



**BLG 335E**

# **ANALYSIS OF ALGORITHMS I**

CRN: 10825

## **REPORT OF HOMEWORK #1**

Submission Date: 22.10.2013

**STUDENT NAME: TUĞRUL YATAĞAN**

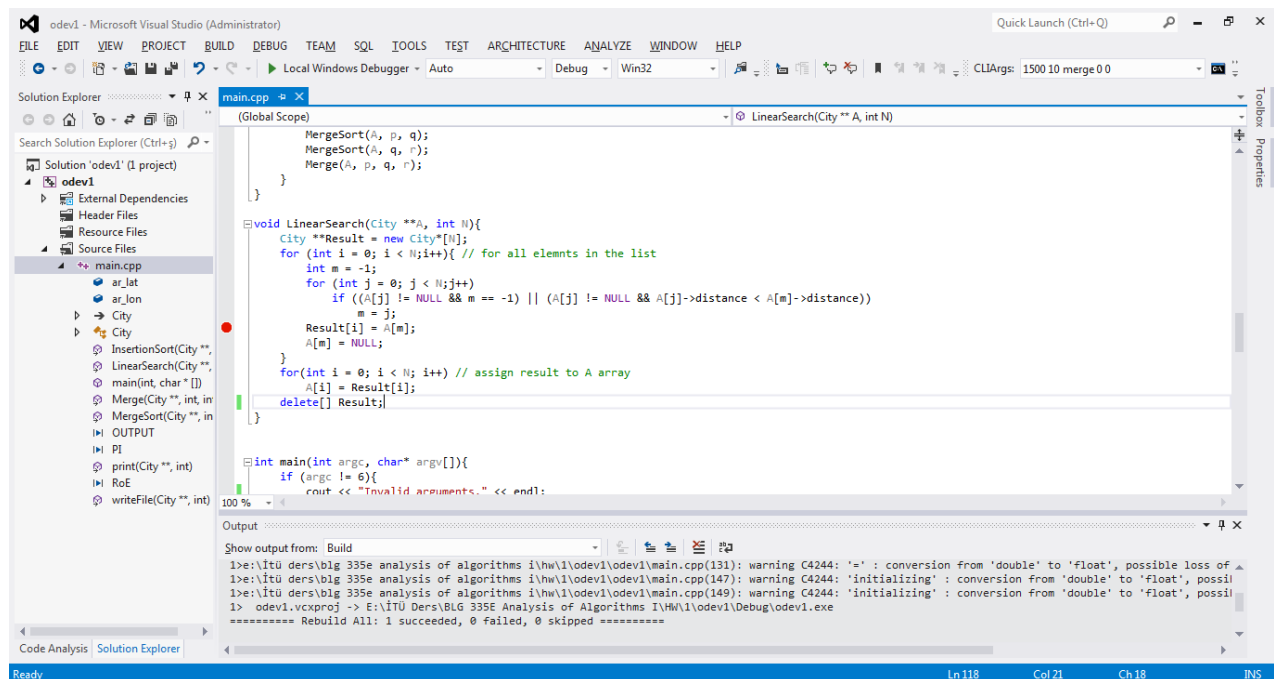
**STUDENT NUMBER: 040100117**

# 1. Introduction

In this project, we will find closest K locations among N locations to a given location considering their coordinates on earth.

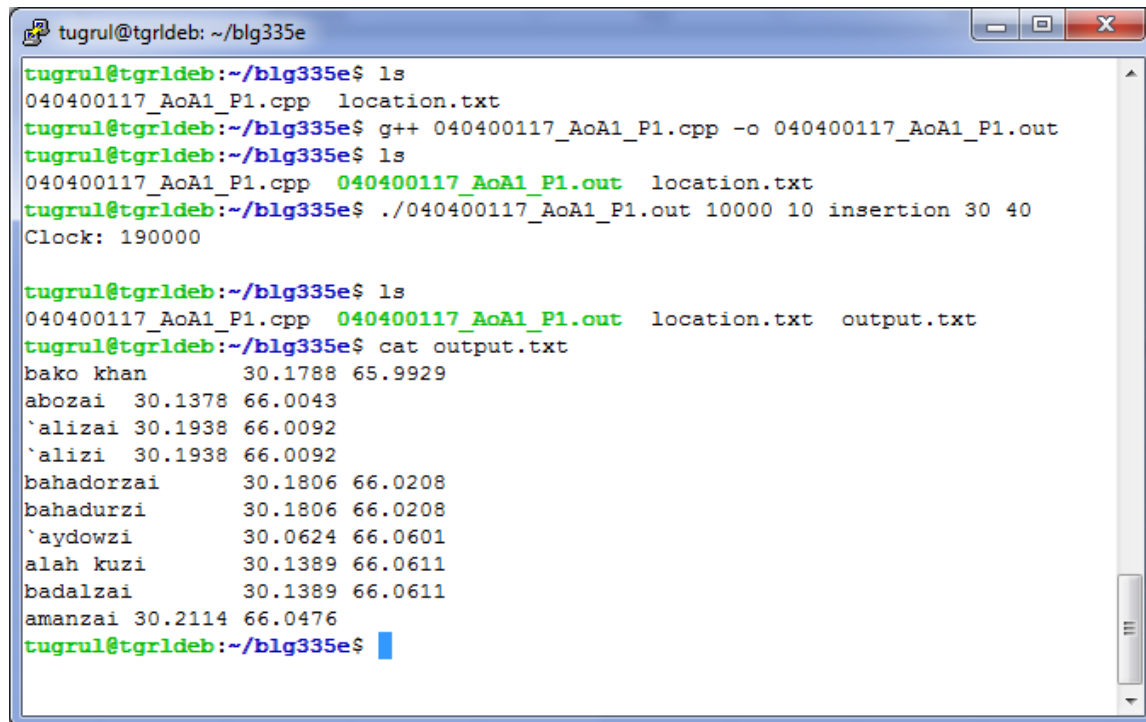
## 2. Development and Operating Environments

Microsoft Visual C++ 2012 environment has been used to write the source code in Windows 7 operation system and GNU g++ compiler has been used for compiling under Debian 7 operation system.



```
odev1 - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Solution Explorer
main.cpp
(Global Scope)
MergeSort(A, p, q);
MergeSort(A, q, r);
Merge(A, p, q, r);
}
}
void LinearSearch(City **A, int N){
    City **Result = new City*[N];
    for (int i = 0; i < N; i++){ // for all elements in the list
        int m = -1;
        for (int j = 0; j < N; j++){
            if ((A[j] != NULL && m == -1) || (A[j] != NULL && A[j]->distance < A[m]->distance))
                Result[i] = A[j];
            A[m] = NULL;
        }
        for (int i = 0; i < N; i++) // assign result to A array
            A[i] = Result[i];
        delete[] Result;
    }
}
int main(int argc, char* argv[]){
    if (argc != 6){
        cout << "Invalid arguments." << endl;
    }
}
Output
Show output from: Build
1>e:\it\ders\blg 335e analysis of algorithms i\hw\1\odev1\odev1\main.cpp(131): warning C4244: '=' : conversion from 'double' to 'float', possible loss of
1>e:\it\ders\blg 335e analysis of algorithms i\hw\1\odev1\odev1\main.cpp(147): warning C4244: 'initializing' : conversion from 'double' to 'float', possi
1>e:\it\ders\blg 335e analysis of algorithms i\hw\1\odev1\odev1\main.cpp(149): warning C4244: 'initializing' : conversion from 'double' to 'float', possi
1> odev1.vcxproj -> E:\IT\ Ders\BLG 335E Analysis of Algorithms I\HW\1\odev1\Debug\odev1.exe
***** Rebuild All: 1 succeeded, 0 failed, 0 skipped *****
Ready Ln118 Col 21 Ch 18 INS
```

The program built and compiled without any warning or error under g++. Finally the program is executed. Sample outcome is below:



```
tugrul@tgrldeb: ~/blg335e
tugrul@tgrldeb:~/blg335e$ ls
040400117_AoA1_P1.cpp  location.txt
tugrul@tgrldeb:~/blg335e$ g++ 040400117_AoA1_P1.cpp -o 040400117_AoA1_P1.out
tugrul@tgrldeb:~/blg335e$ ls
040400117_AoA1_P1.cpp  040400117_AoA1_P1.out  location.txt
tugrul@tgrldeb:~/blg335e$ ./040400117_AoA1_P1.out 10000 10 insertion 30 40
Clock: 190000

tugrul@tgrldeb:~/blg335e$ ls
040400117_AoA1_P1.cpp  040400117_AoA1_P1.out  location.txt  output.txt
tugrul@tgrldeb:~/blg335e$ cat output.txt
bako khan      30.1788 65.9929
abozai 30.1378 66.0043
`alizai 30.1938 66.0092
`alizi 30.1938 66.0092
bahadorzai     30.1806 66.0208
bahadurzi      30.1806 66.0208
`aydowzi       30.0624 66.0601
alah kuzi       30.1389 66.0611
badalzai       30.1389 66.0611
amanzai 30.2114 66.0476
tugrul@tgrldeb:~/blg335e$
```

### 3. Data Structures and Variables

The program takes 5 command line arguments. Example:

**./040100117\_AoA1\_P1 N K algorithmType latitude longitude**

**algorithmType** variables can be {"insertion", "merge" or "linear"}

**N, K, latitude** and **longitude** variables can be integer value.

Example use of the program:

**./040100117\_AoA\_P1 1000 10 insertion 30 40**

### 4. Analysis

Running time of sorting functions according to K and N numbers are shown below in tables:

merge		K			
		1	2	10	N/2
N	10	0	0	0	0
	100	0	0	0	0
	1000	1	1	1	1
	10000	10	10	10	10
	100000	103	102	102	103
	1000000	1140	1150	1129	1138

insertion		K			
		1	2	10	N/2
N	10	0	0	0	0
	100	0	0	0	0
	1000	2	2	2	2
	10000	108	109	109	108
	100000	6886	6768	6811	6848
	1000000	$\infty$	$\infty$	$\infty$	$\infty$

linear		K			
		1	2	10	N/2
N	10	0	0	0	0
	100	0	0	0	0
	1000	0	0	0	3
	10000	0	0	0	296
	100000	1	2	8	38480
	1000000	9	18	82	$\infty$

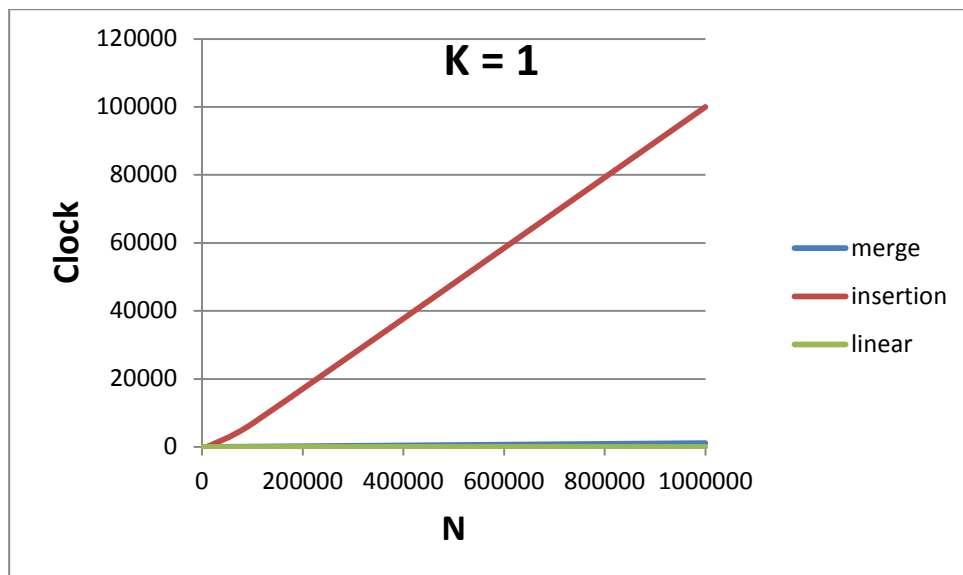
N is number of the element to be sorted and K is number of the element to demanded. As seen by the tables K does not effect on merge and insertion sorting, but K can effects only when linear sorting.

Merge sort is faster than insertion sort for  $N > 1000$ . After 1000, merge sort is becomes faster. Linear sort is handy only when K is very small or N is smaller than 1000.

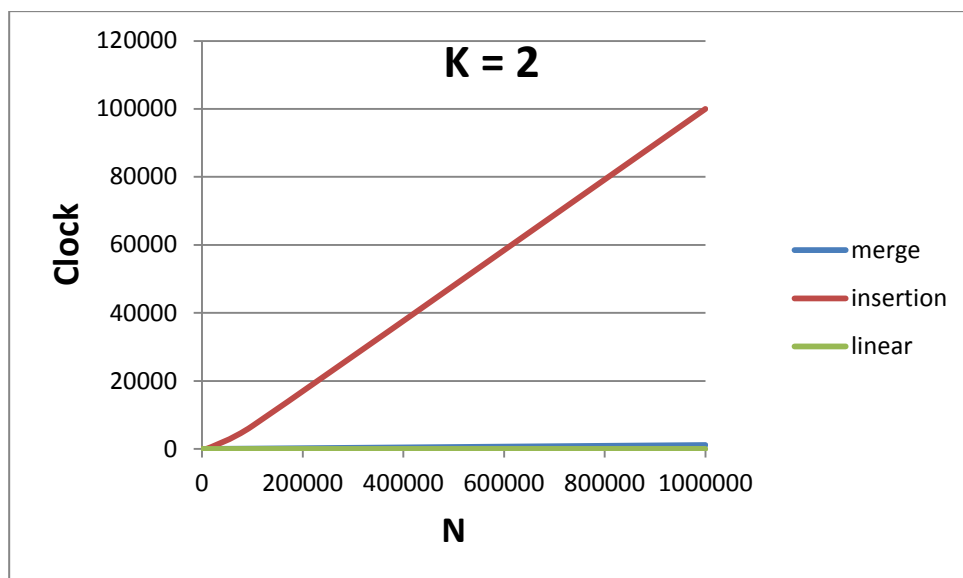
Related graphs according to N and clock numbers for all three sorting algorithms separately for K numbers are shown below.

Note: For clarity of graph, I choose very large number (Ex:1000000) for  $\infty$  when drawing graph.  $\infty$  indicates that program takes very long time to execute or it cannot give any result in reasonable time.

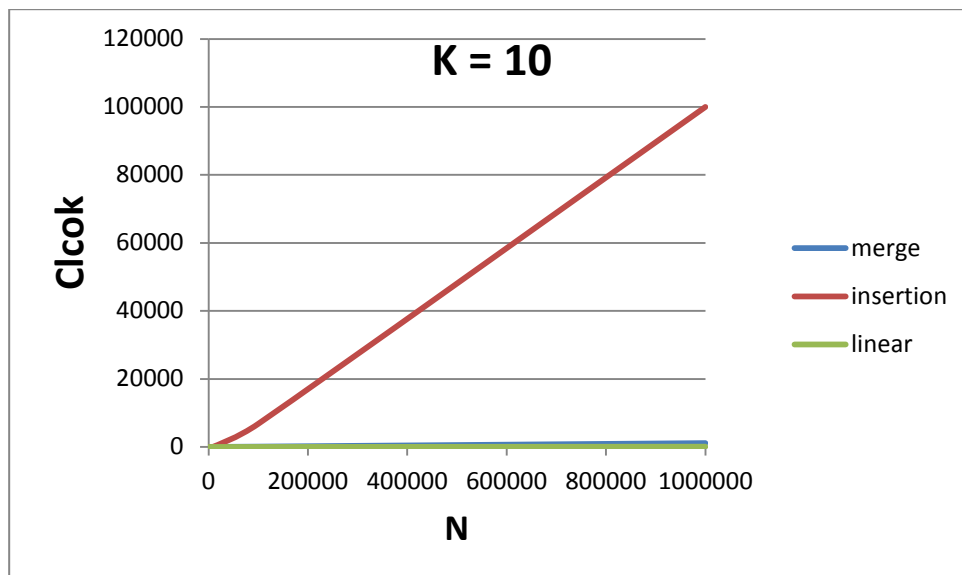
We can investigate the graphs that; if we chose time functions of merge sort is  $m(n)$ , insertion sort is  $i(n)$  and linear sort is  $l(n)$



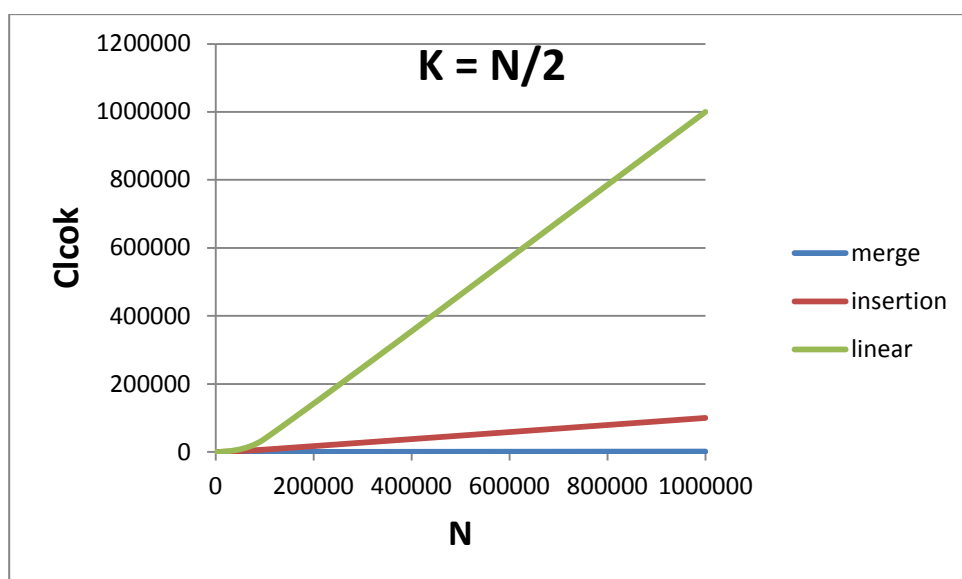
$$m(n) = O(l(n)) = O(i(n))$$



$$m(n) = O(l(n)) = O(i(n))$$



$$m(n) = O(l(n)) = O(i(n))$$



$$m(n) = O(i(n)) = O(l(n))$$

## 5. Conclusion

In this homework, I have become more familiar with the concept of analysis of algorithms. I had the chance to intensify my knowledge about instructing good and efficient algorithms.