Hacettepe University

Computer Engineering Department

BM204 Software Laboratory II - 2021 Spring

ASSIGNMENT 1

March 25, 2021

Student name:
Muhammed Enes Yavuz

Student Number:
b21989712

1 Problem Definition

In this assignment our goal analyze some sorting algorithms based on time complexity then compare them.Also we should show their time complexities on graph representation.For drawing graphs we should run them with random input also with worst case input.I used powers of two in this assignment.I used a variable named count in this assignment. I used 3,4,5,6,7,8,9,10,11,12 as input so my input sizes are 8,16,32,64,128,256,512,1024,2048,4096. I share data table for average case and for average case under:

Count	Comb Sort	Gnome Sort	Shaker Sort	Stooge Sort	Bitonic Sort
8	4695,2	868	911,2	1316	1504,2
16	7341,2	2010	2646,6	5852	2670,4
32	13433,6	6033,8	6649,6	33248,6	6544,8
64	28542,2	22668,4	21640	281334,2	16745,2
128	59083,4	164946,6	79195,6	843815,8	43131
256	137809,8	225966,8	325297,4	6846483,6	134278,4
512	317379	435459	1121918,2	66827370,2	268660,2
1024	770889,4	1276705,2	4690694	194058795	462092,6
2048	1312920,4	4824853,6	9997900,2	1568512758	865191,4
4096	1997151,2	19570994,6	21554799	13271434669	1890741,8

Figure 1: Data for Average Case

Count	Comb Sort	Gnome Sort	Shaker Sort	Stooge Sort	Bitonic Sort
8	7125	1093	1836	1266	1687
16	12080	3038	2973	4823	2633
32	20091	10545	8564	30795	6518
64	63308	41429	32160	252741	18467
128	74915	149812	231269	765200	54344
256	149740	254042	558467	6730016	191012
512	329398	826670	2662961	63956054	322225
1024	1144293	2695631	10522321	181443738	490699
2048	2133349	12582540	19572182	1520875440	953681
4096	3276752	40244037	31531745	13002132780	2123947

Figure 2: Data for Worst Case

2 Comb Sort Algorithm

Our first sort is comb sort.In this sorting algorithm our first gap size is input arrays length and each step we are reducing this gap size by e special value based on experiments by computer scientists.This special value is 1.3.So in this algorithm if we have array size with 10 in second iteration our gap size is 7.In third iteration our gap size is 5. Based on gap size we are compare elements and if left is smaller then right one swap them. We

continue this procedure until gap size is 1.Also I do 5 test cases and measure time with System.nanoTime() method.I noted them and draw a graph with this data.Based on graph and implementation of code I think average time complexity of comb sort is O(n**2).Graph can be seen under:

Figure 3: Comb Sort Average Case

Also we can talk about worst time complexity.At worst time comb sort complexity is O(n**2).For this situation inputs are so complex.Because of that I do different experiment for this type of algorithm.In this algorithm worst case is in maximum swap operations.So I count swap operations with for loop and maximize it and note each time for maximum comparison.After this procedure my graph is look like this:

Figure 4: Comb Sort Worst Case

3 Gnome Sort Algorithm

For this algorithm we have index value and at start point it is 0.We use this value for select index of our value.And for compare we are always using one index back. If our value is 0 we increment it one.So after first iteration it is 1.Than we compare array[0] element with array[1].If they are correct order increment our value else swap them and decrement our value. Do this procedure until our value is equal to input array size.After this steps our array is sorted now.Based on graph and implementation of code I think average time complexity of gnome sort is O(n**2) .My graph for this sort algorithm can be seen under:

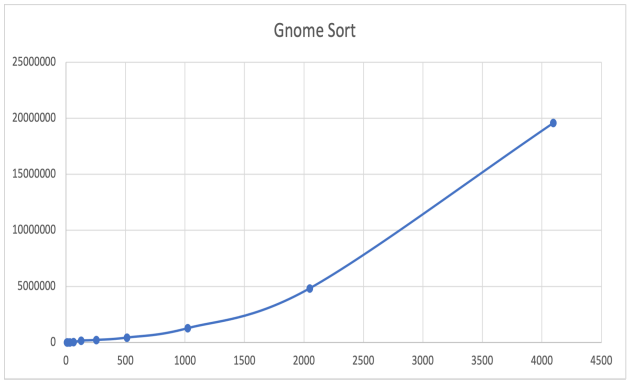


Figure 5: Gnome Sort Average Case

If our values is decreasing order we must go one step back for each step. I think worst case is in this situation.But based on my graph and implementation of code I think worst time complexity of gnome sort is also O(n**2).My graph and data for this sort algorithm can be seen under:

Figure 6: Gnome Sort Worst Case

4 Shaker Sort Algorithm

This type of sorting first we compare adjacent elements from left to right.If left one is smaller then right one we do nothing but if left element is bigger than right swap them.We do this process until reach end of the array.Then we do same thing but reverse order from right to left.Do this two steps until array is sorted.Based on graph and implementation of code I think average time complexity of gnome sort is O(n**2) .My graph for this sort algorithm can be seen under:

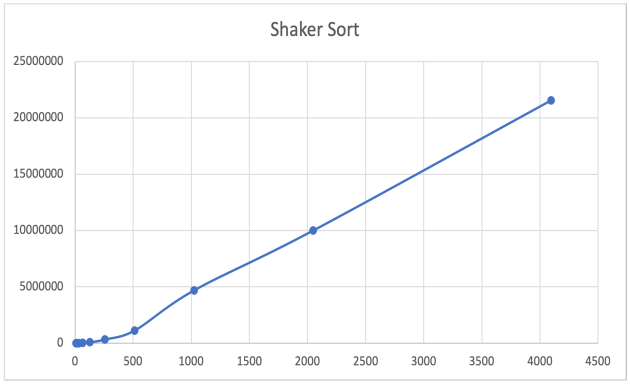


Figure 7: Shaker Sort Average Case

For this algorithm I think worst case is an array with descending order.But complexity is same as average case O(n**2) .My graph and data for this sort algorithm can be seen under:

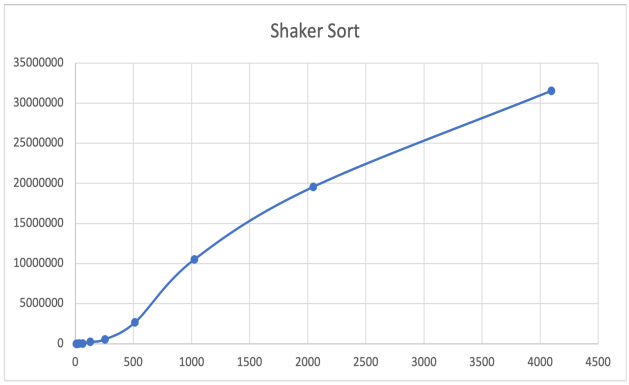


Figure 8: Shaker Sort Worst Case

5 Stooge Sort Algorithm

For this algorithm if the leftmost element is larger than the rightmost element swap them. After that recursively sort first 2/3 of the array then recursively sort last 2/3 of the array and for the finally sort first 2/3 of the array again.Based on graph and implementation of code I think average time complexity of stooge sort is O(n**2.7).My graph for this sort algorithm can be seen under:

Figure 9: Stooge Sort Average Case

For worst case I think everything is same with average case.Because in every situation stooge sort should do same think.So I think worst time complexity of stooge sort is O(n**2.7) still.My graph for this sort algorithm can be seen under:

Figure 10: Stooge Sort Worst Case

6 Bitonic Sort Algorithm

This is very complex algorithm. In this algorithm we must have 2**n size array.First we must bitonic sequence from mixed array.Bitonic sequence meaning is first half is ascending and second half is descending order.To achieve this type of array we do some operations. At the start we pair two of each adjacent values and our first pair is must be ascending order our second pair is descending order third one is again ascending order fourth one is descending and so on.If this style can't be satisfied basically we swap them.Do this process with not adjacent but gap with 2 after this step and after that with gap 4 and so on. After that our first part is ascending order and second part is descending order.After that we should swap them by length of input divided by 2. We continue this process until our gap is 1. After that whole process our array is sorted now. Based on graph and implementation of code I think average time complexity of bitonic sort is O(log(n)**2).My graph for this sort algorithm can be seen under:

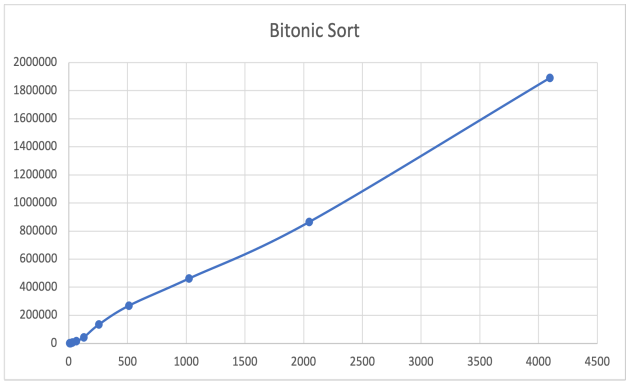


Figure 11: Bitonic Sort Average Case

For worst case I think everything is same with average case.Because in every situation stooge sort should do same think.So I think worst time complexity of stooge sort is O(log(n)**2) still .My graph for this algorithm with worst case can be seen under:

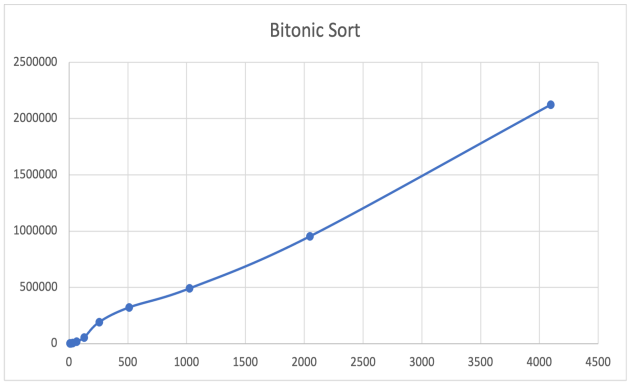


Figure 12: Bitonic Sort Worst Case

7 Stability Of Algorithms

For stability of algorithms I do sample test.For this test I use my Item class.Item class has two attributes first was number and second one is name.Both of them are integers. Number chosen randomly but name chosen with ascending order.I use random method with range 10 and sort them with each algorithm. After sorting, for every same element if my first ascending order is broked this algorithm is not stable else this algorithm is stable.I share results under:

At this picture first values are name and randomly chosen.Second ones are ascending order.For value 1 ascending order same in gnome sort and shaker sort.But in the other algorithms ascending order is broked.Same thing for value 2.So I can say gnome sort and shaker sort is stable bu others are not stable.

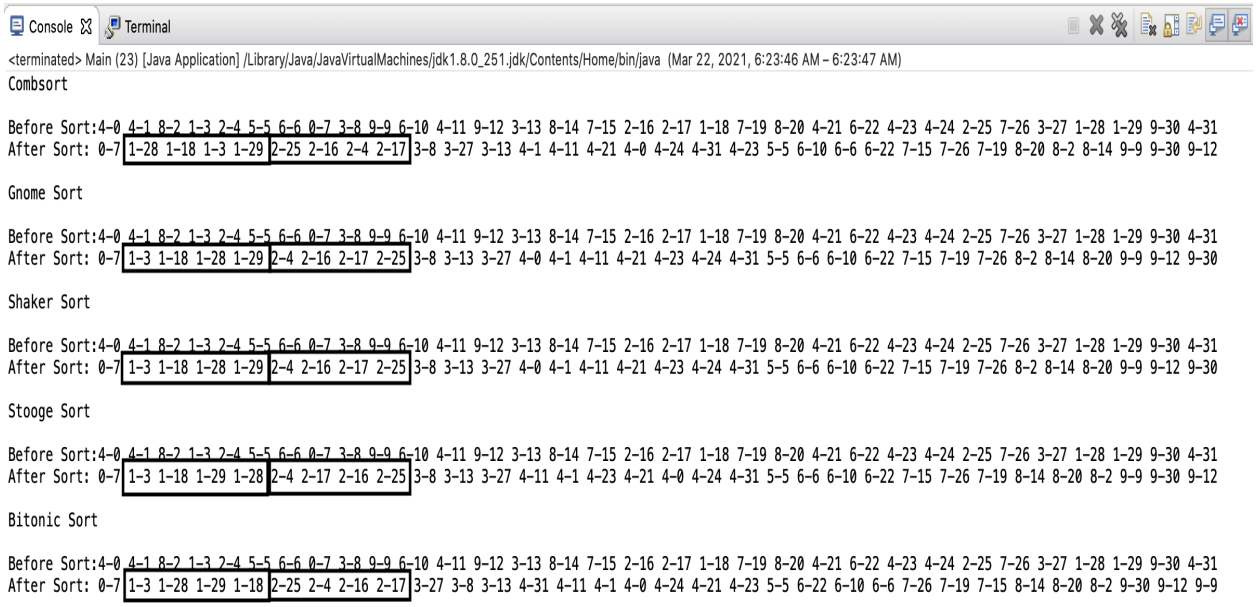
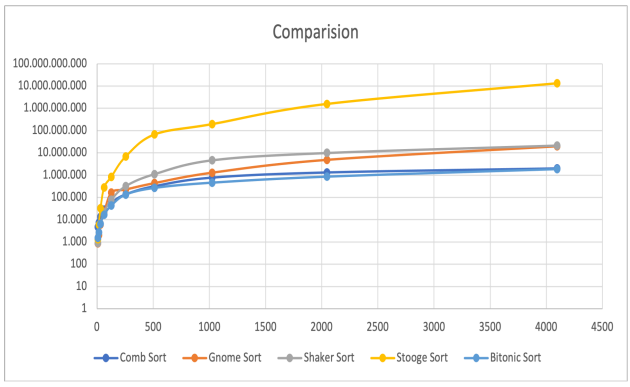


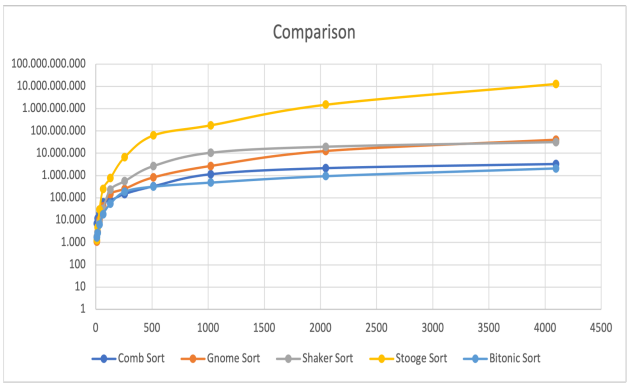
Figure 13: Stability Of Each Algorithm

8 Comparison Of Algorithms

For compare the algorithms I take log of each value based 10.Because stooge sort is so much slower than other algorithms.I draw two graphs. One for average case and the other one is for worst case.I share them under :



(a) Average Case



(b) Worst Case