REPORT

Insertion Sort

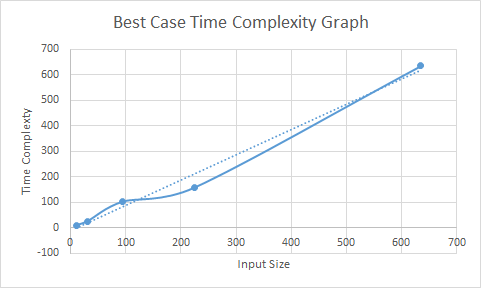
**Insertion sort** is a simple [sorting algorithm](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU29ydGluZ19hbGdvcml0aG0) that builds the final [sorted array](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU29ydGVkX2FycmF5) (or list) one item at a time.

We prepare 5 best cases input files, 5 avarage input cases and 5 worst cases with different kind of inputs and different size (there are range of (0-1000) integers in files) and counts in the codes as metrics for this algorithm’s complexity.

For best case :

We thought that for the best case, the inputs should be ordered from large to small or small to large or be already sorted. And the size of inputs should be as small as possible.

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY(Empirical Results) |
| 10 | 10 |
| 30 | 25 |
| 95 | 103 |
| 225 | 158 |
| 635 | 635 |



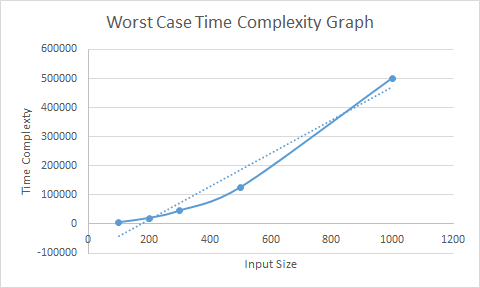
By the empirical results time complexity are approximately n for n input. So we can say O(n) for empirical results and O(n) for theoretical results. And our findings meet theoretical expectations.

For worst case:

We thought that for the **worst case** is when our list is in the exact opposite order our need. For example we need to order our list increasing order but list is in descending order.

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 100 | 5150 |
| 200 | 20100 |
| 300 | 45450 |
| 500 | 125750 |
| 1000 | 501500 |

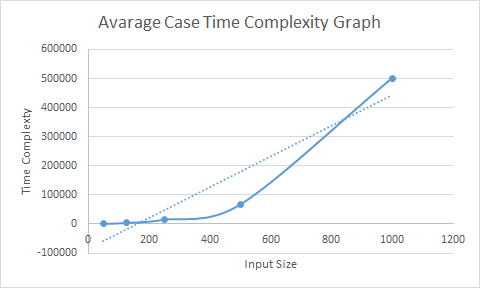
By the empirical results time complexity are approximately n^2/2 for n input. So we can say O(n^2) for empirical results and O(n^2) for theoretical results. And our findings meet theoretical expectations.



For avarage case:The average case is also quadratic.

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 50 | 749 |
| 125 | 3975 |
| 250 | 15229 |
| 500 | 66797 |
| 1000 | 501500 |

By the empirical results time complexity are approximately n^2/4 for n input. So we can say O(n^2) for empirical results and O(n^2) for theoretical results. And our findings meet theoretical expectations.



Merge Sort

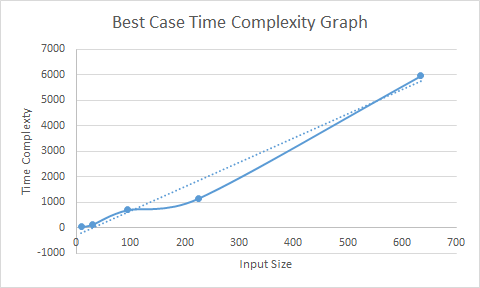
In [computer science](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ29tcHV0ZXJfc2NpZW5jZQ), **merge sort** (also commonly spelled **mergesort**) is an efficient, general-purpose, [comparison-based](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ29tcGFyaXNvbl9zb3J0) [sorting algorithm](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU29ydGluZ19hbGdvcml0aG0).

We prepare 5 best cases input files, 5 avarage input cases and 5 worst cases with different kind of inputs and different size (there are range of (0-1000) integers in files) and counts in the codes as metrics for this algorithm’s complexity.

For best case :

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 10 | 39 |
| 30 | 124 |
| 95 | 704 |
| 225 | 1148 |
| 635 | 5972 |

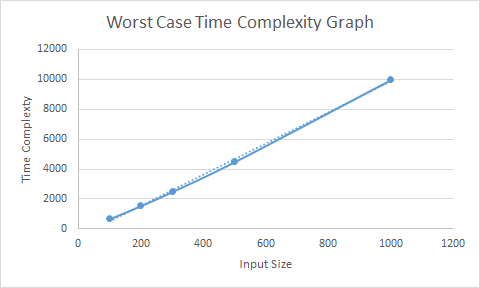
By the empirical results time complexity are approximately 3(n\*logn) for n input. So we can say O(n\*logn) for empirical results and O(n\*logn) for theoretical results. And our findings meet theoretical expectations.

1

For worst case:

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 100 | 680 |
| 200 | 1553 |
| 300 | 2498 |
| 500 | 4498 |
| 1000 | 9987 |

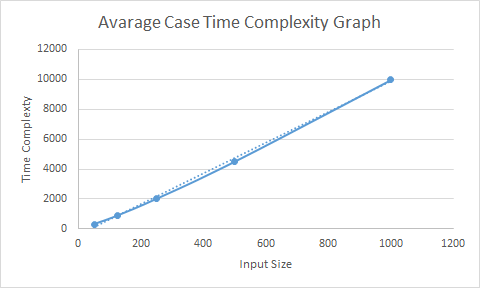
By the empirical results time complexity are approximately 3(n\*logn) for n input. So we can say O(n\*logn) for empirical results and O(n\*logn) for theoretical results. And our findings meet theoretical expectations.



For avarage case:

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 50 | 293 |
| 125 | 880 |
| 250 | 2003 |
| 500 | 4498 |
| 1000 | 9987 |

By the empirical results time complexity are approximately 3(n\*logn) for n input. So we can say O(n\*logn) for empirical results and O(n\*logn) for theoretical results. And our findings meet theoretical expectations.



Quick Sort

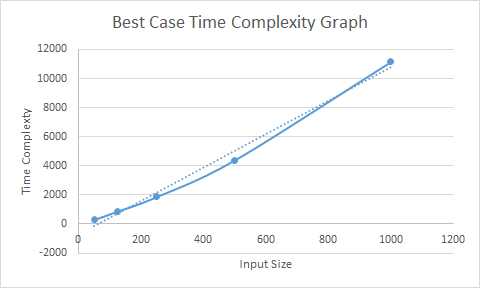
**Quicksort** (sometimes called **partition-exchange sort**) is an [efficient](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQWxnb3JpdGhtX2VmZmljaWVuY3k) [sorting algorithm](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU29ydGluZ19hbGdvcml0aG0), serving as a systematic method for placing the elements of a [random access](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUmFuZG9tX2FjY2Vzcw) [file](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ29tcHV0ZXJfZmlsZQ) or an [array](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXJyYXlfZGF0YV9zdHJ1Y3R1cmU) in order.

We prepare 5 best cases input files, 5 avarage input cases and 5 worst cases with different kind of inputs and different size (there are range of (0-1000) integers in files) and counts in the codes as metrics for this algorithm’s complexity.

For best case :

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 50 | 277 |
| 125 | 856 |
| 250 | 1857 |
| 500 | 4384 |
| 1000 | 11161 |

By the empirical results time complexity are approximately 7/2(n\*logn) for n input. So we can say O(n\*logn) for empirical results and O(n\*logn) for theoretical results. And our findings meet theoretical expectations.

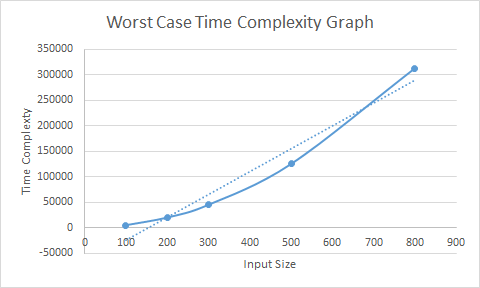


For worst case:

We thought that for the worst case, the inputs should be ordered from large to small or small to large or be already sorted.

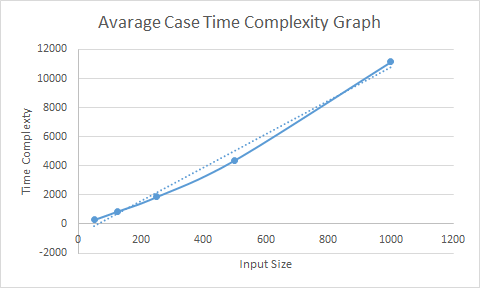
|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 100 | 5050 |
| 200 | 20100 |
| 300 | 45150 |
| 500 | 125250 |
| 800 | 312885 |

By the empirical results time complexity are approximately 1/2(n^2) for n input. So we can say O(n^2) for empirical results and O(n^2) for theoretical results. And our findings meet theoretical expectations.



For avarage case:

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 50 | 277 |
| 125 | 856 |
| 250 | 1857 |
| 500 | 4384 |
| 1000 | 11161 |



By the empirical results time complexity are approximately 7/2(n\*logn) for n input. So we can say O(n\*logn) for empirical results and O(n\*logn) for theoretical results. And our findings meet theoretical expectations.

Heap Sort

In [computer science](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ29tcHV0ZXJfc2NpZW5jZQ), **heapsort** is a [comparison-based](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ29tcGFyaXNvbl9zb3J0) [sorting algorithm](http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU29ydGluZ19hbGdvcml0aG0).

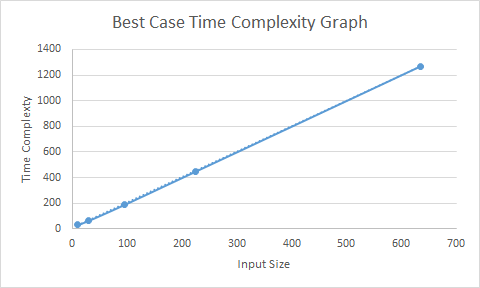
We prepare 5 best cases input files, 5 avarage input cases and 5 worst cases with different kind of inputs and different size (there are range of (0-1000) integers in files) and counts in the codes as metrics for this algorithm’s complexity.

For best case :

We thought that for the best case, the inputs are the same. And the size of inputs should be as small as possible.

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 10 | 32 |
| 30 | 66 |
| 95 | 190 |
| 225 | 450 |
| 635 | 1270 |

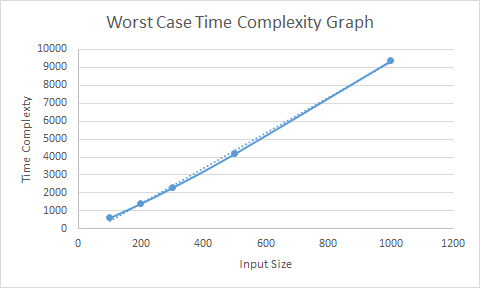
By the empirical results time complexity are approximately 2n for n input. So we can say O(n) for empirical results and O(n//for all input are same) for theoretical results. And our findings meet theoretical expectations.



For worst case:

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 100 | 620 |
| 200 | 1428 |
| 300 | 2302 |
| 500 | 4202 |
| 1000 | 9384 |

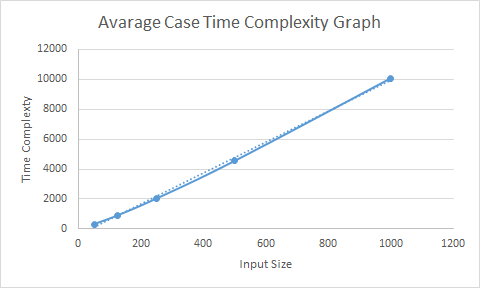
By the empirical results time complexity are approximately 3(n\*logn) for n input. So we can say O(n\*logn) for empirical results and O (n\*logn) for theoretical results. And our findings meet theoretical expectations.



For avarage case:

|  |  |
| --- | --- |
| INPUT SIZE | TIME COMPLEXITY (Empirical Results) |
| 50 | 302 |
| 125 | 891 |
| 250 | 2025 |
| 500 | 4543 |
| 1000 | 10077 |

By the empirical results time complexity are approximately 4(n\*logn) for n input. So we can say O(n\*logn) for empirical results and O(n\*logn) for theoretical results. And our findings meet theoretical expectations.



Counting Sort

The counting sort algorithm is unique in that it can only be implemented on integers. This is part of what limits this algorithm’s usability — and it’s possibly part of the reason that you may have never heard or encountered it before! Counting sort takes in a range of integers to be sorted. It uses the range of the integers (for example, the range of integers between 0–100), and counts the number of times each unique number appears in the unsorted input.Counting sort has advantages if the range (difference between minimum and maximum number) is little .Otherwise its space complexity is increases.

N: input size

K: range of elements

Time Complexity:

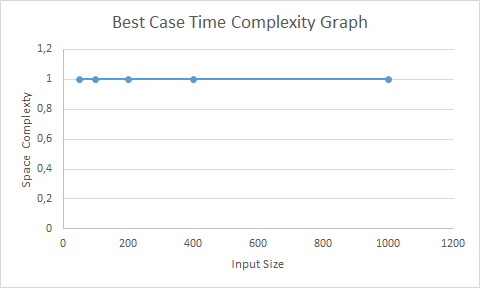
Time complexity is same for best ,avarage, and worst case that is O(n+k) ≈ O(n) .Because in every case , we pick element and find its index in the array and increase the array index.In every time this takes n operation , n times .So we use space complexity.

Space Complexity:

Best Case :

When the algorithm analysize with space complexity , we found that when every input choose same space complexity is O(1) for best case.

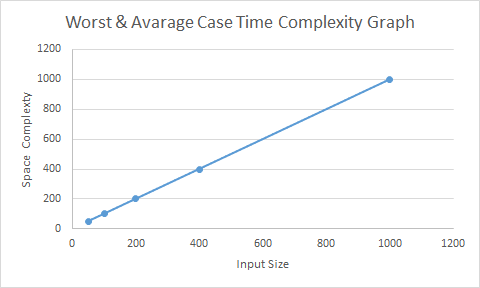
|  |  |  |
| --- | --- | --- |
| Range | Input Size | Space Complexity |
| 1 | 50 | 1 |
| 1 | 100 | 1 |
| 1 | 200 | 1 |
| 1 | 400 | 1 |
| 1 | 1000 | 1 |



Avarage & Worst Case:

The worst case is O(N+K) which is difference between minimum and maximum number is K.

|  |  |  |
| --- | --- | --- |
| Range | Input Size | Space Complexity |
| 50 | 1000 | 50 |
| 100 | 1000 | 100 |
| 200 | 1000 | 200 |
| 400 | 1000 | 400 |
| 1000 | 1000 | 1000 |



Analyzing Results:

\*Insertion sort gives the best performance for the ascending or descending ordered inputs ,however quick sort gives the worst performance for these inputs.

\* Insertion sort is less efficient on large size input lists than quicksort, heapsort, or merge sort algorithms. However insertion sort is more efficient on small size input lists than other algorithms.

\*Merge sort can be applied files of any size. But merge sort less efficient than other sort.

\*Quick sort is very efficient algorithms. because it gives best performance on large list of inputs.But it gives worst performance on ascending or descending ordered input lists.

\* Disadvantage of quick sort is that its worst-case performance is similar to average case performances of the insertion sort.

\* Heap sort is very efficiency algorithms. This means it performs equally well in the best, average and worst cases.

\* Merge sort and heap sort are independent of distribution of inputs.

\* Counting sort is more efficient if the range of inputs is smaller and inefficient for large inputs. Space complexity is change according to range of inputs.

BÜŞRA GÖKMEN