Hacettepe University

BBM204 Algorithms Assignment-1

Mustafa KOLLU 21627485

Cocktail Sort Algorithms

Cocktail sort, also known as bidirectional bubble sort, cocktail sort, shaker sort (which can also refer to a variant of selection sort), ripple sort, shuffle sort, or shuttle sort, is an extension of bubble sort. The algorithm extends bubble sort by operating in two directions. While it improves on bubble sort by more quickly moving items to the beginning of the list, it provides only marginal performance improvements. Like most variants of bubble sort, cocktail shaker sort is used primarily as an educational tool. More performant algorithms such as timsort, or merge sort are used by the sorting libraries built into popular programming languages such as Python and Java.

Complexity

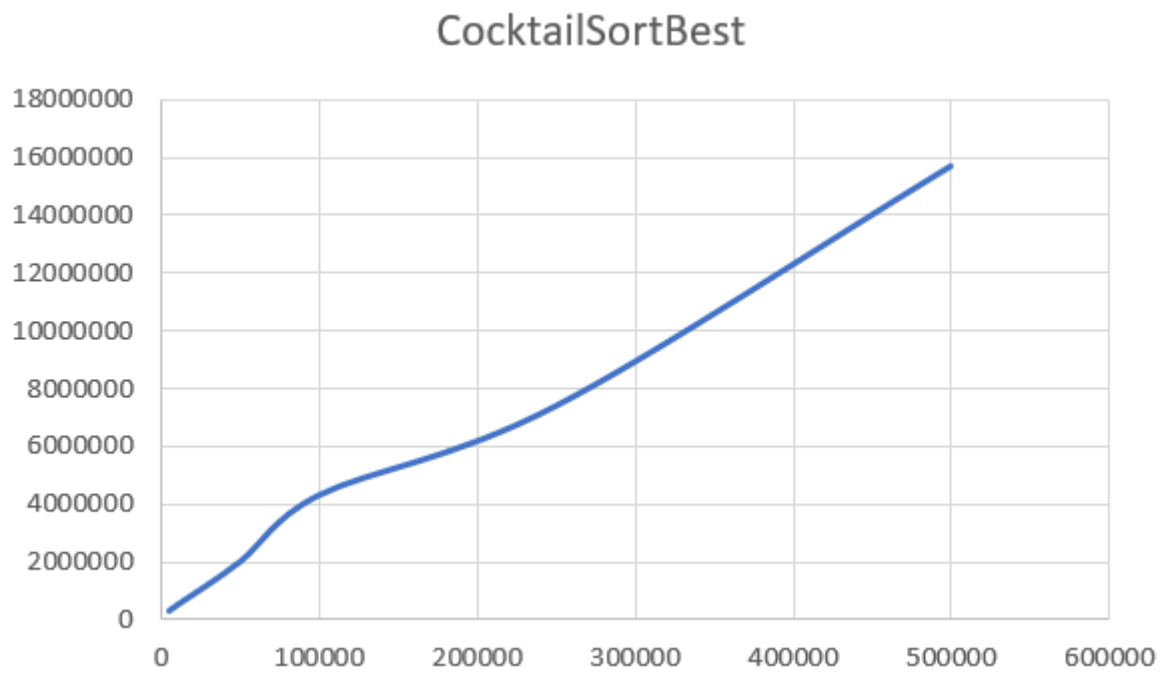
The complexity of the cocktail shaker sort in big O notation O(n^2) is for both the worst case and the average case, but it becomes closer to if the list is mostly ordered before applying the sorting algorithm. For example, if every element is at a position that differs by at most k (k ≥ 1) from the position it is going to end up in, the complexity of cocktail sort becomes O(kn).

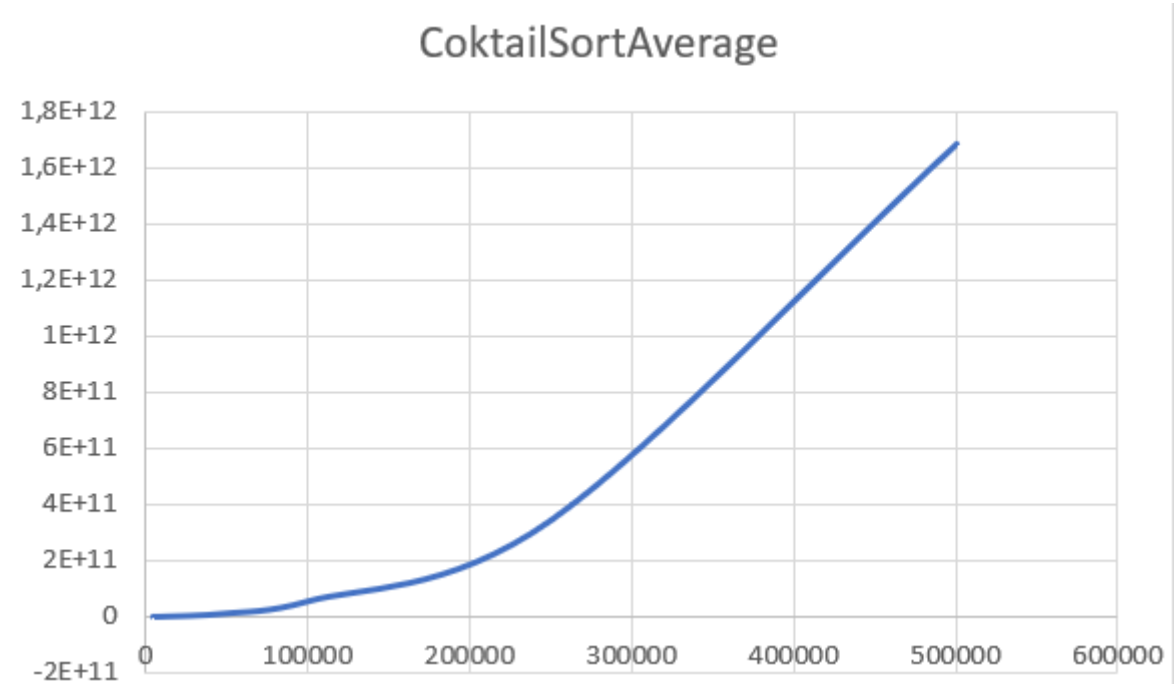
Time			Space
Worst case	Best case	Average case	Worst case
$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$ auxiliary

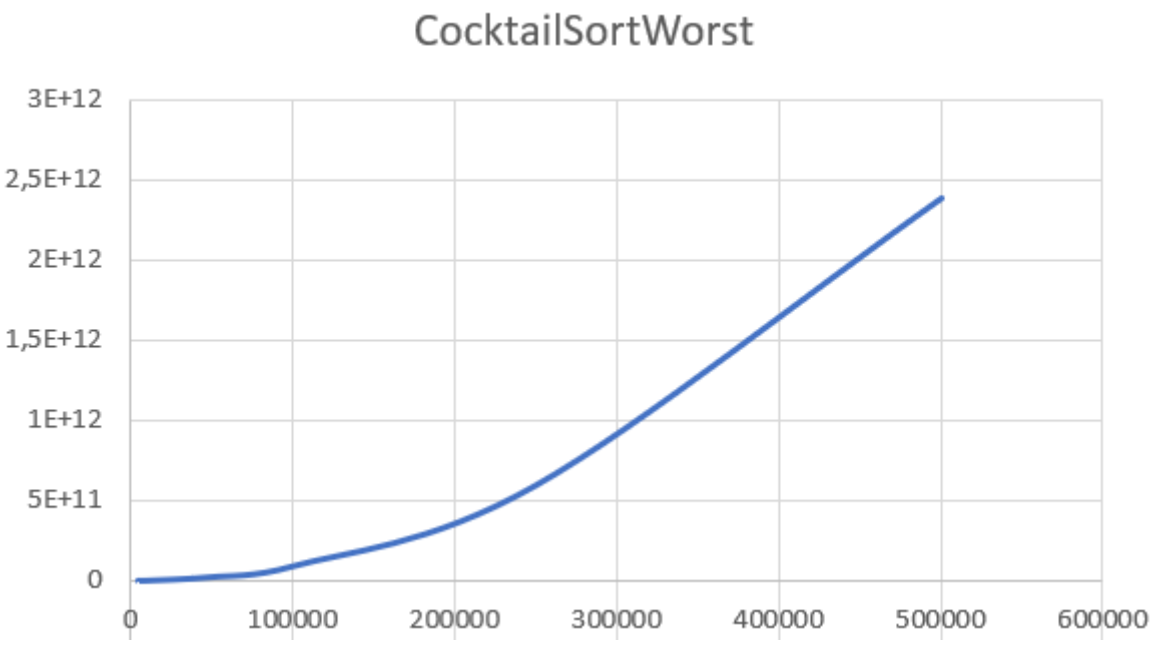
Cocktail sort execution time (Nano second) in Java

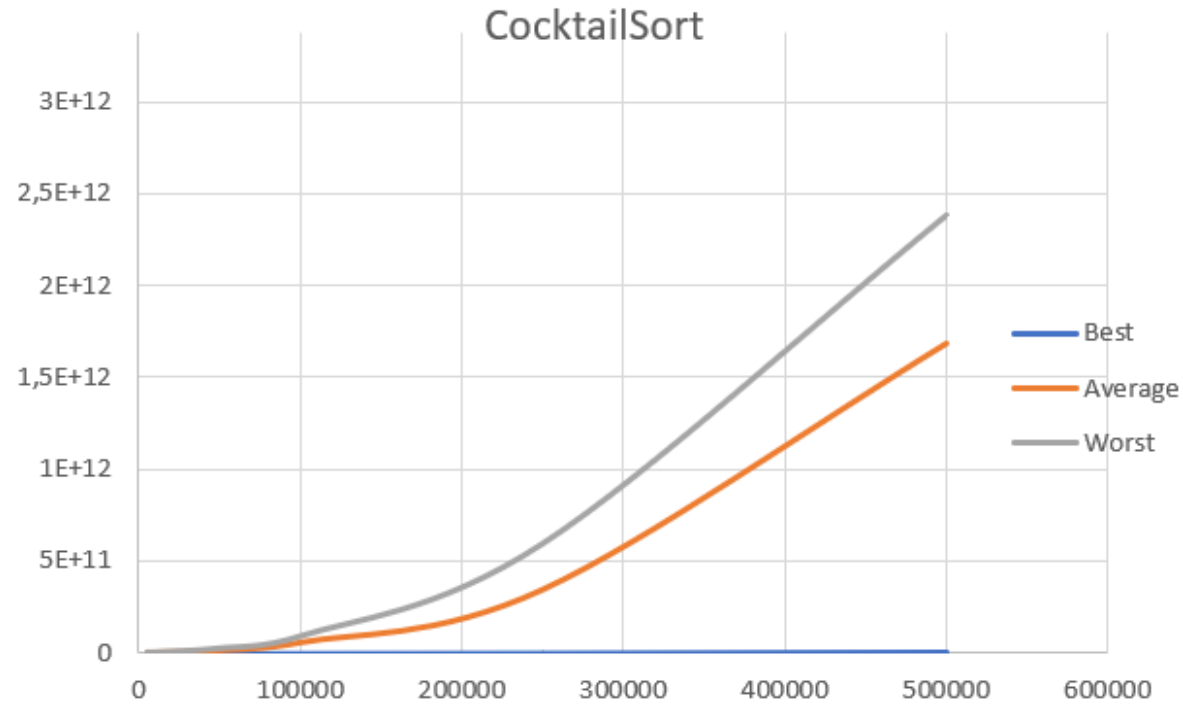
Best	Average	Worst	Input(n)
295200	124279800	130737800	5000
493600	388898800	408907700	10000
1999700	11808744000	22490141800	50000
4299800	54507177400	88933430100	100000
7383700	3,43644E+11	5,93142E+11	250000
15677300	1,68658E+12	2,38113E+12	500000

Cocktail Sort Algorithms Graphs






CocktailSortWorst
3E+12
2,5E+12
2E+12
1,5E+12
1E+12
5E+11
0
0
100000
200000
300000
400000
500000
600000


CocktailSort
3E+12
2,5E+12
2E+12
1,5E+12
1E+12
5E+11
0
0
100000
200000
300000
400000
500000
600000
Best
Average
Worst

Pigeonhole Sort Algorithms

Pigeonhole sorting is a sorting algorithm that is suitable for sorting lists of elements where the number of elements (*n*) and the length of the range of possible key values (*N*) are approximately the same. It requires O(*n* + *N*) time. It is similar to counting sort, but differs in that it "moves items twice: once to the bucket array and again to the final destination [whereas] counting sort builds an auxiliary array then uses the array to compute each item's final destination and move the item there.

The pigeonhole algorithm works as follows:

1. Given an array of values to be sorted, set up an auxiliary array of initially empty "pigeonholes," one pigeonhole for each key through the range of the original array.
2. Going over the original array, put each value into the pigeonhole corresponding to its key, such that each pigeonhole eventually contains a list of all values with that key.
3. Iterate over the pigeonhole array in order, and put elements from non-empty pigeonholes back into the original array.

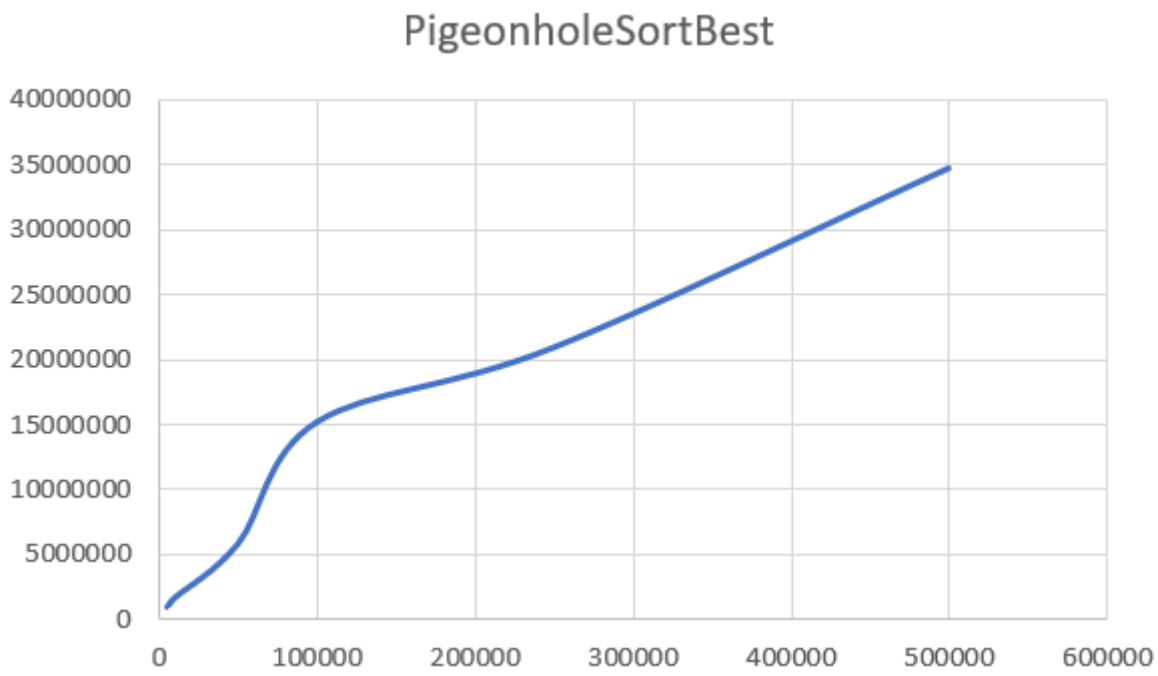
Complexity

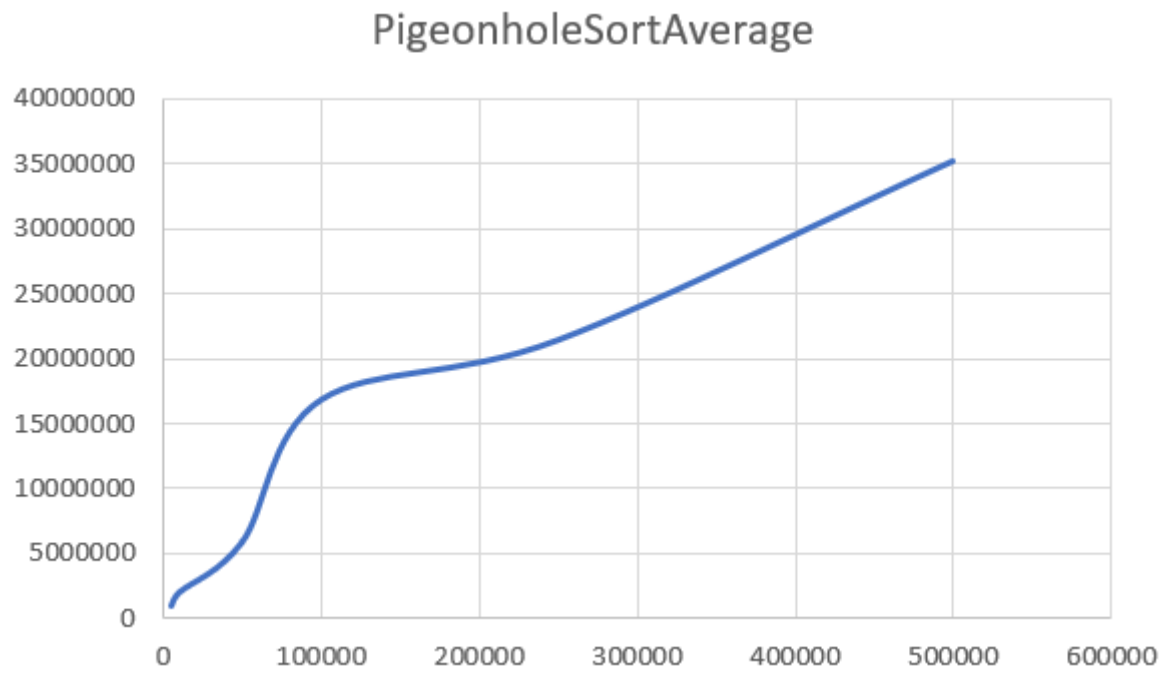
- Worst case time complexity: `θ(n+N)`
- Average case time complexity: `θ(n+N)`
- Best case time complexity: `θ(n+N)`
- Space complexity: `θ(n+N)`

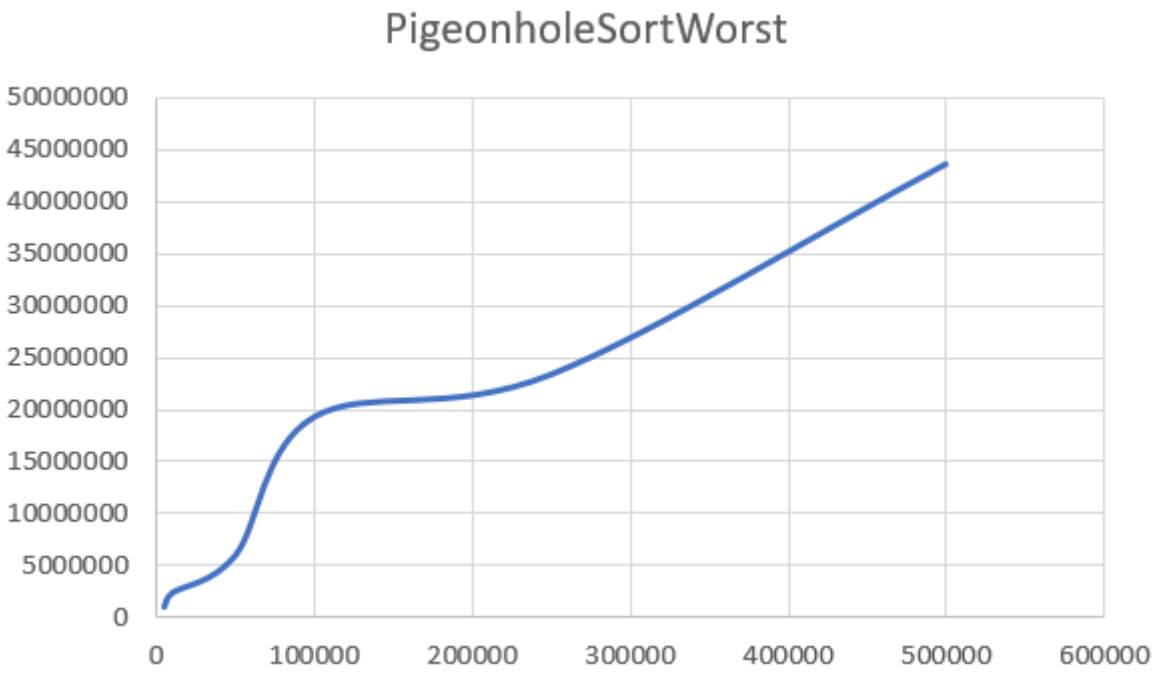
Best	Average	Worst	Input(n)
908600	976100	1117801	5000
1631400	1999699	2439400	10000
5799500	5889000	6040000	50000
15183900	16809600	19340100	100000
20932000	21393800	23413000	250000
34755300	35134300	43590400	500000

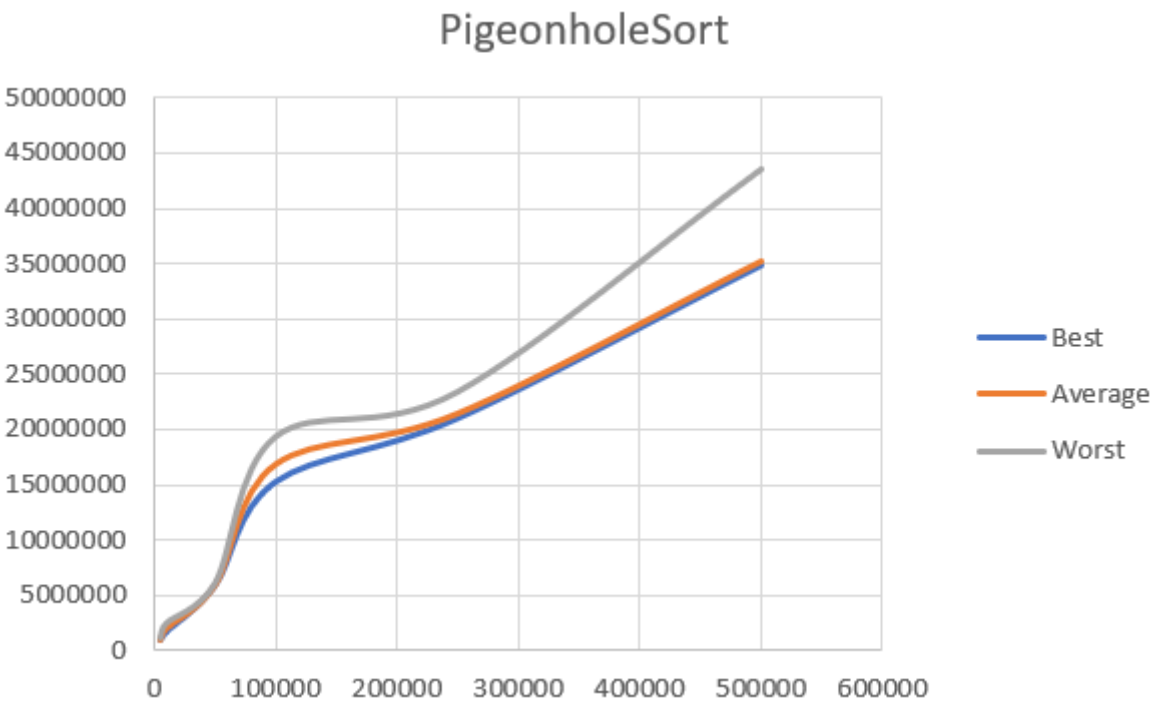
Pigeonhole sort execution time(Nano second) in Java

Pigeonhole Sort Algorithms Graphs






PigeonholeSortWorst
50000000
45000000
40000000
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
0
100000
200000
300000
400000
500000
600000


PigeonholeSort
50000000
45000000
40000000
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
0
100000
200000
300000
400000
500000
600000
Best
Average
Worst

Bitonic Sort Algorithms

Bitonic is a parallel algorithm for sorting. It is also used as a construction method for building a sorting network. The algorithm was devised by Ken Batcher. The resulting sorting networks consist of O(nlog^2(n)) comparators and have a delay of O(log^2(n)), where n is the number of items to be sorted.

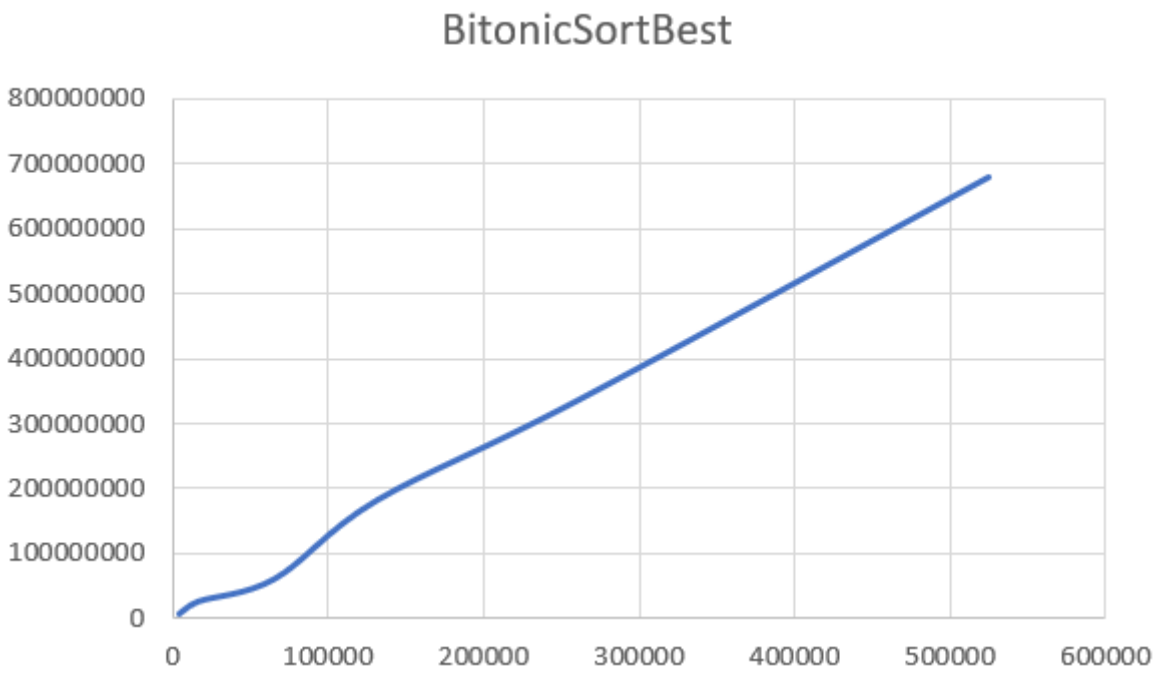
A sorted sequence is a monotonically non-decreasing (or non-increasing) sequence. A *bitonic* sequence is a sequence with for some , or a circular shift of such a sequence.

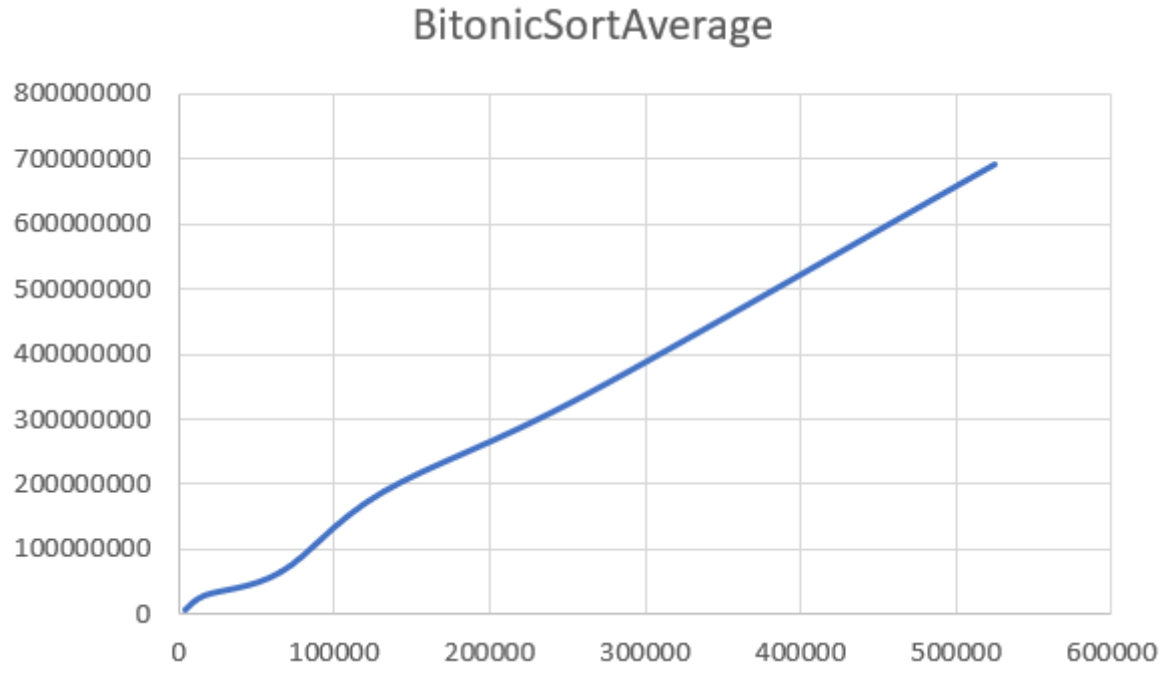
Complexity

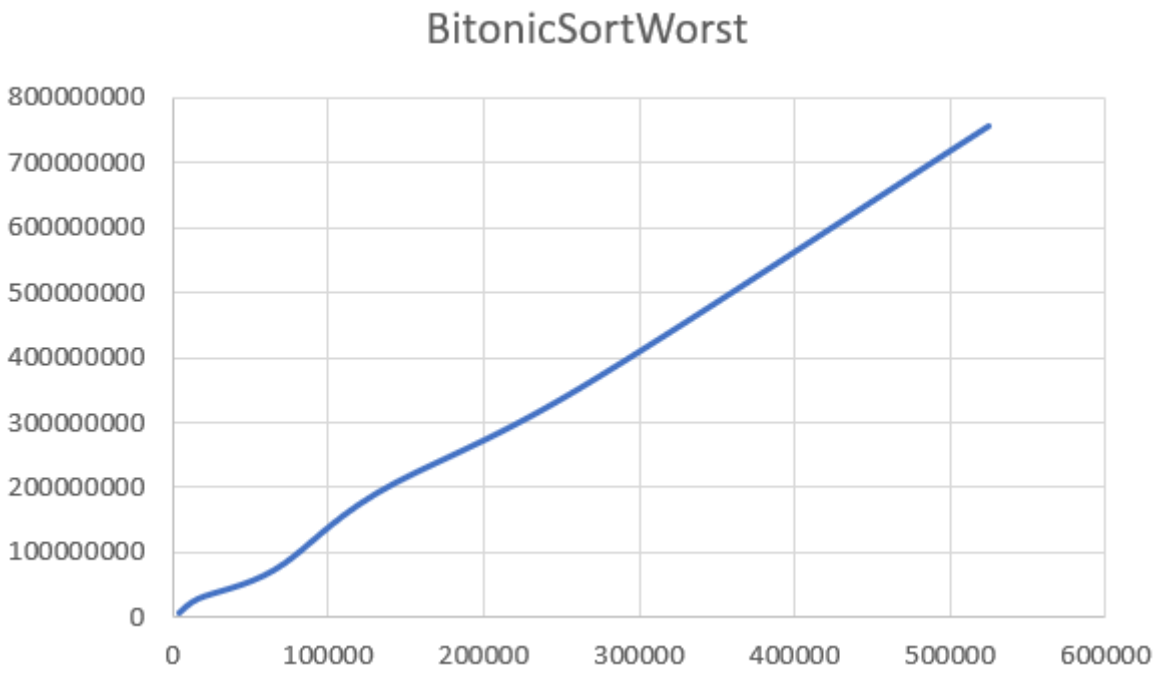
Worst-case performance	$O(\log^2(n))$ parallel time
Best-case performance	$O(\log^2(n))$ parallel time
Average performance	$O(\log^2(n))$ parallel time
Worst-case space complexity	$O(n\log^2(n))$ non-parallel time

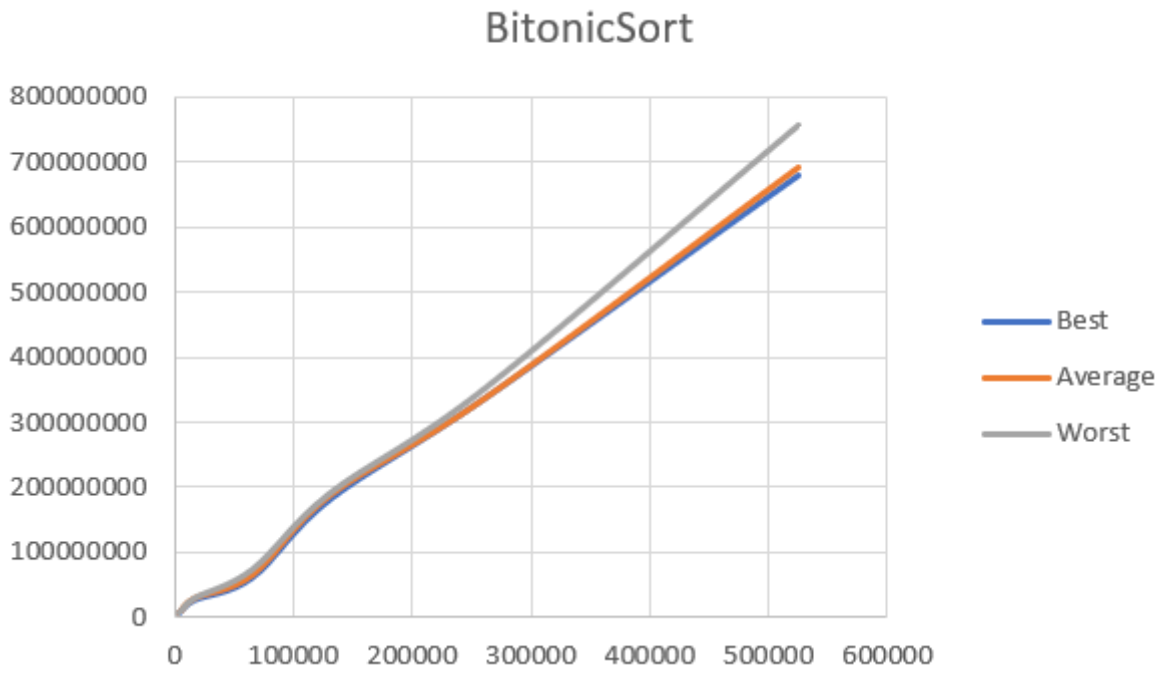
Bitonic sort execution time(Nano second) in Java

Best	Average	Worst	Input(n)
5924800	6043500	6761400	4096
24948500	27877700	28574600	16384
59959300	64175100	73263500	65536
180662000	187366400	191054800	131072
337048000	338189800	353499500	262144
678248600	692351000	756333200	524288


BitonicSortBest
800000000
700000000
600000000
500000000
400000000
300000000
200000000
100000000
0
0
100000
200000
300000
400000
500000
600000


BitonicSortAverage
800000000
700000000
600000000
500000000
400000000
300000000
200000000
100000000
0
0
100000
200000
300000
400000
500000
600000


BitonicSortWorst
800000000
700000000
600000000
500000000
400000000
300000000
200000000
100000000
0
0
100000
200000
300000
400000
500000
600000


BitonicSort
800000000
700000000
600000000
500000000
400000000
300000000
200000000
100000000
0
0
100000
200000
300000
400000
500000
600000
Best
Average
Worst

Comb Sort Algorithms

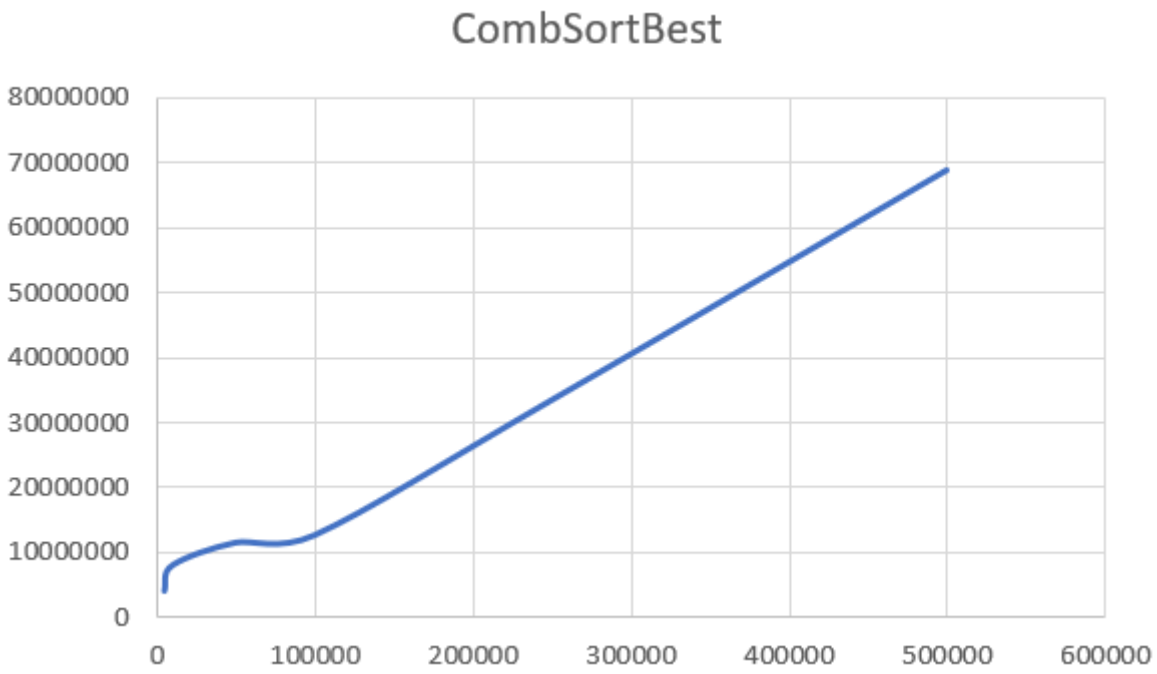
The basic idea of comb sort and the bubble sort is same. In other words, comb sort is an improvement on the bubble sort. In the bubble sorting technique, the items are compared with the next item in each phase. But for the comb sort, the items are sorted in a specific gap. After completing each phase, the gap is decreased. The decreasing factor or the shrink factor for this sort is 1.3. It means that after completing each phase the gap is divided by 1.3.

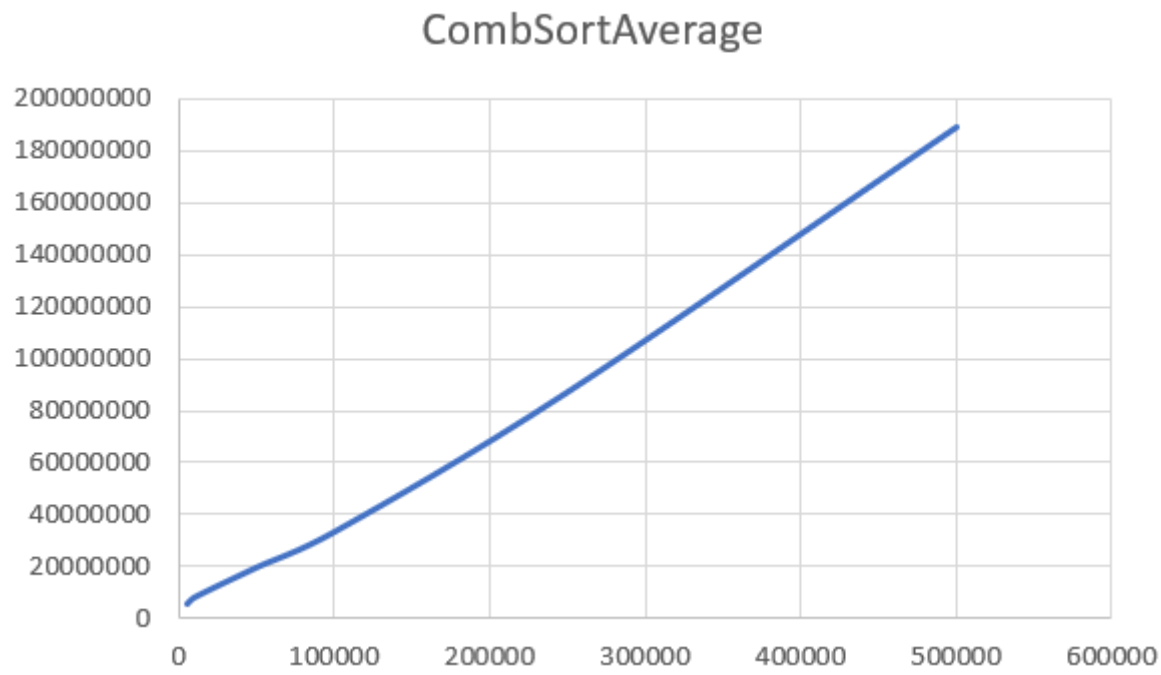
Complexity

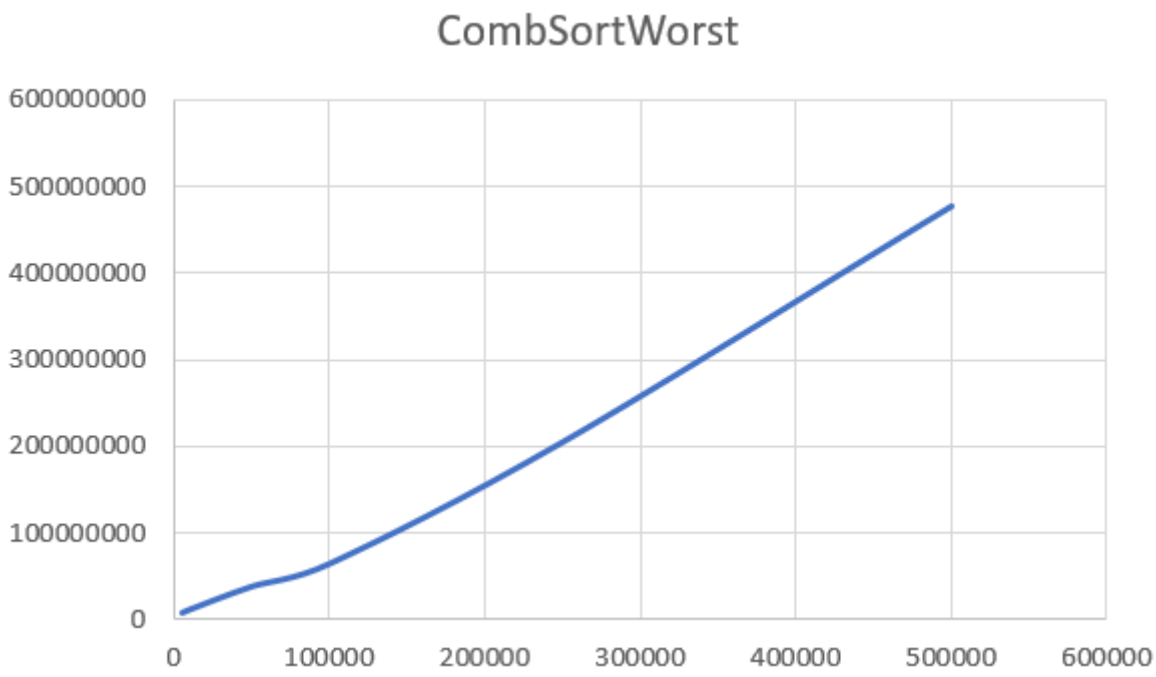
Worst-case performance	$O(n^2)$[1]
Best-case performance	$\Theta(n \log n)$
Average performance	$\Omega(n^2/2^p)$, where p is the number of increments[1]
Worst-case space complexity	$O(1)$

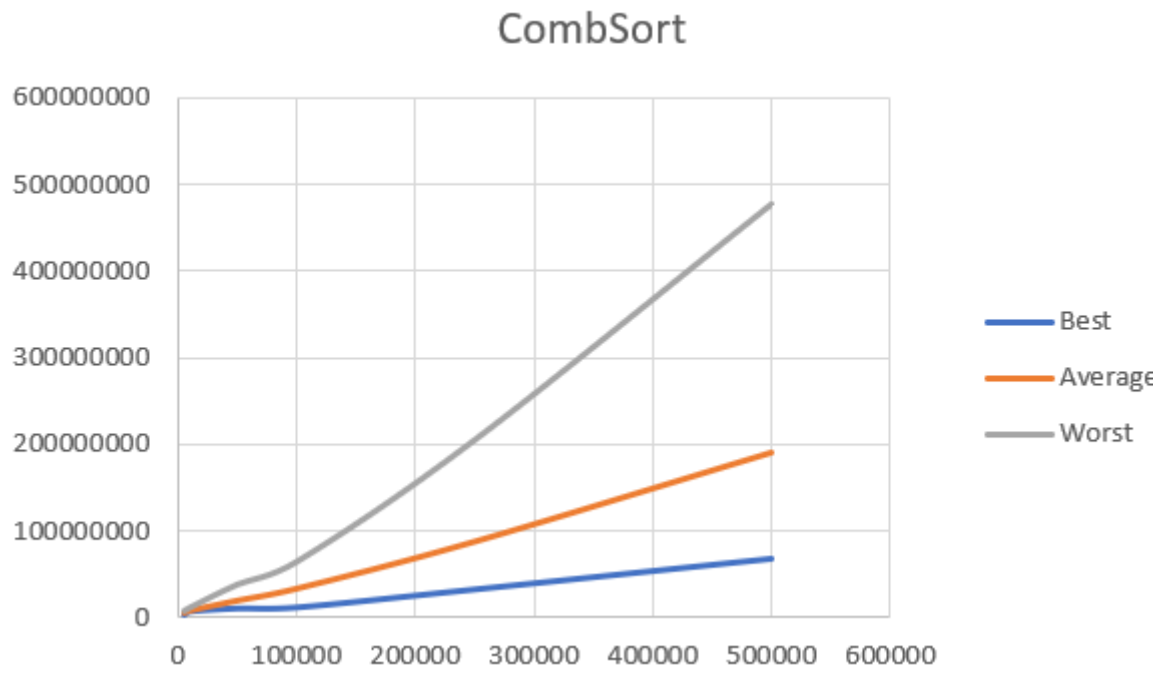
Comb sort execution time(Nano second) in Java

Best	Average	Worst	Input(n)
4031100	5540800	6821500	5000
7949500	8122100	10709300	10000
11446600	19656400	37371200	50000
12586800	33353600	63733700	100000
33371100	87239100	204522300	250000
68742800	189237400	477079600	500000

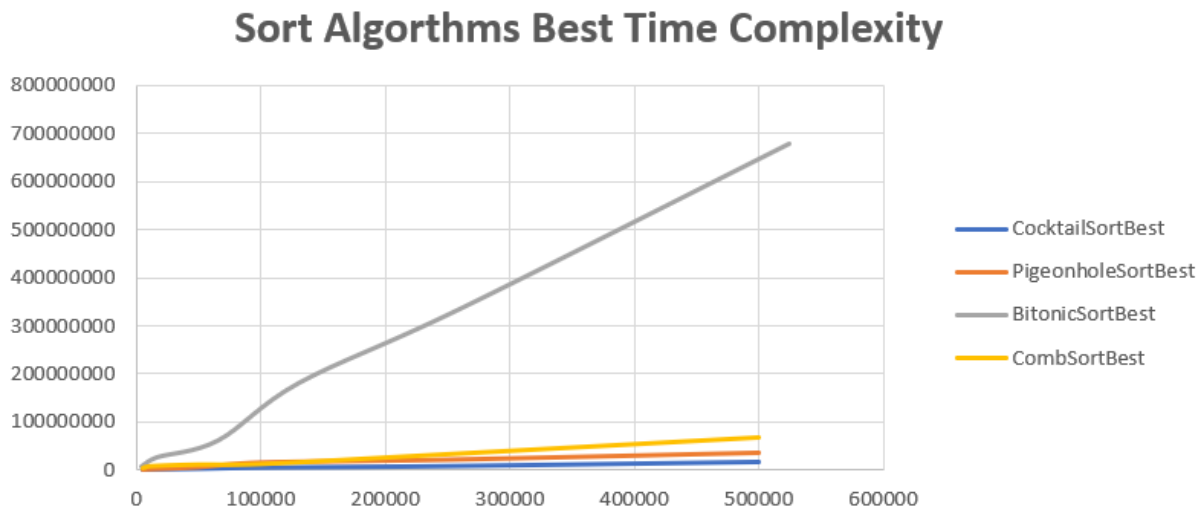

CombSortBest
80000000
70000000
60000000
50000000
40000000
30000000
20000000
10000000
0
0
100000
200000
300000
400000
500000
600000

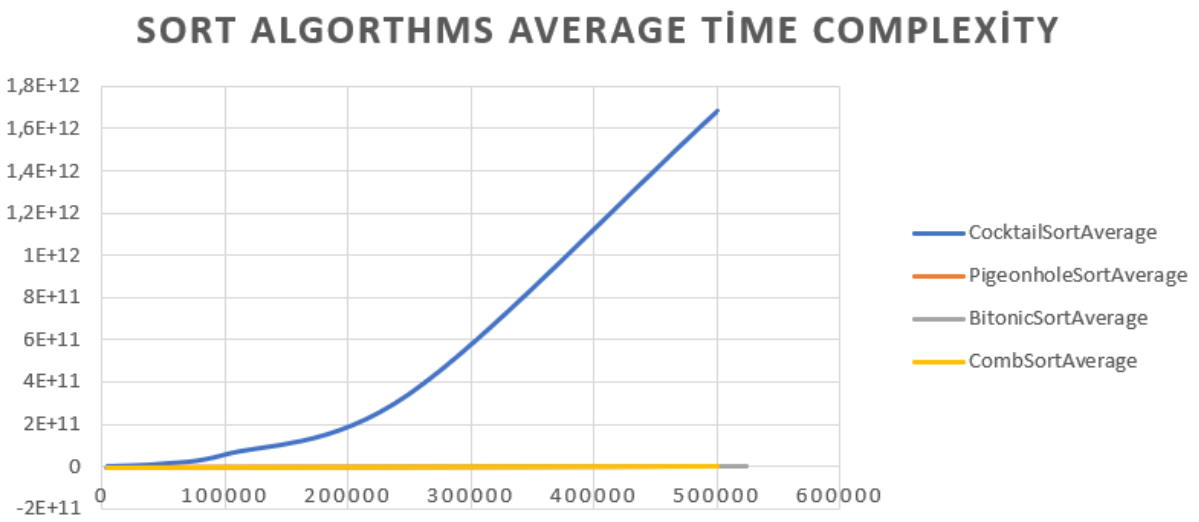

CombSortAverage
200000000
180000000
160000000
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
0
100000
200000
300000
400000
500000
600000

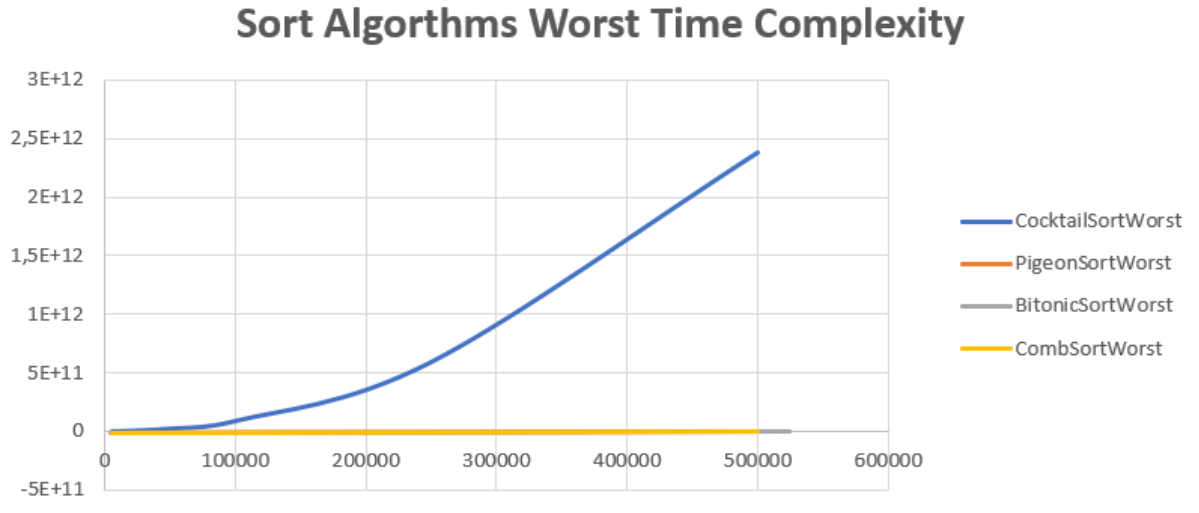

CombSortWorst
600000000
500000000
400000000
300000000
200000000
100000000
0
0
100000
200000
300000
400000
500000
600000


CombSort
600000000
500000000
400000000
300000000
200000000
100000000
0
0
100000
200000
300000
400000
500000
600000
Best
Average
Worst

Algorithm Comparisons







Algorithm Comparisons Continue

- For the best case, the most effective algorithm of all n values is the cocktail sort algorithm. Since there is an algorithm that sorts the entered value of n as soon as possible.
- For the average case, the most effective algorithm of all n values is the pigeonhole sort algorithm. Since there is an algorithm that sorts the entered value of n as soon as possible.
- For the worst case, the most effective algorithm of all n values is the pigeonhole sort algorithm. Since there is an algorithm that sorts the entered value of n as soon as possible.
- As the N value increases, the performance of the cocktail sort algorithms decreases due to O (n^2).
- While bitonic sort algorithms and comb sort algorithms give bad results in average case scenario, bad case scenario also give average results.
- In the worst case scenario, the pigeonhole sort algorithms works best, but in comb and bitonic it gives an average result.
- The cocktail sort algorithms gives bad results in worst case scenario and average case scenario.
- In worst case scenario and average case scenario, comb sort algorithms and bitonic sort algorithms give average results.
- But In worst case scenario and average case scenario, pigeonhole sort algorithm give best results.
- The most effective algorithm is the pigeonhole algorithm according to the observations I have made and the values I have tried.