HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

BBM204 SOFTWARE LABORATORY II

2021 SPRING

ASSIGNMENT I

Şevval Atmaca

21827115

March 23, 2021

1. THE IDEA BEHIND THIS ASSIGNMENT

The idea behind this assignment is to find the running times/size graphs of each algorithm in different input sizes and in different situations (such as descending, ascending or random). Then, find the average and worst cases for each algorithm according to these datas. Finally, compare the average cases and worst cases of all 5 algorithms.

2. TIME COMPLEXITIES AND COMPARING OF ALL ALGORITHMS

2.1 AVERAGE CASES COMPLEXITIES

2.1.1 COMPARING AVERAGE CASES OF ALL ALGORITHMS

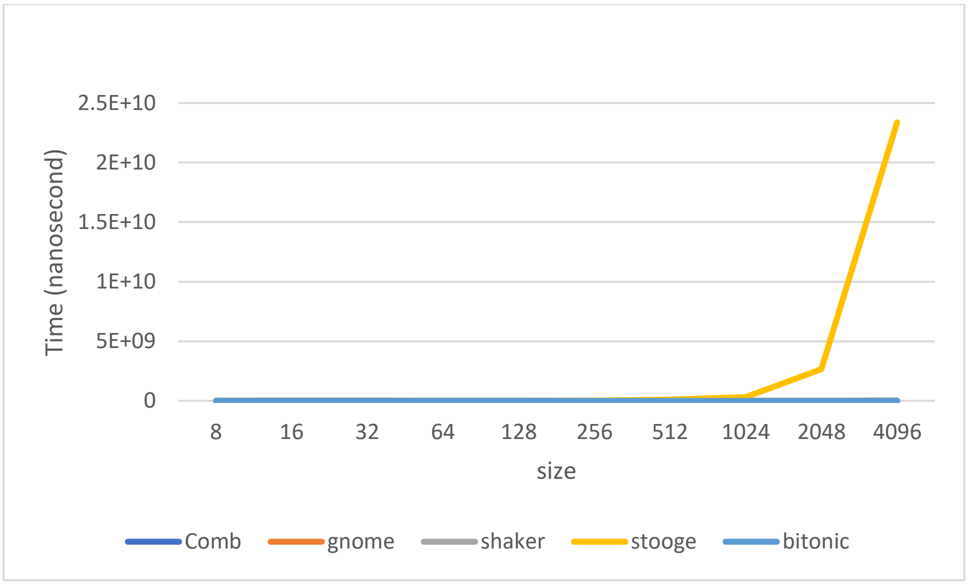

Figure 1

I converted all run time datas in Figure 1 to "***logarithmic scale"*** you to see all of these algorithms together in Figure 2.

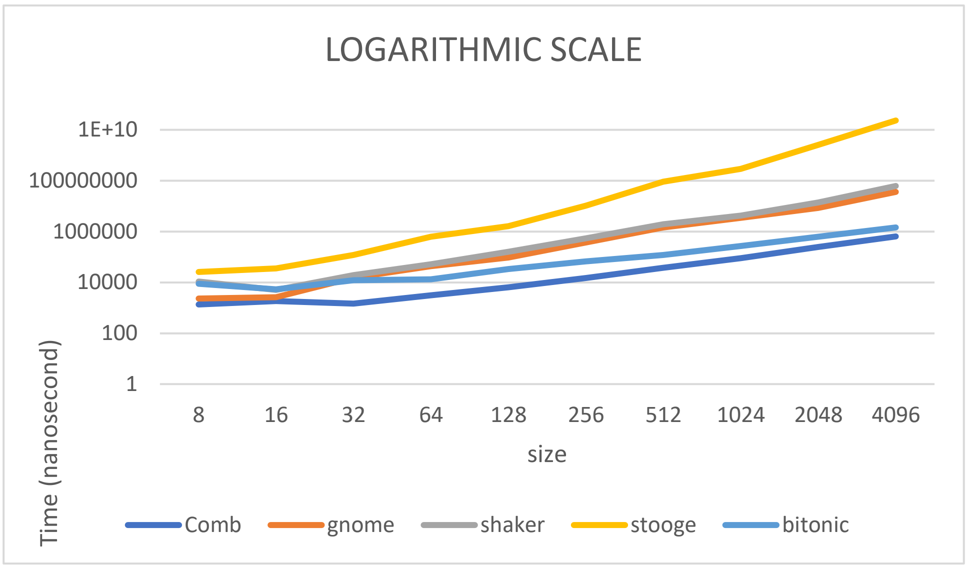

Figure 2

When I compare their speeds according to the graph on the Figure 2:

Stooge < Gnome ~ Shaker < Bitonic < Comb

Then stooge is the slowest and comb is the fastest in of them.

2.1.2 AVERAGE CASES TIME COMPLEXITIES OF ALGORITHMS

To see complexities easily, I design my graphs like that:

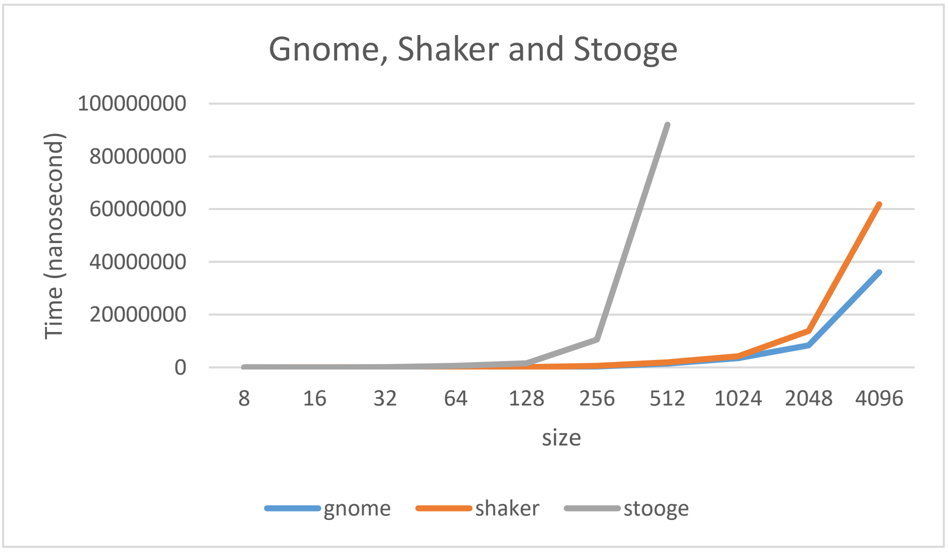

Figure 3

I created the graph in Figure 3 to understand the complexities of Stooge, Gnome and Shaker. And by this, When I examine the graph, Stooge is like O(n^3).Gnome and Shaker are very close and they are **O(n^2).**

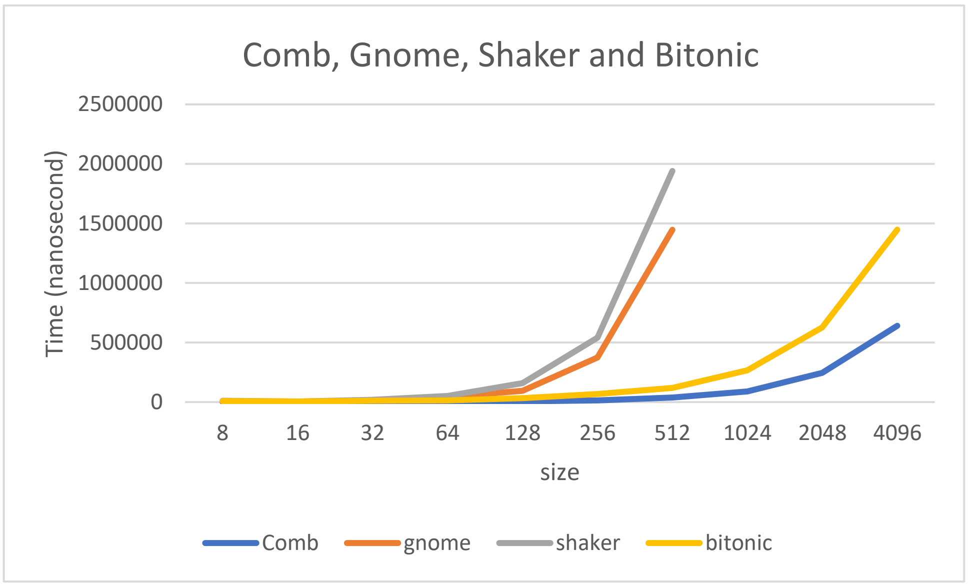

Figure 4

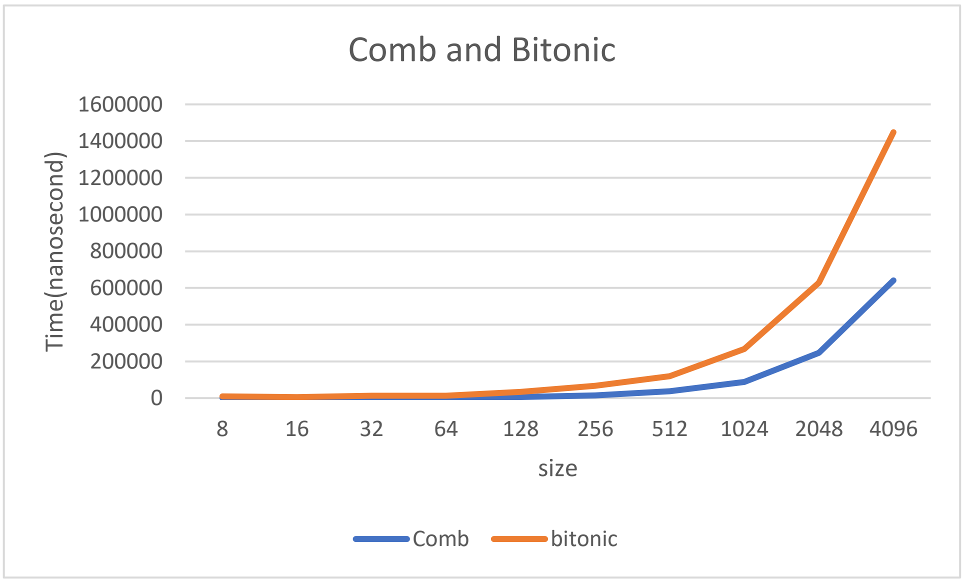

Figure 5

When I compare the comb and the bitonic, Bitonic is similar to **O ((logn) ^ 2)** and the comb is similar to the bitonic according to Figure 4 and Figure 5.

2.2 WORST CASES COMPLEXITIES

2.2.1 COMPARING WORST CASE OF ALL COMPLEXITIES

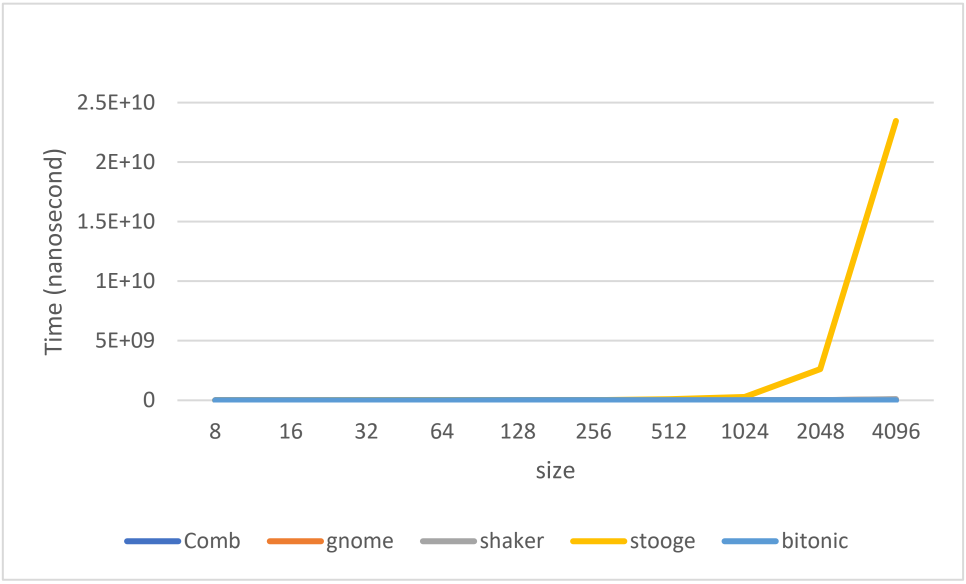

Figure 6

I converted all run time datas in Figure 6 to "***logarithmic scale"*** you to see all of these algorithms together in Figure 7.

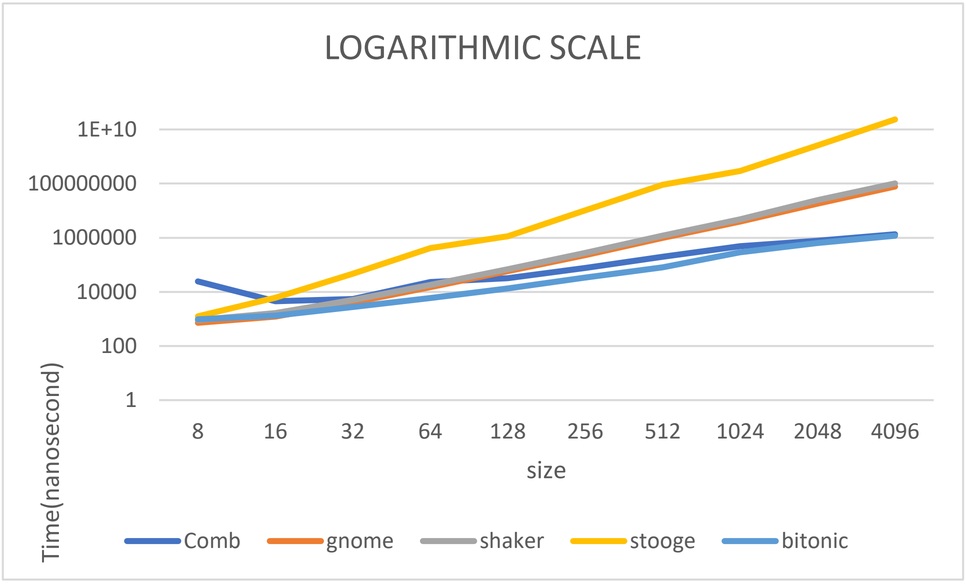

Figure 7

When I compare their speeds according to the graph on the Figure 7:

Stooge < Gnome ~ Shaker < Com < Bitonic

Then stooge is again the slowest and Bitonic is the fastest in of them.

2.2.2 WORST CASES TIME COMPLEXITIES OF ALGORITHMS

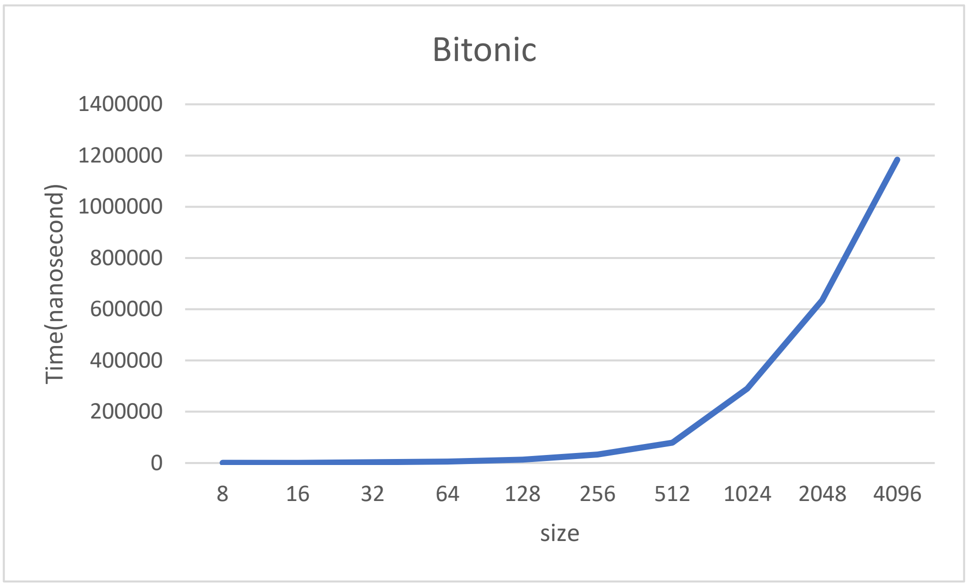

Figure 8

When we examine Figure 8, it is similar to the bitonic O((logn) ^ 2). So time complexity is O (log (n) ^ 2)

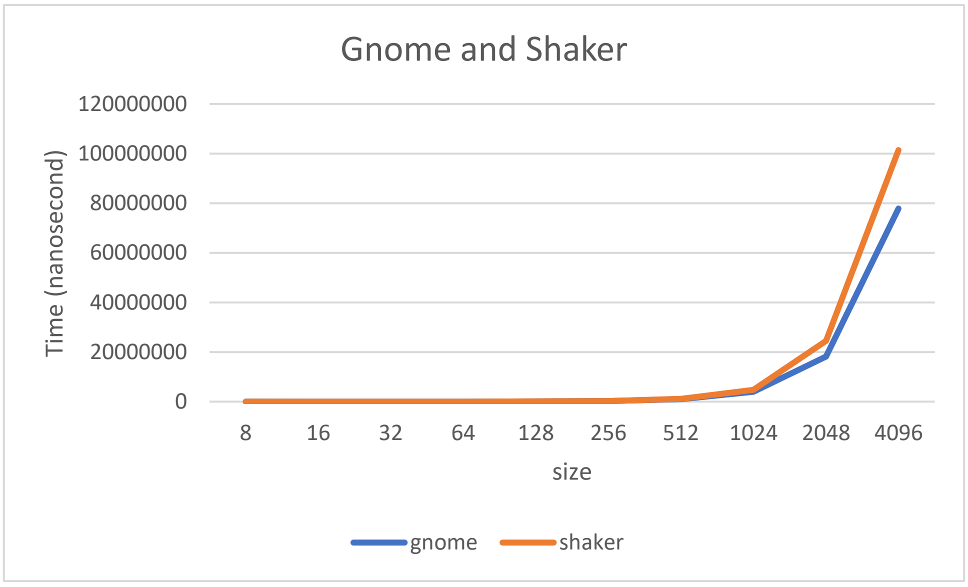

Figure 9

Also, Gnome and Shaker are very close and their lines are similar to O(n ^ 2).

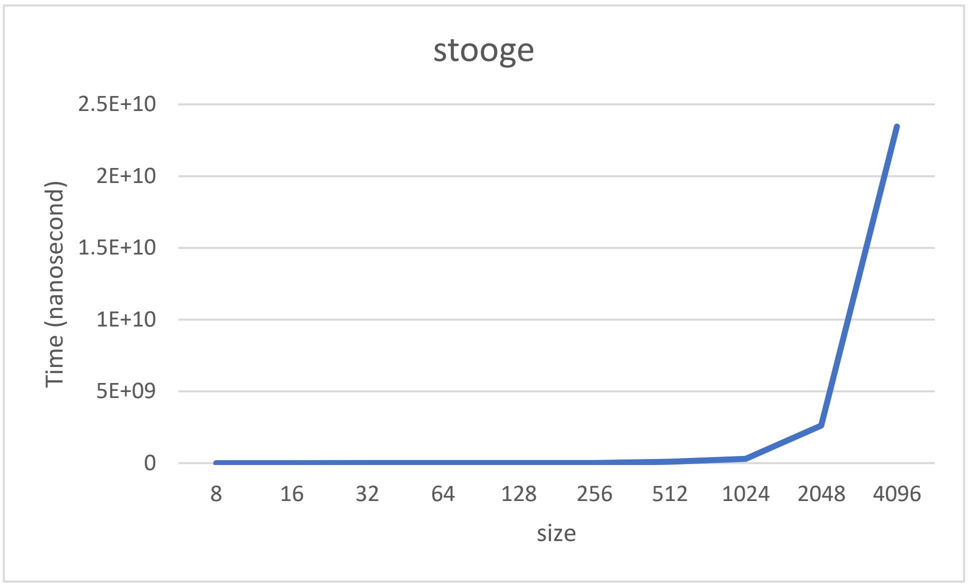

Figure 10

In the Stooge graph in Figure 10, the values have also increased very rapidly and are close to O (n ^ 3).

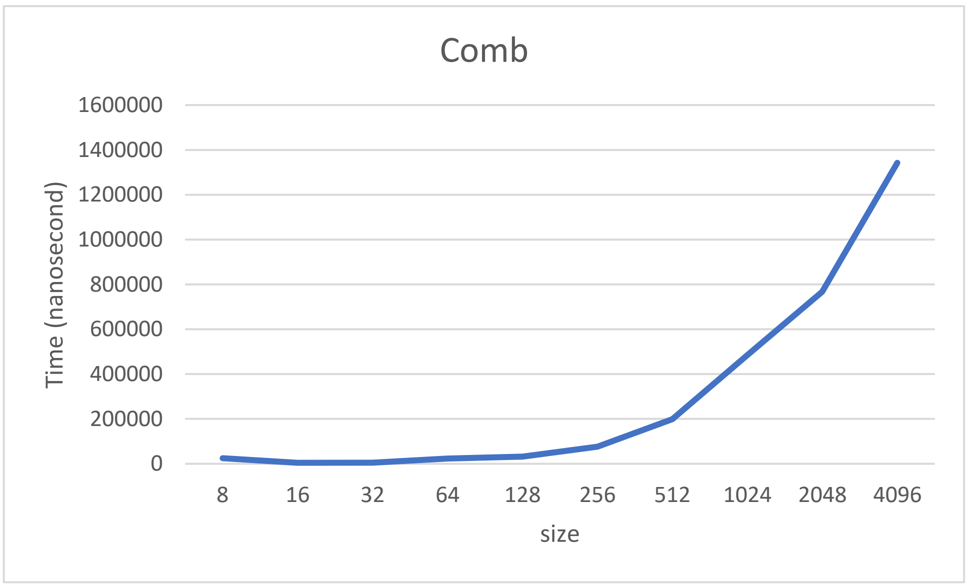

Figure 11

Comb is like O(n^2) in Figure 11.

3. EXPLANATION OF HOW I DESIGNED

In the beginning, I created an ***"Employees"*** class and I created attributes ***("int year*** and ***String name"***) and an ***"arraylist"***. Then, I turned the given pseducodes into Java code and made the sort according to ***"int year"*** attribute.

I used 10 sizes. I saved the sizes to an array ***(int size [] = {8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096}).*** I created 5 arraylists ***(comb_arr, gnome_arr, shaker_arr, stooge_arr, bitonic_arr)*** to record the run times of each algorithms.

Firstly, I used the random order.

1. I have added employees as many as first element in "size" to ***"arraylist".*** Also, while adding these, I put the "String and int attributes" completely randomly.

2. Then, With using the arraylist I created in step 1, I execute "this random arraylist" in all algorithms.

3. Then I applied step 1 and step 2 to you for the other sizes in my **"size array"**. I saved the run times of each algorithm and each size in the random order to their arraylists and printed them console.

Secondly, I used the ascending order.

1. I applied all the steps in random order. But in addition, before run my algorithms, I converted my ***"arraylist"*** to an ascending order and then Run it.

Thirdly, I used the ascending order.

1. I applied all the steps in random order. But in addition, before run my algorithms, I converted my ***"arraylist"*** to a descending order and then Run it.

And I have run 10 times. Then, I calculated their averages. Finally, I have learned in which situations or order these algorithms are worst, best or average.

4. STABILITY

For stability firstly, I created an ***"Employees"*** class and I created attributes ***("int year and String name")*** and an ***"arraylist".***

And I added 8 people to this arraylist with their int attributes and String. While adding these people, I gave some of them the same number of their int attribute. And I have sorted this arraylist by string attribute.

Then I run this arraylist on all 5 algorithms. I realized that if they paid attention the arraylist which I sorted by the string attribute, while they are sorted, they are stable and otherwise not.

According to these informations, Gnome and Shaker are stable. Comb, Bitonic and Stooge are not stable.

ADVANTAGES AND DİSADVANTAGES OF STABILITY

Disadvantages:

1. I examined that Generally, stable algorithm is slower than unstable algorithm.For example, Gnome and Shaker is slower than comb and bitonic.
2. Stability use extra space because it keep the array before sorted.

Advantages:

1. Stability can be useful if You care the order of other attributes of an array.
2. Order is always same in same inputs for all stable algorithms. But Unstable algorithms are not. They may be change.

,5. FINDING THE AVERAGE AND WORST CASES OF ALL ALGORİTHMS

5.1 COMB SORT

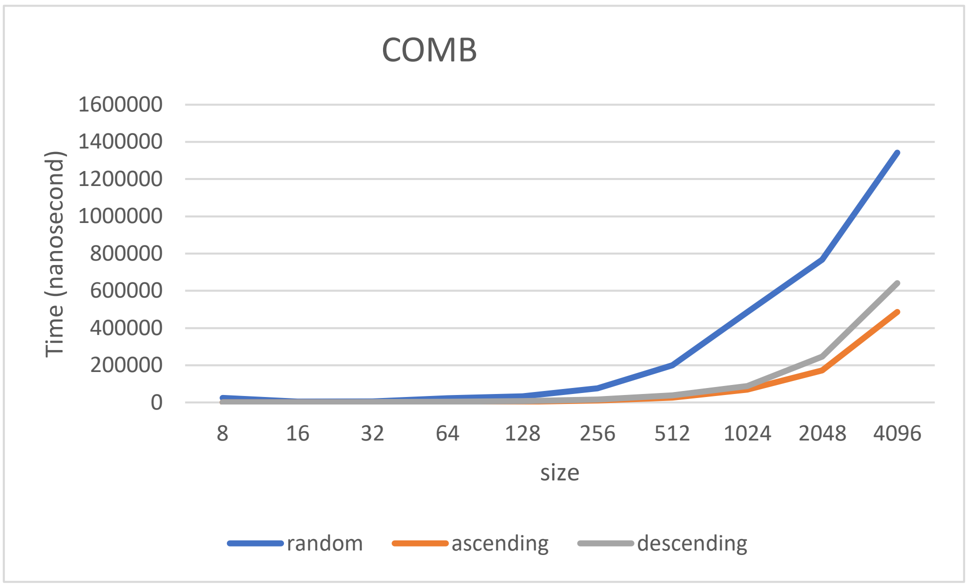

Figure 12

When I compare their speeds according to the graph on the Figure 12:

random < descending < ascending

The slowest is random order and the fastest is ascending order for Comb sort. Average case is descending order and worst case is random order. Best case is ascending for Comb.

5.2 GNOME SORT

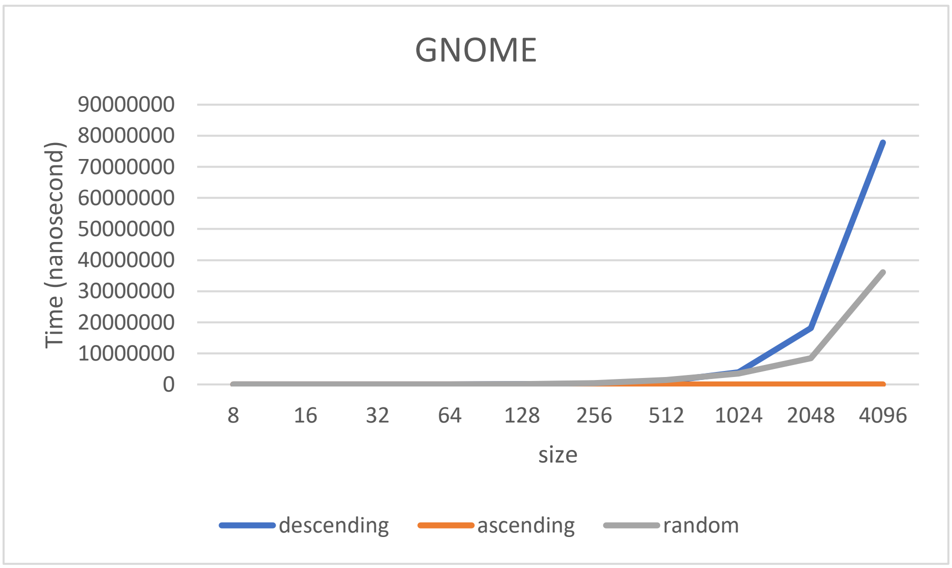

Figure 13

When I compare their speeds according to the graph on the Figure 13:

Random ~ descending <ascending

The slowest is random and descending order and the fastest is ascending order for Gnome sort. Average and worst case are equal and they are random and descending. Best case is ascending for Gnome.

5.3 SHAKER SORT

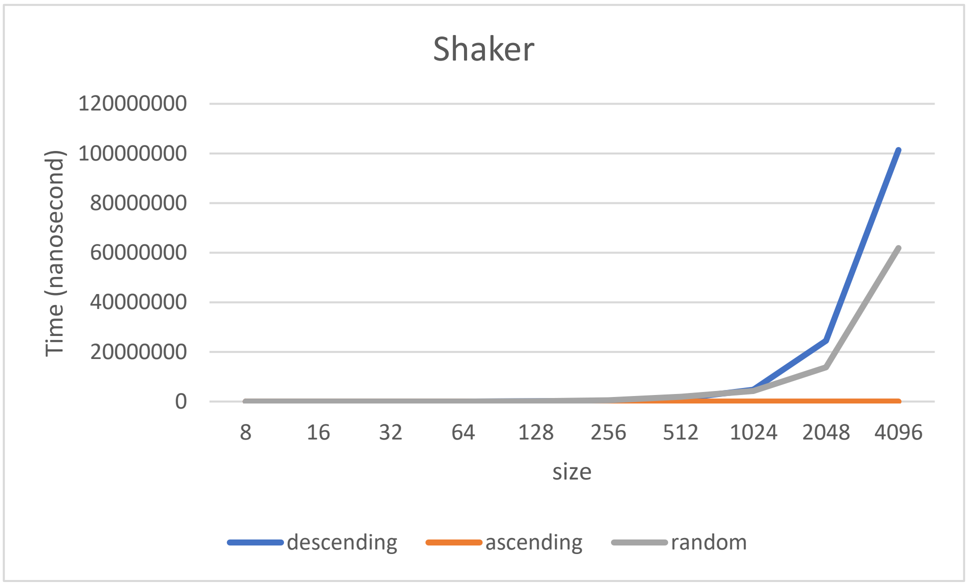

Figure 14

When I compare their speeds according to the graph on the Figure 14:

Random ~ descending <ascending

The slowest is random and descending order and the fastest is ascending order for Shaker sort. Average and worst case are equal and they are random and descending. Best case is ascending for Shaker.

5.4 STOOGE SORT

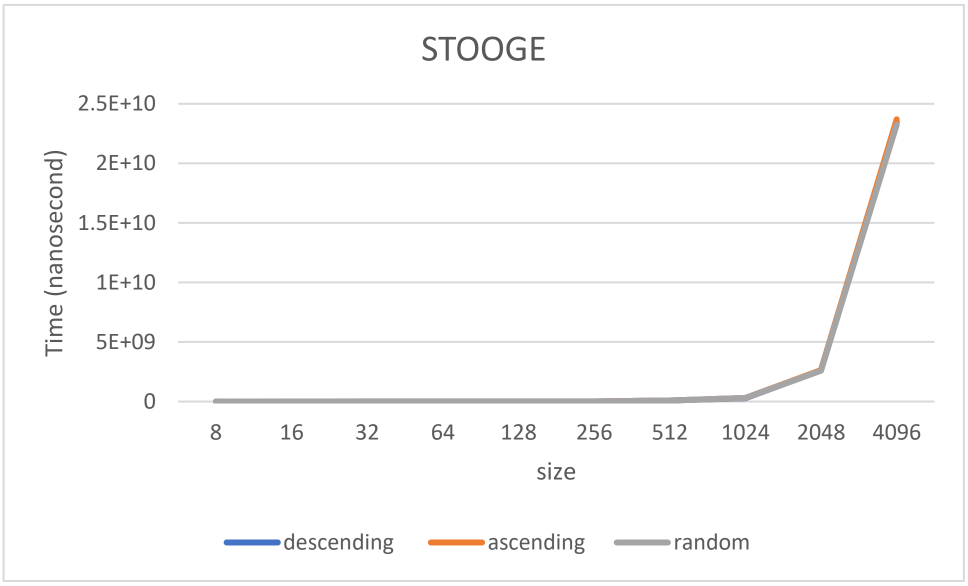

Figure 15

When I compare their speeds according to the graph on the Figure 15:

Random = descending = ascending

All orders has the same speed for Stooge. Best, worst and average case are equal for Stooge.

5.5 BITONIC

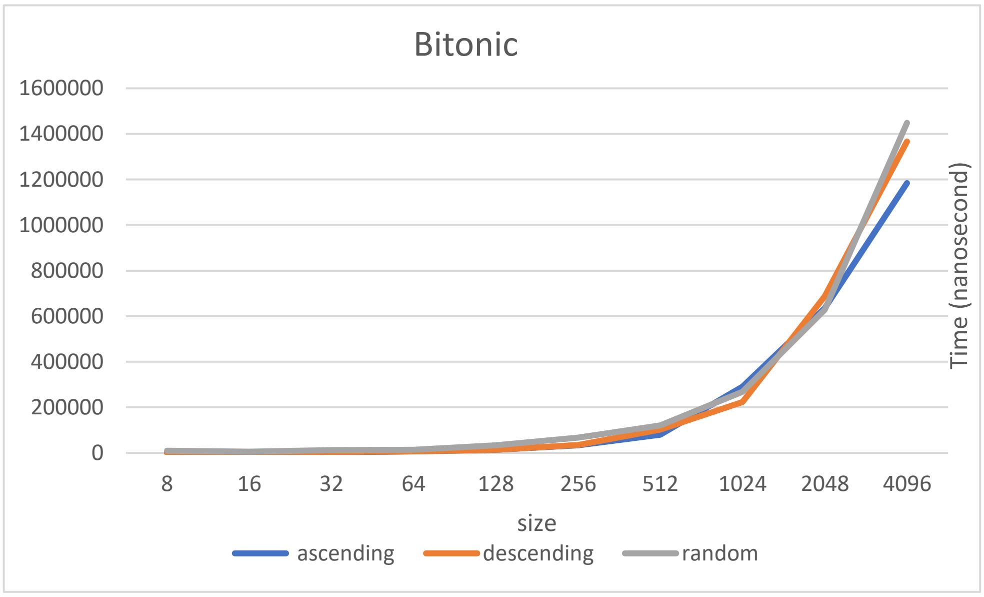

Figure 16

When I compare their speeds according to the graph on the Figure 16:

Random = descending = ascending

All orders has the same speed for Bitonic. Best, worst and average case are equal for Bitonic.

6. MEMORY REQUIREMENTS

When I examine these 5 algorithms, they are all in place algorithm. Because when sorting elements or swap does not create or use extra arrays. In place sorting algorithms also do not use extra space.

7. EXECUTION TIME

Average Cases

	Comb	gnome	shaker	stooge	Bitonic
8	1356	2320	10810	25740	8720
16	1864	2630	5070	35750	5340
32	1450	15450	19210	119190	12050
64	3070	43500	52160	633050	13270
128	6500	95790	158530	1604270	33780
256	14860	375780	541290	10527540	67230
512	38010	1447230	1941130	92052840	119860
1024	88480	3447910	4244740	292915390	267840
2048	246790	8399310	13866400	2582841420	628050
4096	641090	36093590	61866190	23248358090	1448210

Worst Cases

	Comb	gnome	shaker	stooge	Bitonic
8	24520	720	910	1250	970
16	4500	1210	1640	6070	1320
32	5390	4030	4960	46330	2730
64	22800	14920	17980	414220	5930
128	32150	58390	69280	1132590	13330
256	76590	229460	272150	10148200	33570
512	199090	988430	1188500	90868360	79670
1024	484750	3914490	4741420	290357220	290640
2048	767530	18143490	24537590	2611597660	635260
4096	1342310	77827370	1.01E+08	23455559070	1183850