BLG 335E
Analysis of Algorithms 1
Project 1

<u>Lecturer</u>:

Hazım Kemal Ekenel

<u>Student</u>:

Göktürk GÖK

150110029

<u>Due Date</u>:

21.10.2016, 23:59

A. Asymptotic Upper Bound

There are 3 main situation as worst-case, average-case and best case of algorithms. And each of them is described as big O notation which specify the upper bound for the worst-case , big Theta Notation which is located between upper and lower bound for the average-case and big Omega Notation that specify the lower bound for the best-case.

1. Linear Search

my implementation : Normally there would be a key then you can search the list item by item until the key matches the element of an array. But in our project there is a temporary array(**temp**) which keeps K-closest warehouses instead of just a key. My implementation includes two nested loops which the inner loop is **K** times and outer loop is **N-K** times. So **N*K** is the upper bound of linear search in this project instead of finding a key in an array has just N.

best case: 1 (big omega notation)
worst case: n (big O notation)

2. Insertion Sort

my implementation : There are two nested loops which both are run N (range) times. So upper-bound of my **insertion_sorter** is **N^2** .
Here N is the number of warehouses which is taken as an argument from terminal.

best case: n (big omega notation)
average case: n^2 (big theta notation)
worst case: n^2 (big O notation)

3. Merge Sort

my implementation : There is a divide and conquer algorithm here using merge sort algorithm. Asymptotic upper bound of Merge Sort Algorithm is **O(n*log(n))**. So when the number of warehouses is taken as N, execution time will be multiplied by **N*(log(N))**.

best case: nlogn (big omega notation)
average case: nlogn (big theta notation)
worst case: nlogn (big O notation)

B. Execution Time (sec)

1. Linear Search

	K = 1	K = 2	K = 10	K = N/2
N = 10	0	0	0	0
N = 100	0	0	0	0
N = 1000	0	0	0.0001	0.004
N = 1 000 000	0.001	0.03	0.13	3294

2. Insertion Sort

	K = 1	K = 2	K = 10	K = N/2
N = 10	0	0	0	0
N = 100	0	0	0	0
N = 1000	0.003	0.003	0.0029	0.0029
N = 1 000 000	3304	3387	3467	3412

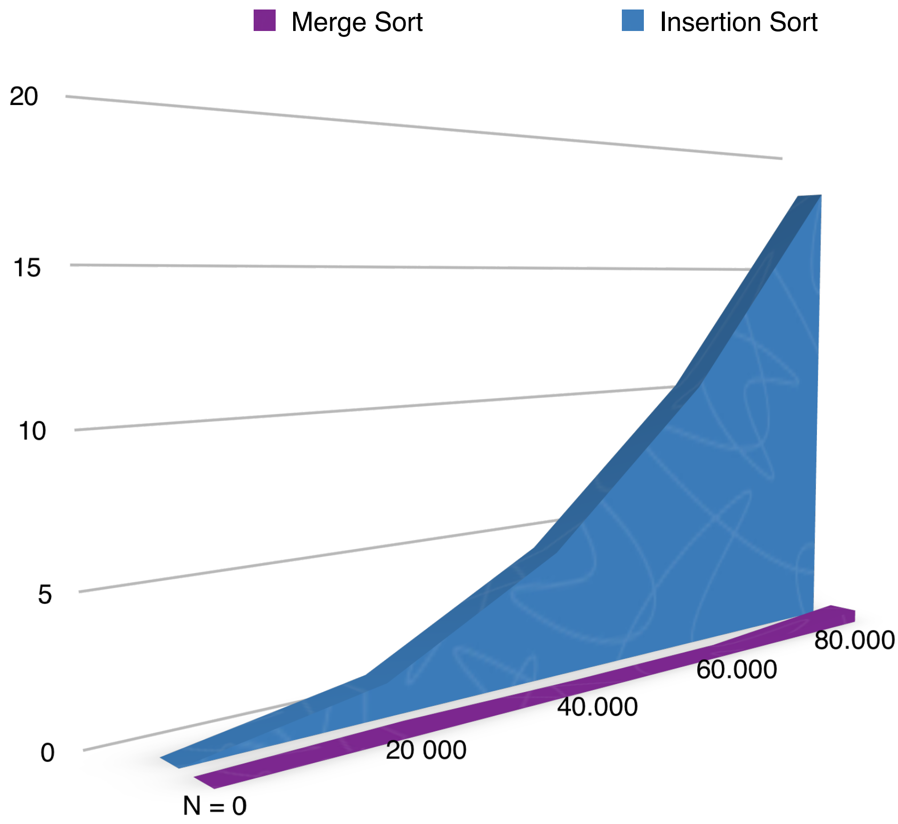
3. Merge Sort

	K = 1	K = 2	K = 10	K = N/2
N = 10	0	0	0	0
N = 100	0	0	0	0
N = 1000	0.0004	0.0005	0.0004	0.0004
N = 1 000 000	0.7132	0.7013	0.6926	0.6883

4. Average Times of Execution Time of Algorithms

	Linear Search	Insertion Sort	Merge Sort
N = 10	0	0	0
N = 100	0	0	0
N = 1000	0.002	0.00295	0.00042
N = 1 000 000	823.54	3392.5	0.526

C. 2-Line Plotting of Running Time Complexity



N (Number of data)

As it is shown above that if there is a sorting around small amount of data(ex: 500) , there are not so big differences between insertion sort and merge sort

But Merge sort is more effective than insertion sort when sorting big amount of data (ex;more than 1000) because it has much more less complexity than insertion sort (**logn < n after specific point**) .