Hacettepe University

Department of Computer Engineering

BBM204 Programming Lab.

Spring 2021

Programming Assignment 1

Emirhan Topcu

21827899

Idea Behind This Assignment

The purpose of this experiment is trying five sorting algorithms on specific input sizes, noting down the executing times and analysing them to see if we can get the expected results. Along with that, we test the algorithms if they are stable.

Random Number Generation Strategy

I created my objects with two attributes. One is for comparing and sorting, the other is for testing the stability of the sort algorithm.

I decided my input sizes to be powers of 2 because the bitonic sort can only give true outputs if the input size is a power of 2. So I created an array of powers of 2 until 8192 for I will use it as my maximum input size.

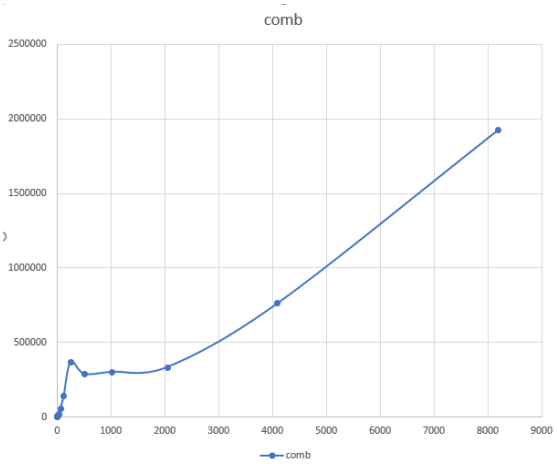
I designed a method to iterate a loop that creates an object with random valued attributes each time. It takes a number as an input and creates a list with that size.

Determining Stability

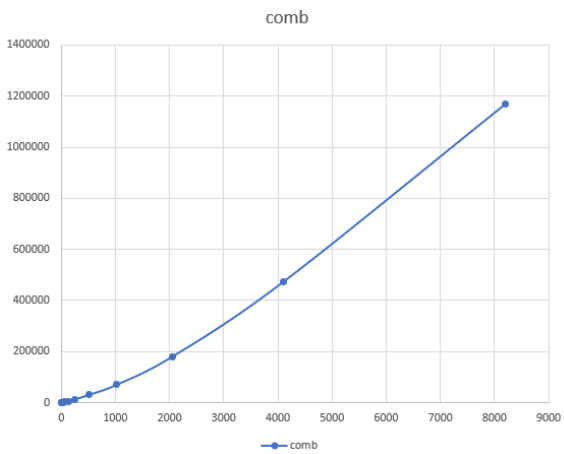
I created a custom list for checking stability. It’s size is 16 and every element in it is customly given. The program checks if the order of the second attributes of the same valued objects in the list are the same before and after the sorting process.

Algorithm Analysis

1. Comb Sort



Average



Worst

Analysis

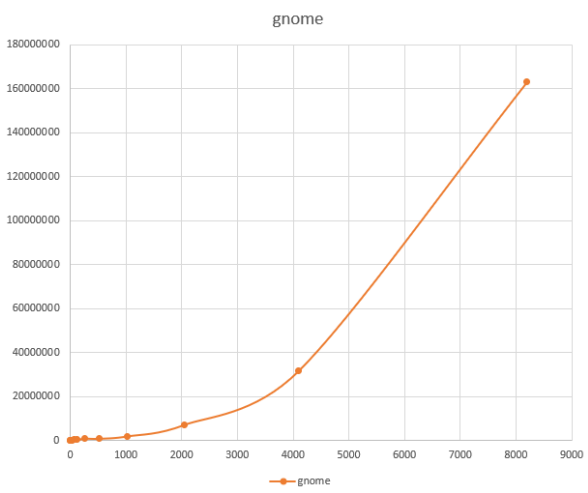
Comb sort has nested loops in it, it's average and worse case time complexities are $O(n^2)$. This can be seen clearly in the graph that demonstrates the worst case execution times. The average case graph is not smooth, but I think the reason for that is my hardware properties. Comb sort is an in place sorting algorithm so it's space complexity is O(1).

Stability

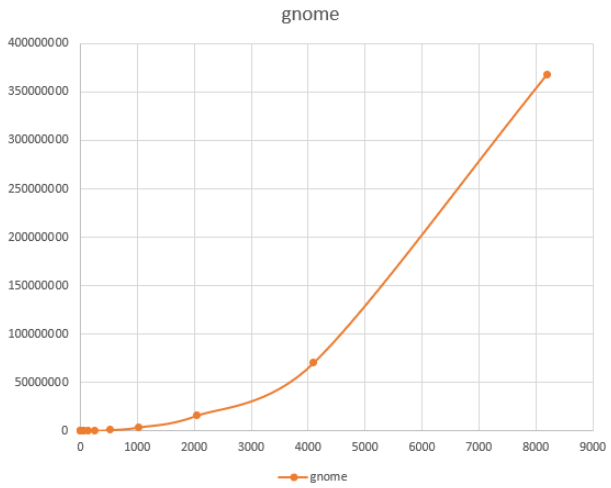
The way this algorithm works makes it impossible to be stable. Swapping occurs on random places in the array so the order of the objects with same values can change.

```
Comb sort stability analysis:
2d 1a 2e 1b 3f 1c 4h 3g 5j 3k 5l 1m 3x 6q 7t 1u
1u 1a 1m 1b 1c 2e 2d 3f 3x 3k 3g 4h 5l 5j 6q 7t
Comb sort is unstable
```

2. Gnome Sort



Average



Worst

Analysis

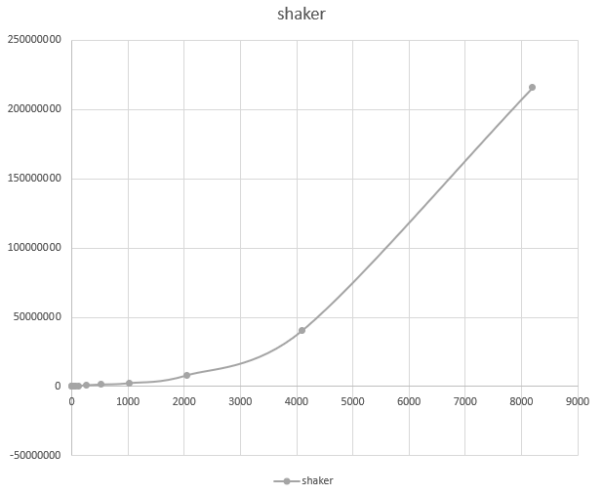
Gnome sort is similar to insertion sort but it tries to be more efficient and does the same thing but adds to this by looping back when a swap occurs, saving an iteration. It's complexity is $O(n^2)$ but it tends to be like O(n) as the list gets sorted as time passes. Gnome sort makes every swap on it's original array so it's space complexity is O(n).

Stability

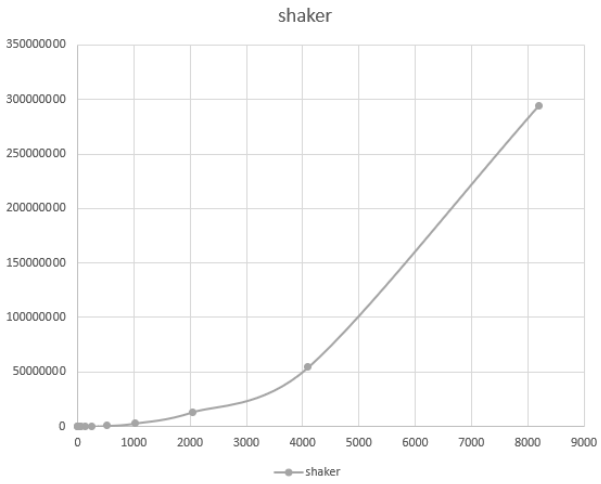
This algorithm compares and swaps only adjacent elements in the array so it preserves the original order of the values of same value. This makes gnome sort a stable sorting algorithm.

```
Gnome sort stability analysis:
2d 1a 2e 1b 3f 1c 4h 3g 5j 3k 5l 1m 3x 6q 7t 1u
1a 1b 1c 1m 1u 2d 2e 3f 3g 3k 3x 4h 5j 5l 6q 7t
Gnome sort is stable
```

3. Shaker Sort



Average



Worst

Analysis

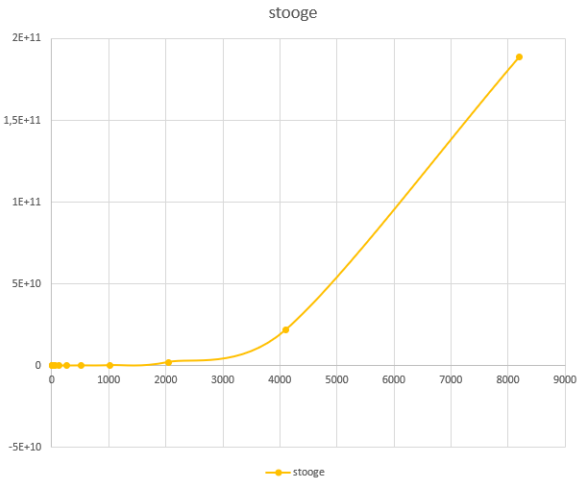
There are two nested for loops in this algorithm and this makes it's time complexity $O(n^2)$. The average case and worst case graphs show this situation since the algorithm's average and worst case complexities are same. Space complexity of the Shaker Sort is O(1).

Stability

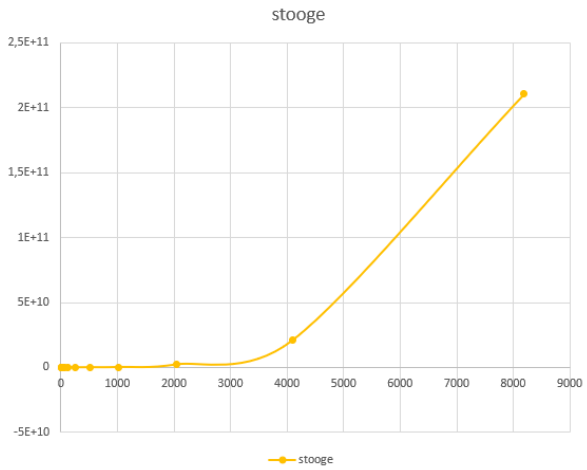
Like with the Gnome Sort, the situation is the same with this sort algorithm. Swapping only occurs between adjacent elements in the array. This makes the algorithm stable.

```
Shaker sort stability analysis:
2d 1a 2e 1b 3f 1c 4h 3g 5j 3k 5l 1m 3x 6q 7t 1u
1a 1b 1c 1m 1u 2d 2e 3f 3g 3k 3x 4h 5j 5l 6q 7t
Shaker sort is stable
```

4. Stooge Sort



Average



Worst

Analysis

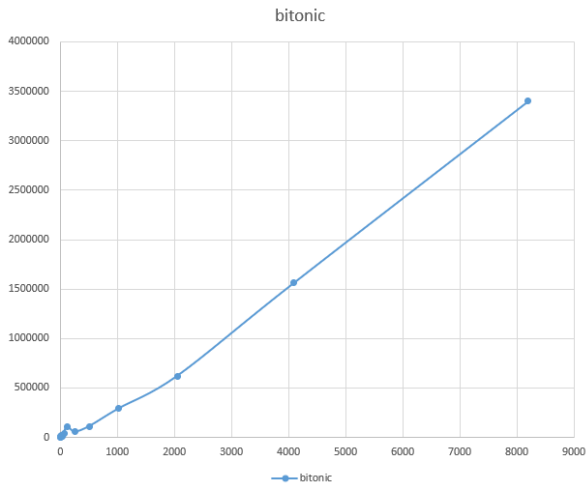
Stooge sort is a recursive sorting algorithm. It is notable for its exceptionally bad time complexity of O(nlog 3 / log 1.5) = O(n2.7095...). The running time of the algorithm is thus slower compared to reasonable sorting algorithms, a canonical example of a fairly inefficient sort. It’s space complexity is O(n).

Stability

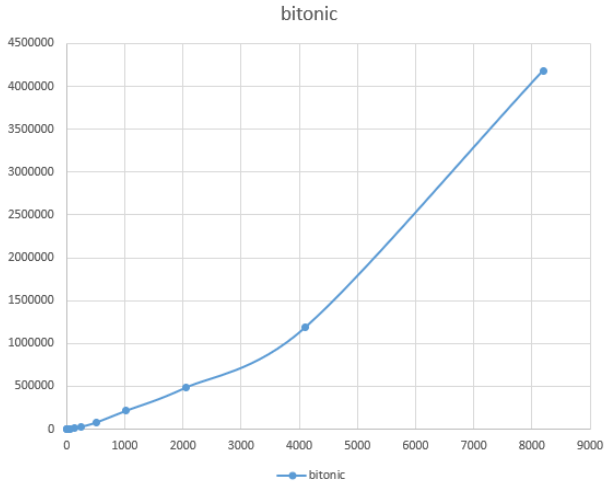
Stooge sort is not a stable as a sorting algorithm because it works in a recursive manner. This makes swapping objects random in a way so the order of the equally valued elements might change.

```
Stooge sort stability analysis:
2d 1a 2e 1b 3f 1c 4h 3g 5j 3k 5l 1m 3x 6q 7t 1u
1u 1a 1b 1c 1m 2e 2d 3g 3k 3x 3f 4h 5l 5j 6q 7t
Stooge sort is unstable
```

5. Bitonic Sort



Average



Worst

Analysis

Bitonic sort is a comparison-based sorting algorithm that can be run in parallel. It focuses on converting a random sequence of numbers into a bitonic sequence, one that monotonically increases, then decreases. It's time complexity is $O(\log^2 n)$. This makes it a lot more efficient than the other sorting algorithms that we worked on this assignment. The $O(\log^2 n)$ curve can be easily seen in the average case graph. It's space complexity is $O(n \log^2 n)$.

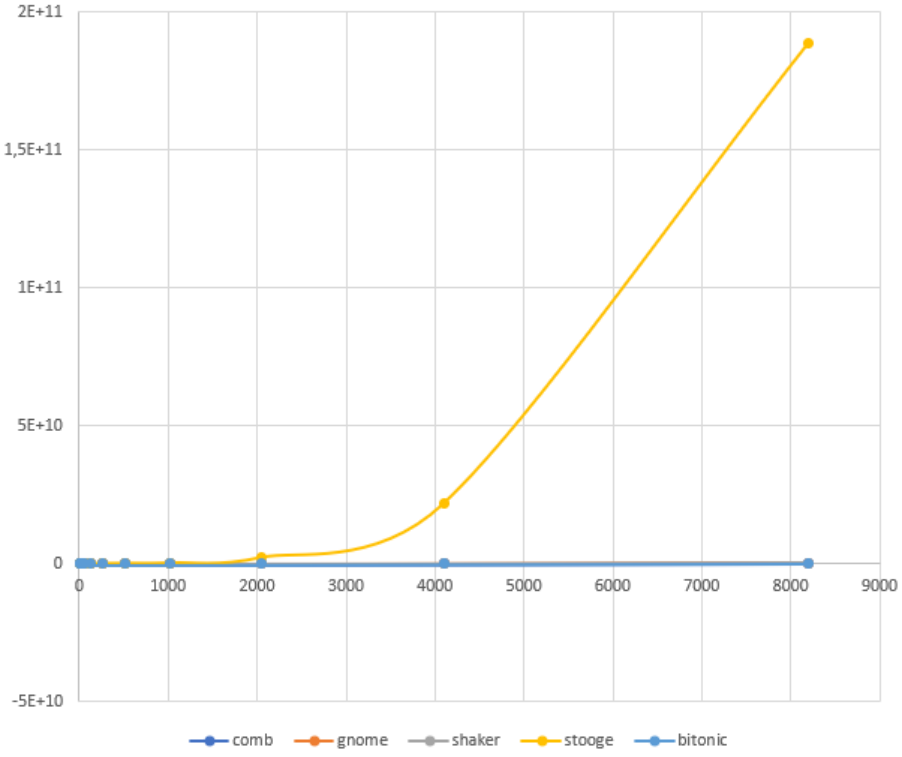
Stability

Bitonic sort divides the array and works recursively. This makes the algorithm unstable.

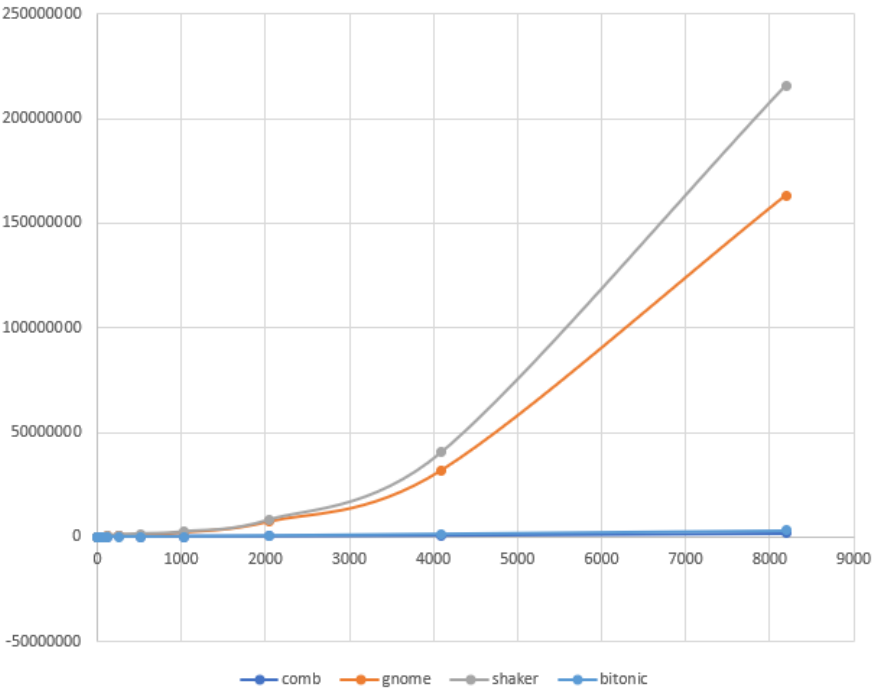
```
Bitonic sort stability analysis:
2d 1a 2e 1b 3f 1c 4h 3g 5j 3k 5l 1m 3x 6q 7t 1u
1a 1b 1c 1u 1m 2e 2d 3g 3x 3k 3f 4h 5l 5j 6q 7t
Bitonic sort is unstable
```

Average-case Time Complexity Table

	AVERAGE CASE												
	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192
comb	1260	2589	7379	8049	19370	55410	138049	366459	288150	301419	333210	762109	1926360
gnome	669	2779	7429	13640	83540	269029	433100	980870	924340	2009840	7260899	31814140	1,63E+08
shaker	870	2810	3710	59250	54240	169799	499169	974850	1488429	2490480	8187749	40505400	2,16E+08
stooge	1200	2150	2970	21950	188740	1318370	990390	8298709	80925500	2,43E+08	2,17E+09	2,17E+10	1,89E+11
bitonic	969	2940	6920	20830	14270	42010	110269	54919	110460	289219	618200	1562639	3401249

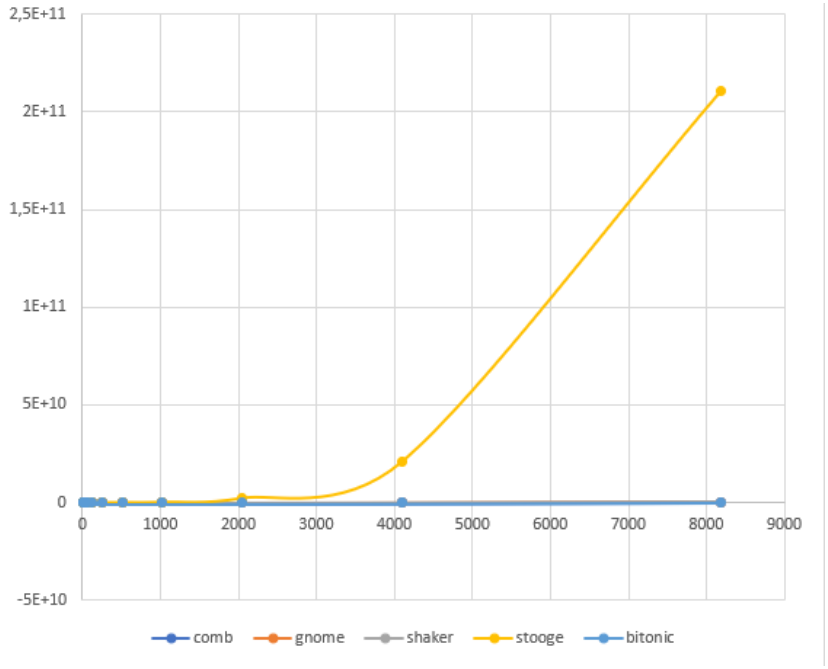


Without Stooge Sort Graph



Worst-case Time Complexity Table

	WORST CASE												
	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192
comb	170	140	200	770	1350	2449	5119	14170	31739	71289	180569	473130	1168659
gnome	120	170	299	1569	6160	14860	53260	235250	927200	3742820	15925430	70371559	3,68E+08
shaker	210	130	300	2700	5469	11490	52840	190229	762129	3065329	13434819	54353190	2,94E+08
stooge	160	130	769	7700	66439	311440	850500	7824900	71452010	2,52E+08	2,27E+09	2,09E+10	2,11E+11
bitonic	119	280	390	1499	2229	4910	13440	32820	82230	218150	487360	1185410	4186190



Without Stooge Sort Graphic