CSCI311 ASSIGNMENT 2

PRIYANKA CHORDIA

PROGRAM 2 PROJECT REPORT

The program is developed to implement the three sorting algorithms viz. Insertion Sort, Merge Sort and Quick Sort on a census data. The program sorts the data using the three sorting algorithms, by population and by city name.

The tables below show the time(in seconds: nanoseconds) required to sort the program by these algorithm for multiple(5) runs.

- Time provided in seconds: nanoseconds

1. Running time CENSUS2010POP-Alabama-Alabama.csv 1102 records.

	1st Run	2nd Run	3rd Run	4th Run	5th Run
Insertion Sort					
By Population	0.4400445	0.4508988	0.4417334	0.4385773	0.4368111
By Name	0.20879069	0.20931317	0.20968456	0.20726751	0.20952766
Merge Sort					
By Population	0.1002239	0.879087	0.875952	0.874083	0.870569
By Name	0.1443109	0.1310222	0.1397384	0.1316638	0.1327779
Quick Sort					
By Population	0.423502	0.389002	0.403569	0.397982	0.394144
By Name	0.1036641	0.1033033	0.1027522	0.1084883	0.1029835

2. Running time CENSUS2010POP-Alabama-California.csv 3920 records.

	1st Run	2nd Run	3rd Run	4th Run	5th Run
Insertion Sort					
By Population	0.43620134	0.43766010	0.43664417	0.43943518	0.43504438
By Name	0.268735640	0.321958749	0.267693858	0.268392236	0.268141511
Merge Sort					
By Population	0.3416063	0.3357110	0.3386167	0.3379568	0.3337032
By Name	0.5473422	0.5547398	0.5483472	0.5371034	0.5420575
Quick Sort					
By Population	0.1658147	0.1620133	0.1622376	0.1613099	0.1633257
By Name	0.4606617	0.4520029	0.4616848	0.4534065	0.4561560

3. Running time CENSUS2010POP-Alabama-Idaho.csv 7932 records.

	1st Run	2nd Run	3rd Run	4th Run	5th Run
Insertion Sort					
By Population	0.27090306	0.25740032	0.25715944	0.25538633	0.25157605
By Name	1.38009985	1.20187786	1.20513918	1.20131837	1.18218541
Merge Sort					
By Population	0.8063837	0.7084811	0.7433826	0.7163790	0.7147416
By Name	0.1501892 5	0.1320939 4	0.12649078	0.12125215	0.12095124
Quick Sort					
By Population	0.4168797	0.3686329	0.3708098	0.3639391	0.3656084
By Name	0.1216684 0	0.9925192	0.9844475	0.9844947	0.9735514

4. Running time CENSUS2010POP-Alabama-Iowa.csv 21236 records.

	1st Run	2nd Run	3rd Run	4th Run	5th Run
Insertion Sort					
By Population	2.24485875	2.51185074	2.51103286	2.52735910	2.49954634
By Name	8.81288953	9.13456888	9.45105267	9.40541749	9.55215893
Merge Sort					
By Population	0.2086002	0.2069122	0.2063688	0.2070793	0.2067420
By Name	0.3820659	0.4025376	0.3995033	0.3822355	0.3778023
Quick Sort					
By Population	0.1172582	0.1185660	0.1175656	0.1174978	0.1191018
By Name	0.3240440	0.3239977	0.3242069	0.3231871	0.3255827

5. Running time CENSUS2010POP-Alabama-Missouri.csv 41712 records.

	1st Run	2nd Run	3rd Run	4th Run	5th Run
Insertion Sort					
By Population	9.97440183	8.74142511	8.93351138	10.5257023	9.38537302
By Name	40.2096467	35.4481915	38.8388777	35.9380599	34.9293412
Merge Sort					
By Population	0.48890682	0.46130955	0.48603755	0.46167166	0.47270541
By Name	0.10197400	0.8622190	0.10228877	0.9010975	0.9176421
Quick Sort					
By Population	0.37671433	0.34466105	0.37154936	0.35283526	0.36060035
By Name	0.86429121	0.77369449	0.84876411	0.77087031	0.79348167

6. Running time CENSUS2010POP.csv 81746 records.

	1st Run	2nd Run	3rd Run	4th Run	5th Run
Insertion Sort					
By Population	32.4041318	35.5791088	30.9649202	35.4687335	29.7461647
By Name	135.234572	139.442350	189.349013	137.369905	139.194654
Merge Sort					
By Population	0.9503002	0.9573011	0.9516645	0.9439723	0.9460579
By Name	0.1906403	0.1959393	0.2011915	0.1900717	0.2017816
Quick Sort					
By Population	0.8883025	0.9094618	0.9037235	0.8840174	0.8795343
By Name	0.1805886	0.1886537	0.1889348	0.1774877	0.1823877

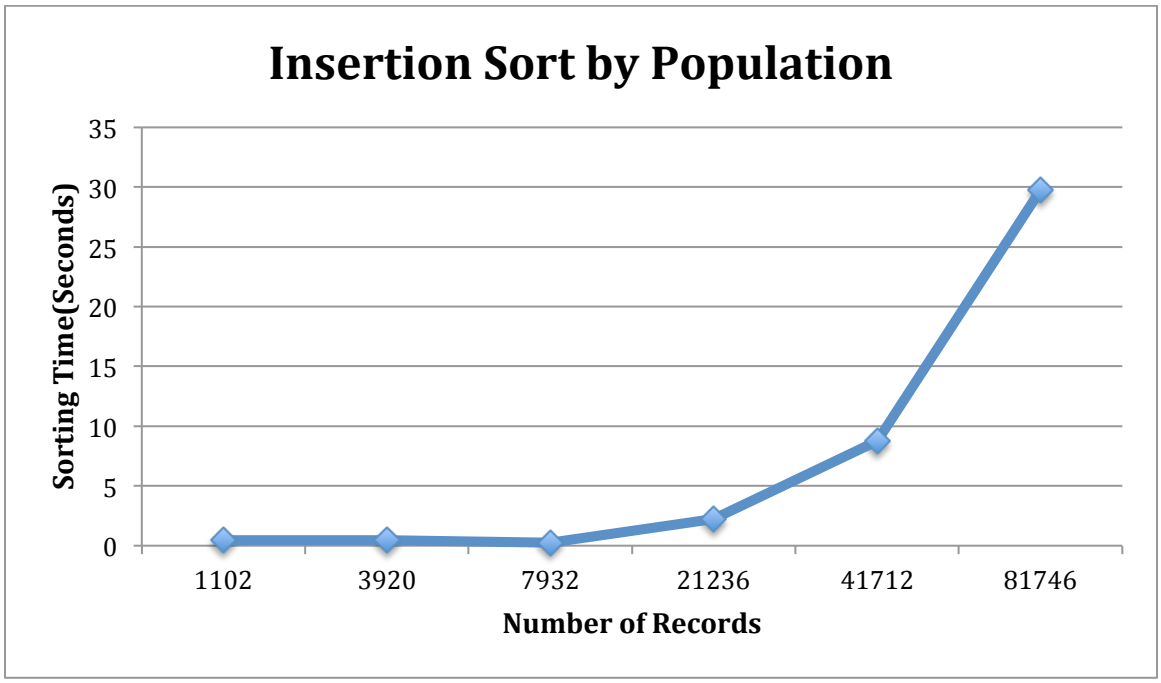
Now performing a close analysis on the data obtained from executing the 6 census record files for 5 times, we obtain a minimum value that is the minimum execution time from every run performed.

Following is the analysis table of all the sorts performed on all the files and the minimum running time obtained in it.

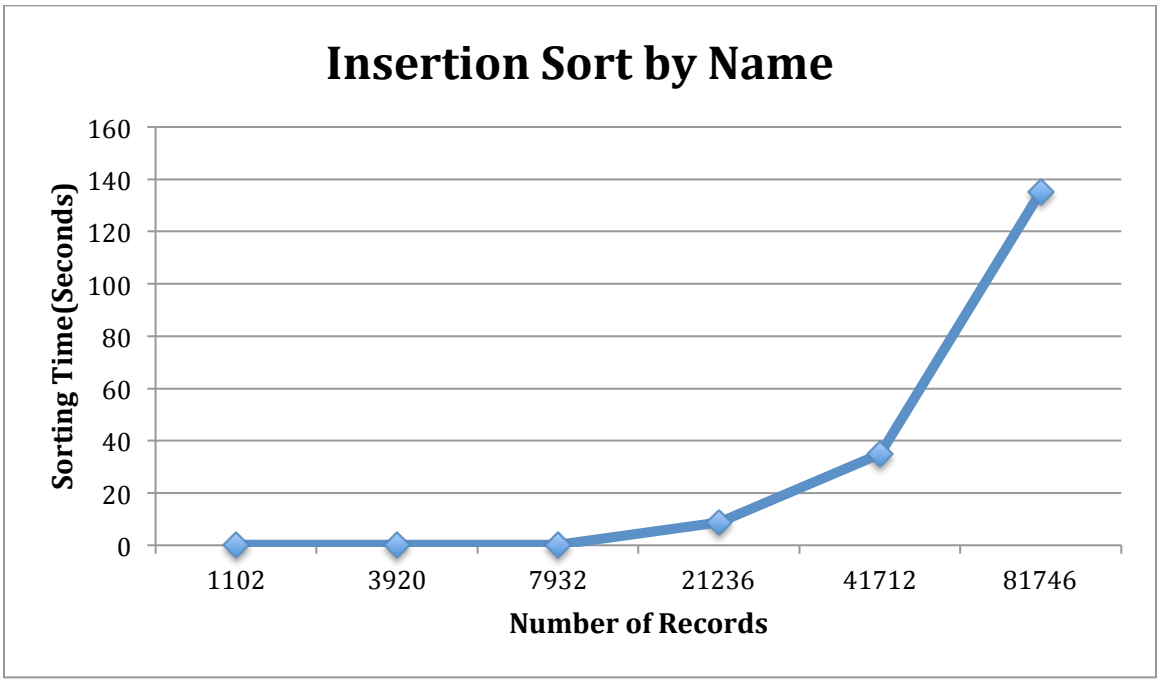
Analysis table of sorting on all the records.

# Records	1102	3920	7932	21236	41712	81746
Insertion Sort						
By Population	0.4368111	0.4350443	0.25157605	2.24485875	8.74142511	29.7461647
By Name	0.2072675	0.2676938	0.26769385	8.81288953	34.9293412	135.234572
Merge Sort						
By Population	0.870569	0.3337032	0.7084811	0.2063688	0.46130955	0.9439723
By Name	0.1310222	0.5371034	0.1209512	0.3778023	0.8622190	0.1900717
Quick Sort						
By Population	0.389002	0.1613099	0.3639391	0.1172582	0.34466105	0.8795343
By Name	0.1027522	0.4520029	0.9735514	0.3231871	0.77087031	0.1774877

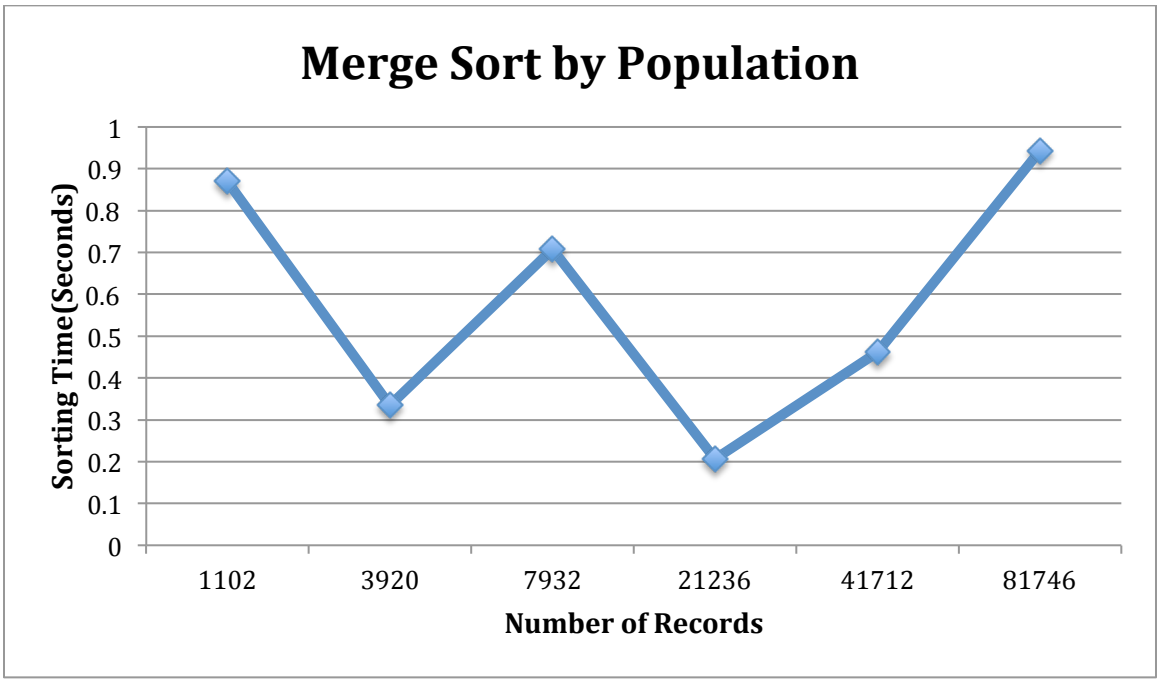
1. Graph of Insertion Sort (By Population)



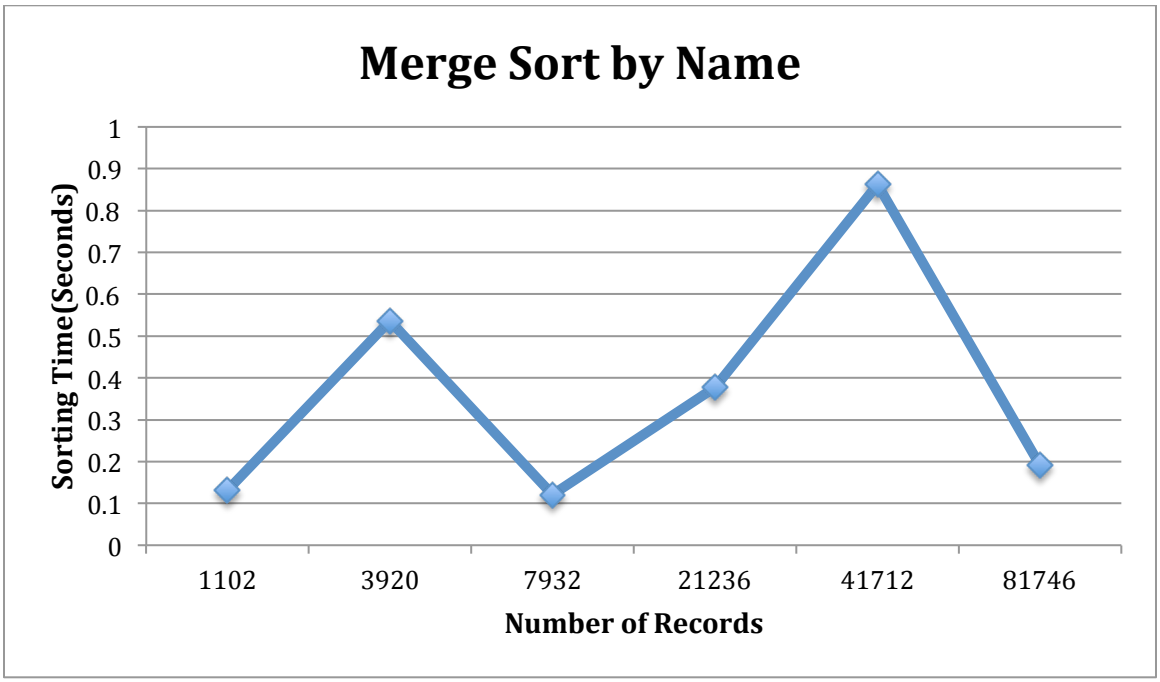
2. Graph of Insertion Sort (By Name)



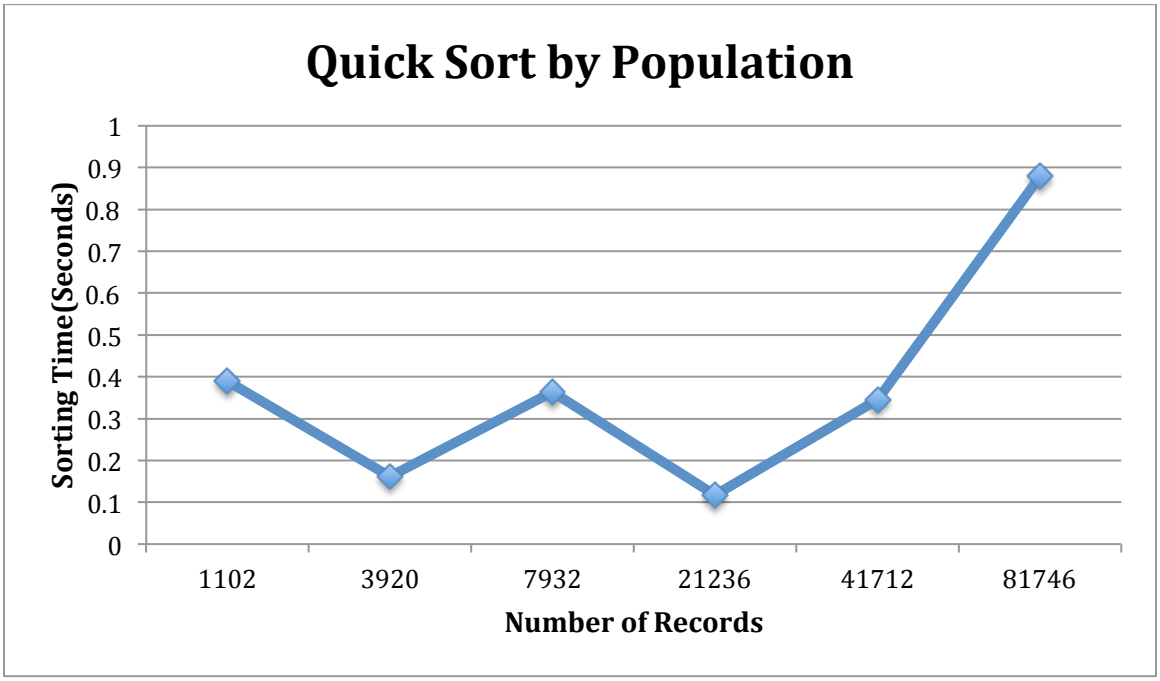
3. Graph of Merge Sort (By Population)



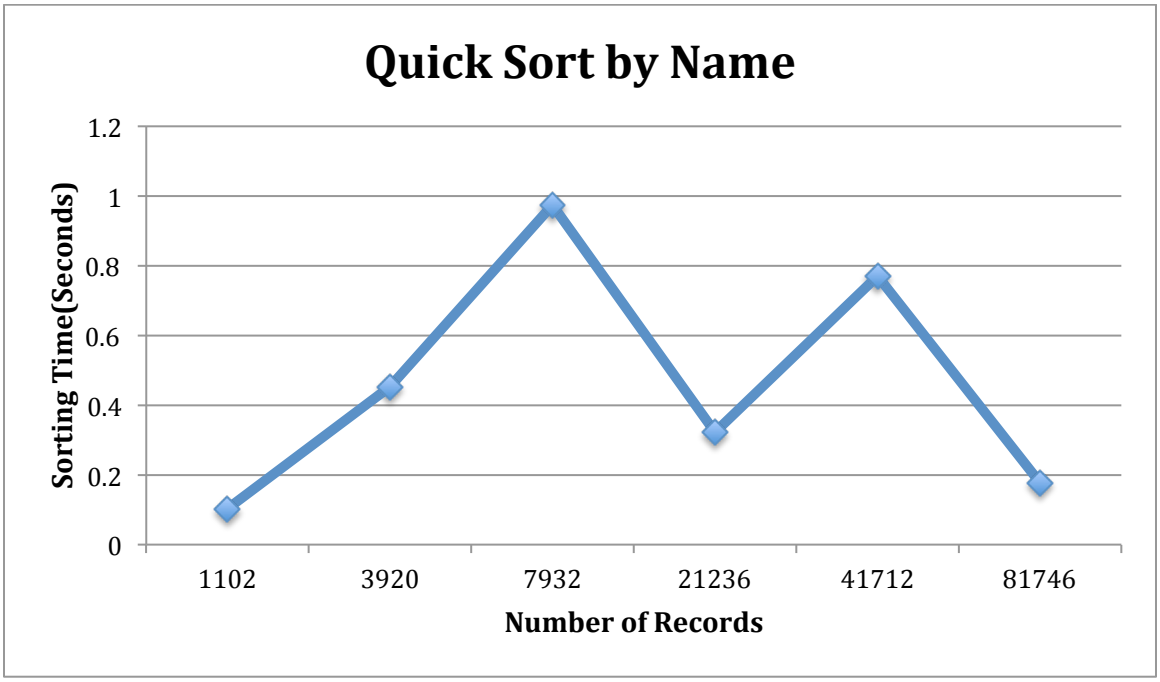
4. Graph of Merge Sort (By Name)



5. Graph of Quick Sort (By Population)



6. Graph of Quick Sort (By Name)



From the tables and graphs above we can realize that the more the number of records are the more time it takes to sort these records. Insertion Sort for sorting by city name takes the maximum sorting time, as the number of records increases. Also the sorting time for sorting by population for insertion sort takes more time then the other sorting algorithms. This shows us that insertion sort is a very efficient algorithm if the number of records to sort is small. That is Insertion sort works efficiently on small data sets.

Merge Sort works the same as per the sorting time required for sorting by population and city name. Merge sort takes average running time to run on small as well as large data sets. It is much efficient then Insertion Sort on large data sets.

Quick Sort works very efficiently on small as well as large data sets. It takes the least amount on time to perform sorting on large datasets. Quick sort by City name performs the best amongst the three sorting algorithms on large datasets.

In conclusion, Quicksort performs the best on large data sets whereas Insertion Sort performs the worst on large data sets. Quick Sort is the best sorting algorithm amongst the three sorting algorithms.