ (n*n)

Worst case time complexity: Θ (n*n)

Space complexity: Θ(1) auxillary space

Gnome Sort is an in-place sorting algorithm

Gnome Sort does not require any extra storage.

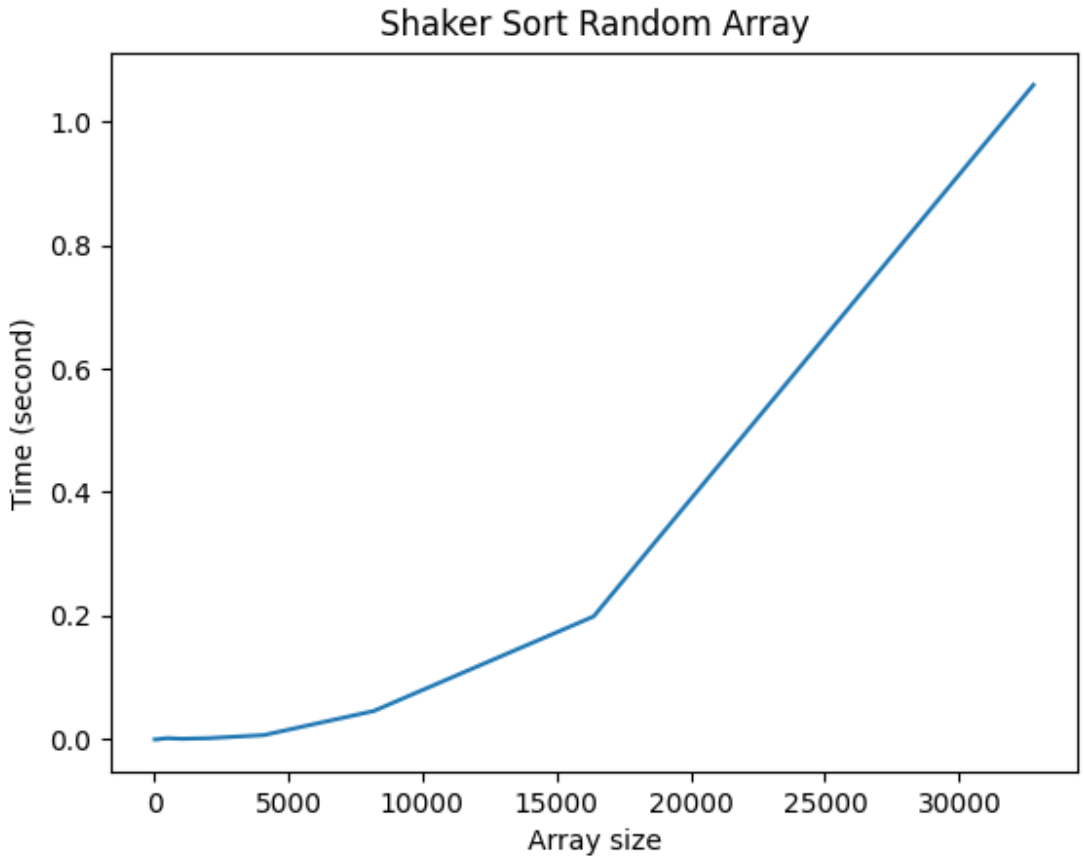
Gnome sort is similar to Insertion sort

Gnome sort is a stable sorting algorithm.

It is conceptually simple, requiring no nested loops.

The average running time is O(n^2). If the list is initially almost sorted, average running time tends towards O(n). Sorting algorithm is adaptive and performs better if the array is already/partially sorted.

4. Shaker Sort Algorithm



Best case time complexity: Θ(n)

Average case time complexity: Θ(n*n)

Worst case time complexity: Θ(n*n)

Space complexity: Θ(1)

Shaker Sort is an in-place sorting algorithm

Shaker Sort does not require any extra storage.

Best case occurs when array is already sorted.

Shaker Sort is stable

5. Stooge Sort Algorithm

Best-Case time complexity = O(n ^ log(3) /log(1.5))

Average time complexity = O(n ^ log(3) /log(1.5))

Worst-Case time complexity = O(n ^ log(3) /log(1.5))

Worst-case space complexity = O(n)

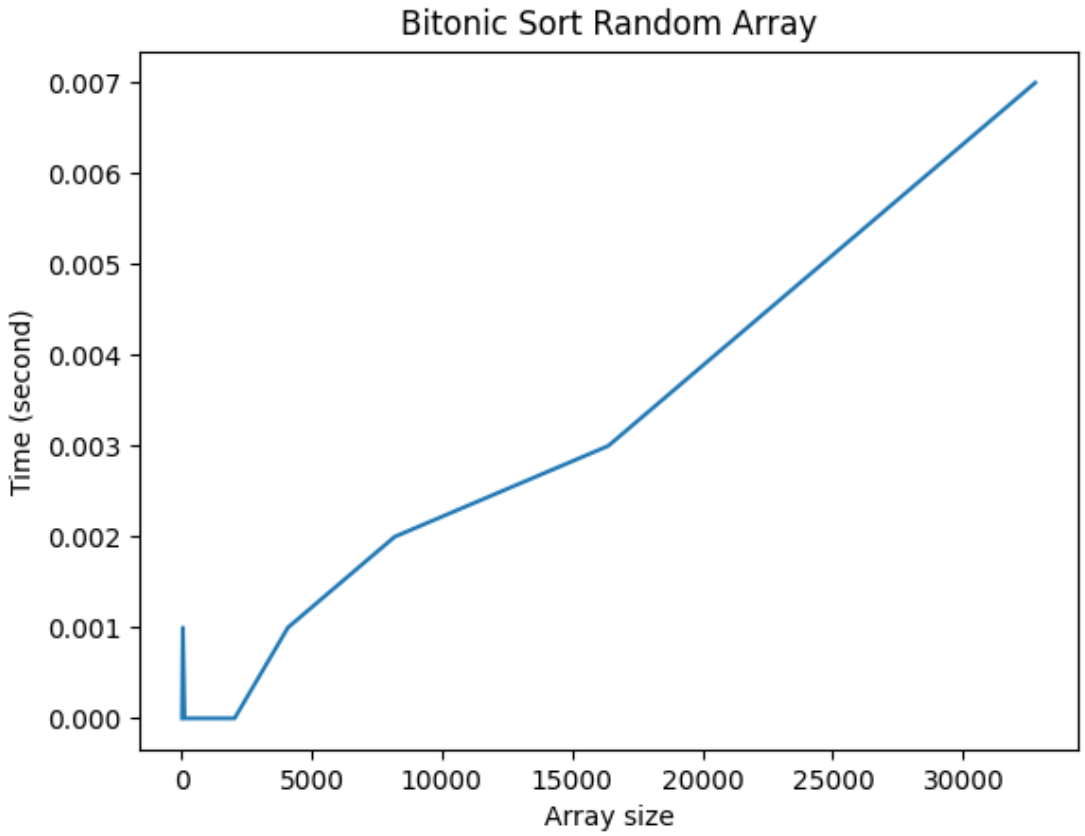
Stooge Sort has the same time complexity under any case.

Stooge Sort is used to store the input array

Stooge sort is not a stable sorting algorithm. It is because the elements with identical values do not appear in the same order in the output array as they were in the input array.

Stooge sort is not an adaptive sorting algorithm. This is because it does not perform better in the case when the array is already/almost sorted.

6. Bitonic Sort



Best case time complexity: Θ(logn*logn) parallel time

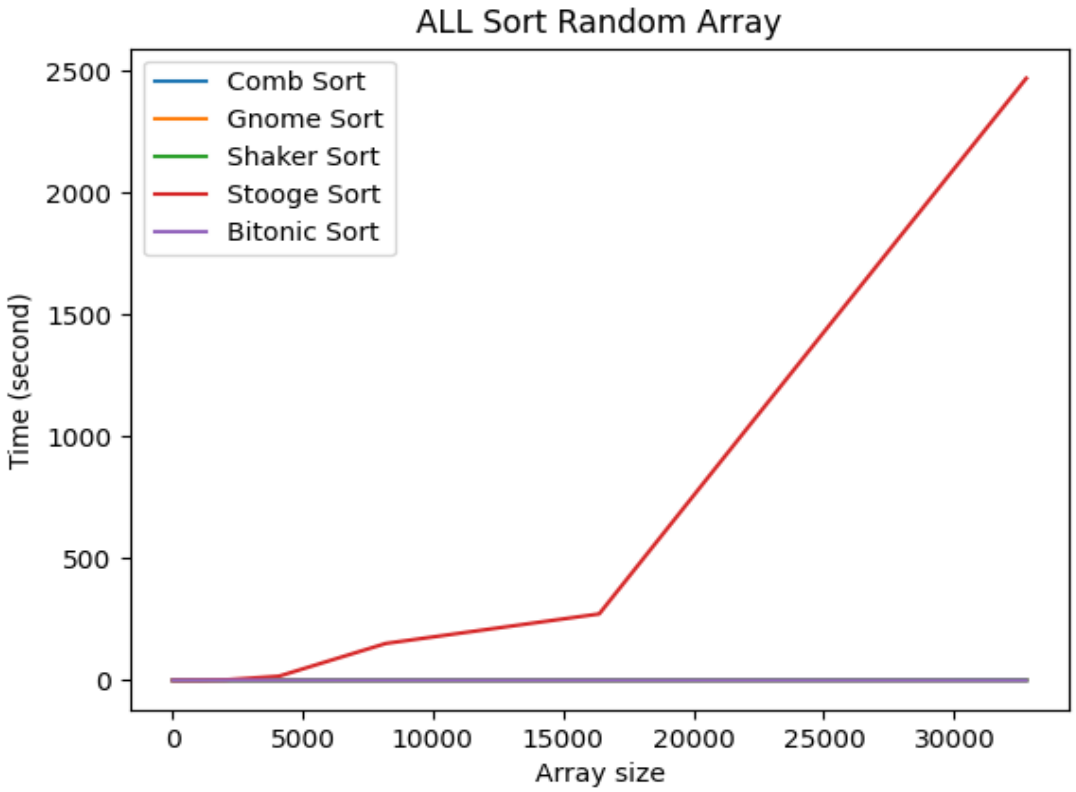
Average case time complexity: Θ(logn*logn) parallel time

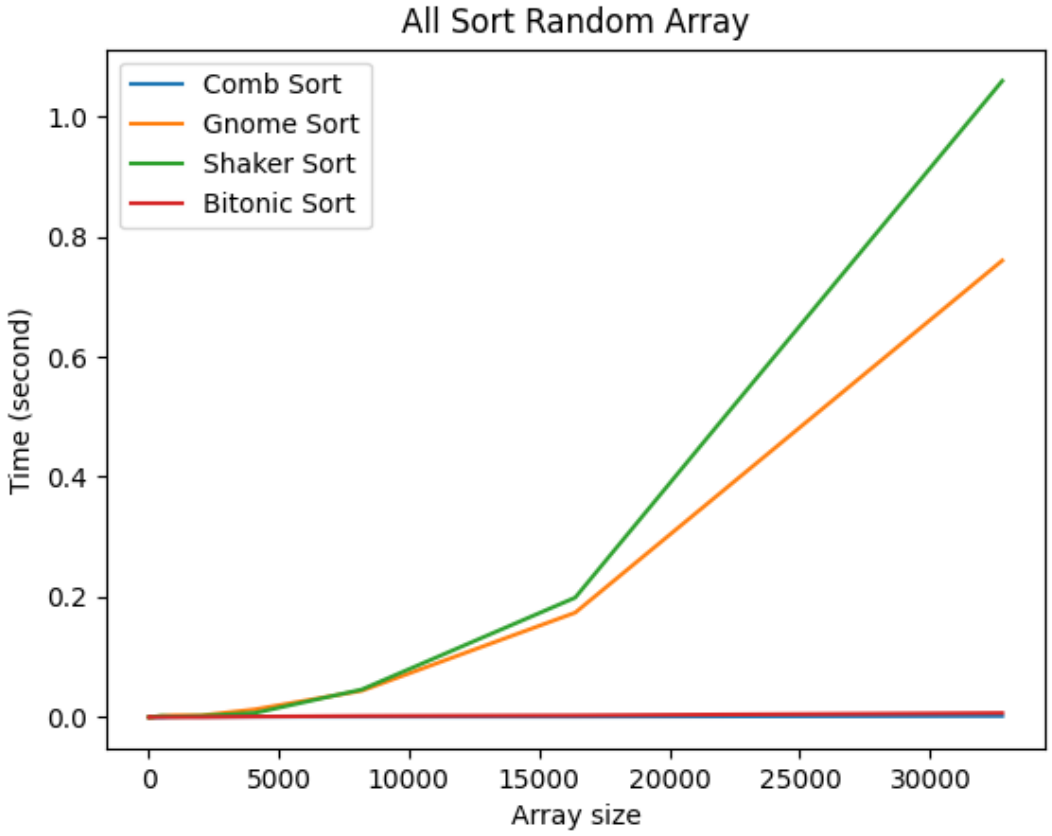
Worst case time complexity: Θ(logn*logn) parallel time

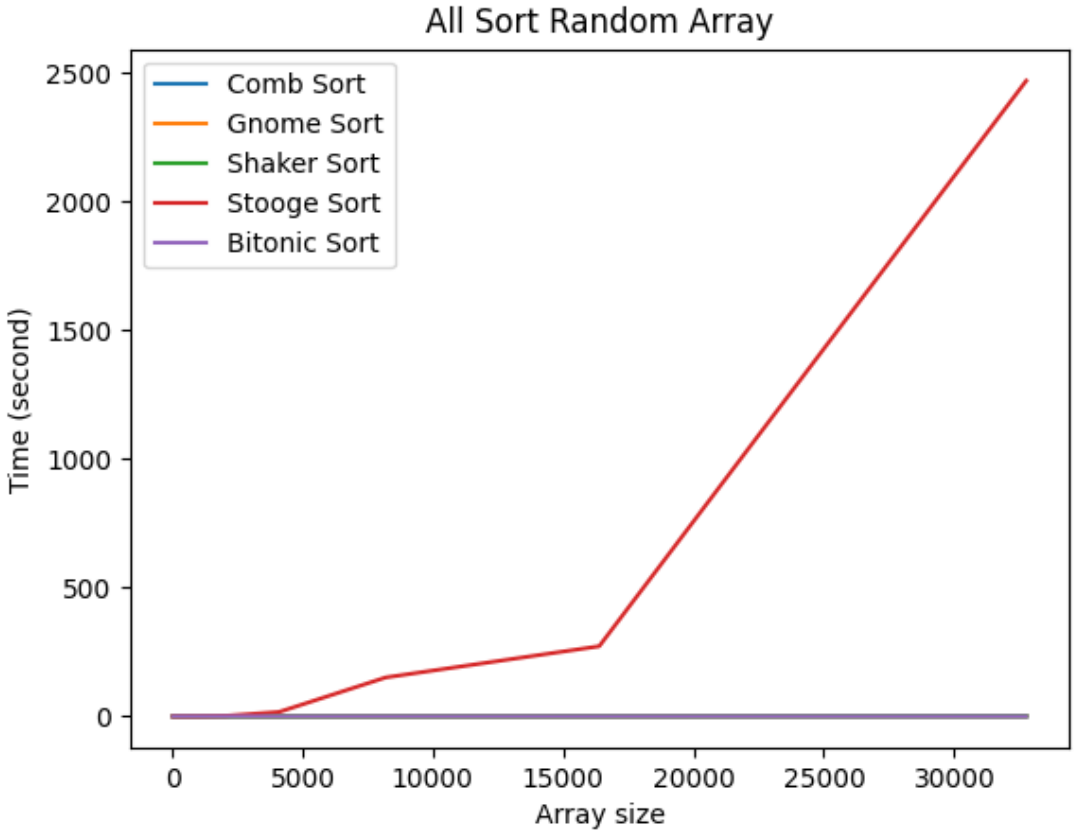
Worst case time complexity: Θ(n*logn*logn) non - parallel time

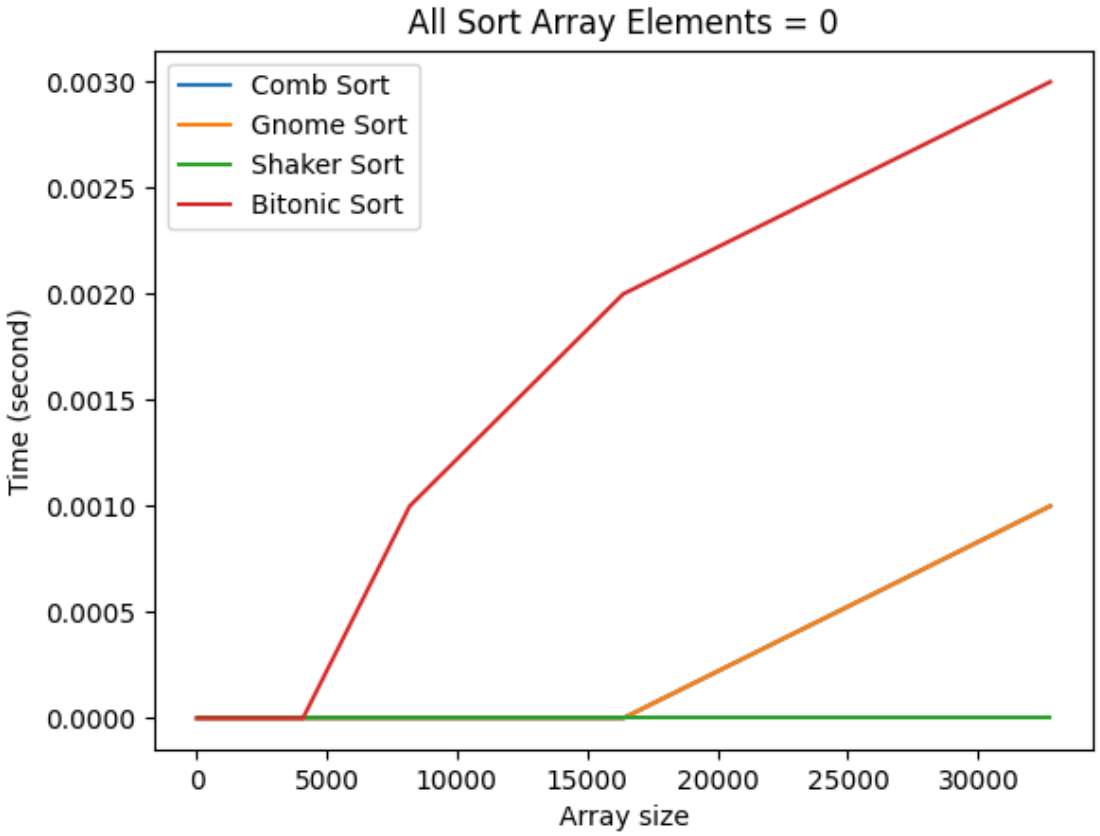
Space complexity: Θ(n*logn*logn)

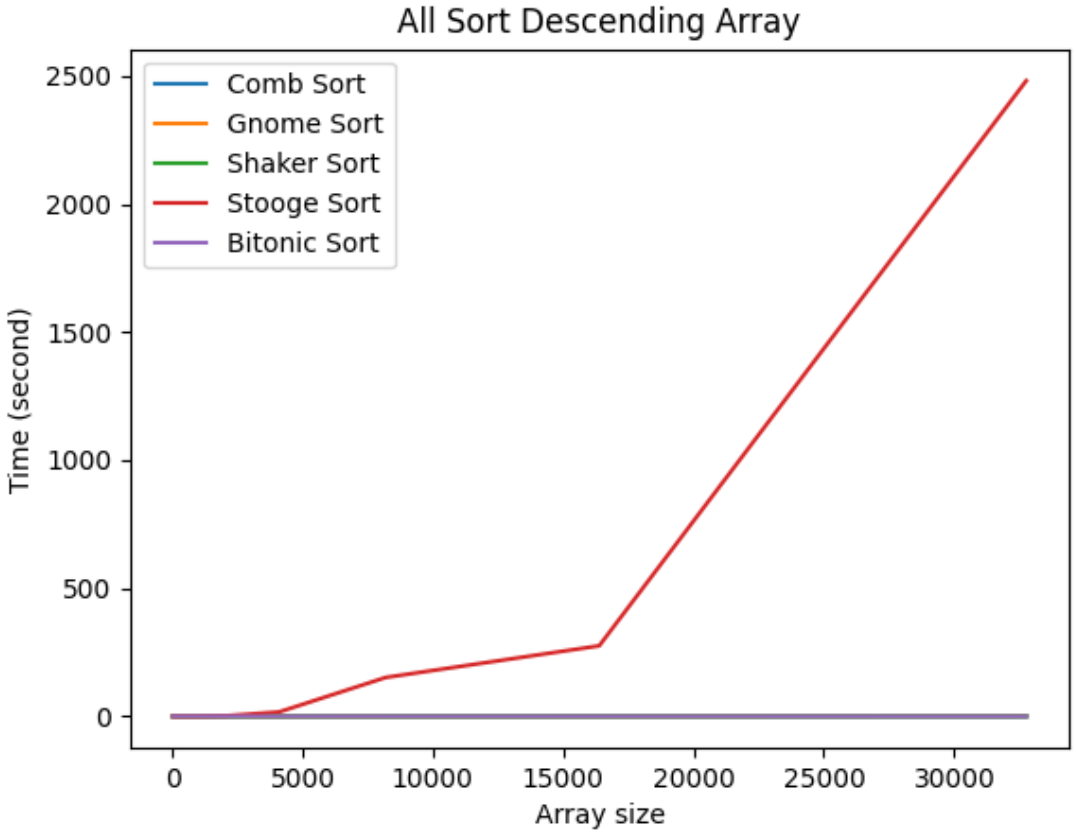
Bitonic sort is not stable.

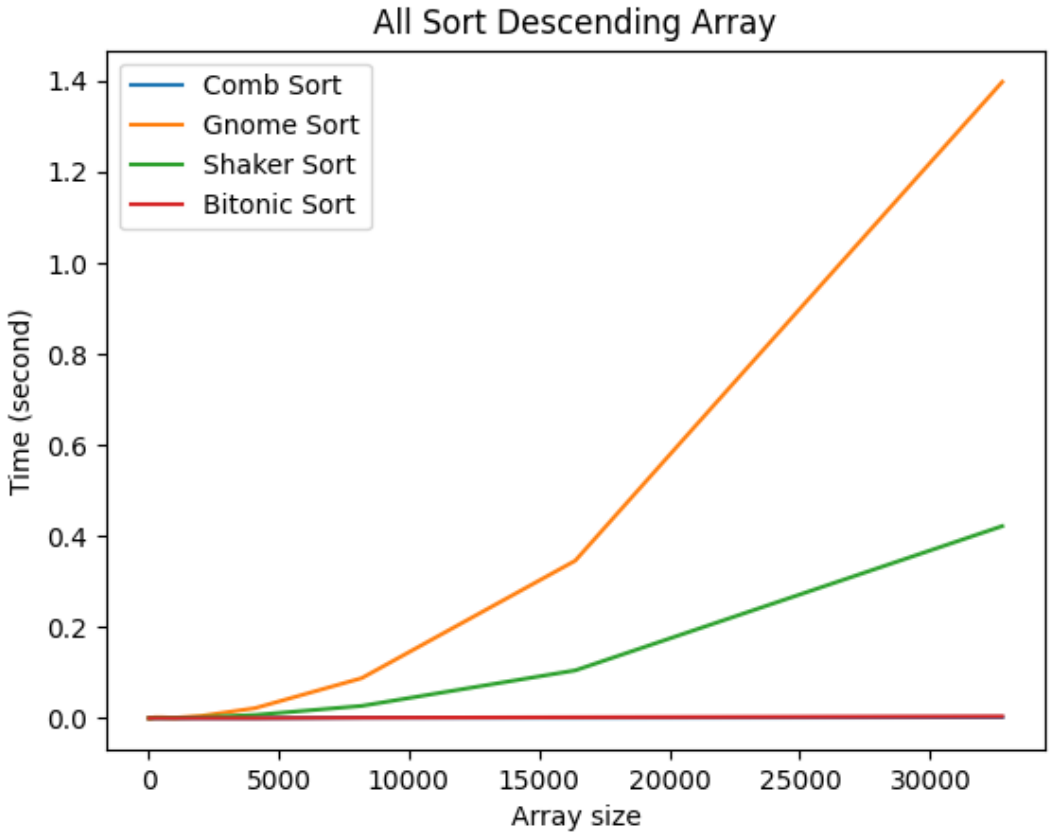

ALL Sort Random Array
Comb Sort
Gnome Sort
Shaker Sort
Stooge Sort
Bitonic Sort
Time (second)
2500
2000
1500
1000
500
0
0
5000
10000
15000
20000
25000
30000
Array size

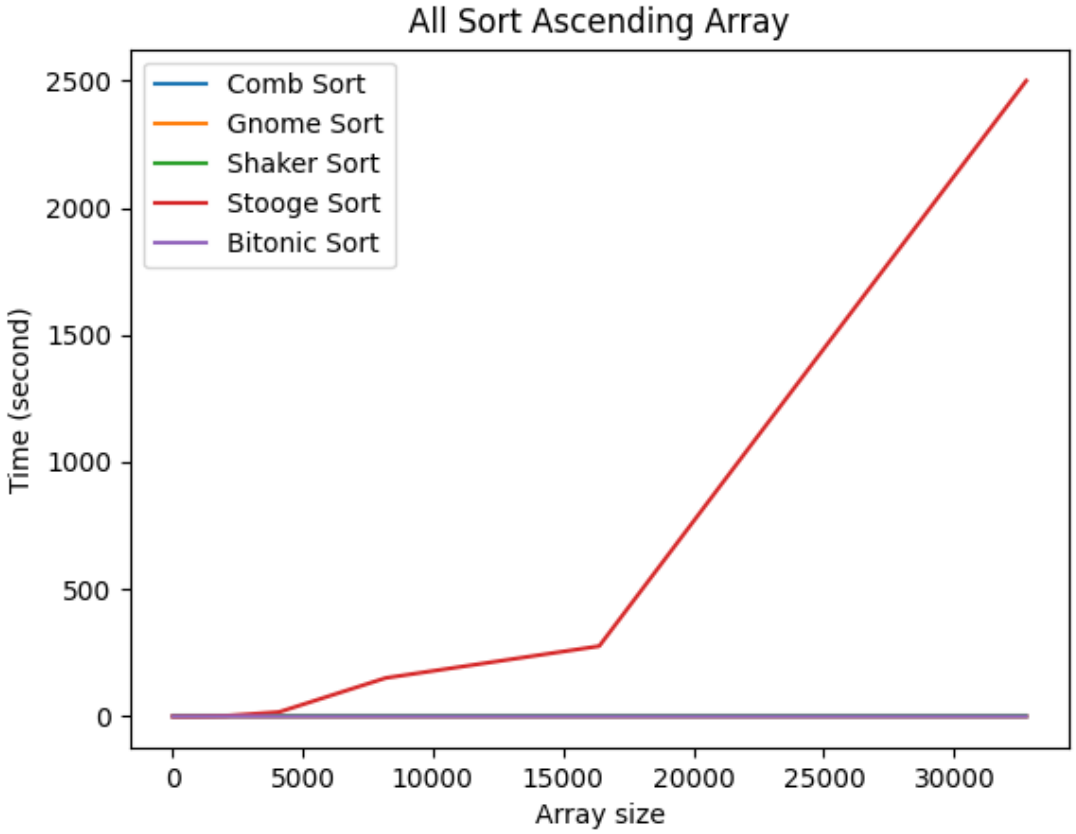

All Sort Random Array
Comb Sort
Gnome Sort
Shaker Sort
Bitonic Sort
Time (second)
1.0
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000
25000
30000
Array size

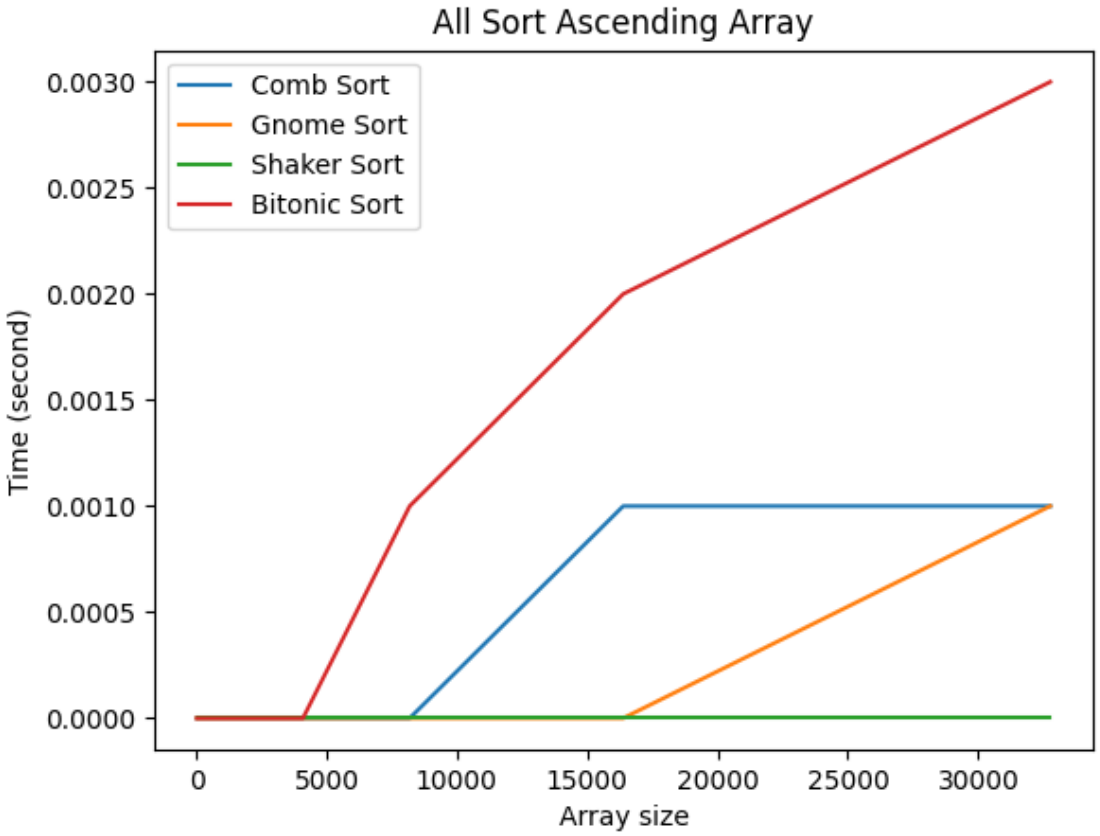

All Sort Random Array
Comb Sort
Gnome Sort
Shaker Sort
Stooge Sort
Bitonic Sort
Time (second)
2500
2000
1500
1000
500
0
0
5000
10000
15000
20000
25000
30000
Array size


All Sort Array Elements = 0
Comb Sort
Gnome Sort
Shaker Sort
Bitonic Sort
Time (second)
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
5000
10000
15000
20000
25000
30000
Array size


All Sort Descending Array
Comb Sort
Gnome Sort
Shaker Sort
Stooge Sort
Bitonic Sort
Time (second)
2500
2000
1500
1000
500
0
0
5000
10000
15000
20000
25000
30000
Array size


All Sort Descending Array
Comb Sort
Gnome Sort
Shaker Sort
Bitonic Sort
Time (second)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000
25000
30000
Array size


All Sort Ascending Array
Comb Sort
Gnome Sort
Shaker Sort
Stooge Sort
Bitonic Sort
Time (second)
2500
2000
1500
1000
500
0
0
5000
10000
15000
20000
25000
30000
Array size


All Sort Ascending Array
Comb Sort
Gnome Sort
Shaker Sort
Bitonic Sort
Time (second)
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
5000
10000
15000
20000
25000
30000
Array size