Report

<u>DATA:</u>

<u>Running on [n, n-1, … , 1]</u>

Selection Sort

n	**Comparisons**
100	1597
200	5376
400	15582
800	33391
1600	73647
3200	154834
6400	306263

Insertion Sort

n	**Comparisons**
100	2306
200	9585
400	39337
800	155119
1600	636177
3200	2510318
6400	10230071

Merge Sort

n	**Comparisons**
100	536
200	1265
400	2925
800	6716
1600	15046
3200	33228
6400	72870

Quick Sort

n	**Comparisons**
100	199
200	449
400	1003
800	2341
1600	5477
3200	12385
6400	27621

Running on [1, 2, … n]

Selection Sort

n	**Comparisons**
100	1,954
200	5,609
400	14,548
800	34,732
1600	71,099
3200	155,168
6400	314,150

Insertion Sort

n	**Comparisons**
100	2,301
200	9,691
400	39,383
800	153,023
1600	636,832
3200	2,546,395
6400	10,039,836

Merge Sort

n	**Comparisons**
100	540
200	1,290
400	2,949
800	6,699
1600	15,046
3200	33,231
6400	72,879

Quick Sort

n	**Comparisons**
100	187
200	436
400	1,021
800	2,376
1600	5,408
3200	1,2297
6400	27,543

Averages

Selection Sort

n	**Average**
100	1,597
200	5,376
400	15,582
800	33,391
1600	73,647
3200	154,834
6400	306,263

Insertion Sort

n	**Average**
100	2,306
200	9,585
400	39,337
800	155,119
1600	636,177
3200	2,510,318
6400	10,230,071

Merge Sort

n	**Average**
100	536
200	1,265
400	2,925
800	6,716
1600	15,046
3200	33,228
6400	72,870

Quick Sort

n	**Average**
100	199
200	449
400	1,003
800	2,341
1600	5,477
3200	12,385
6400	27,621

```
Selection Sort Averages
Size of Sequence [100], Average comparisons: 1762.
Size of Sequence [200], Average comparisons: 5621.
Size of Sequence [400], Average comparisons: 14795.
Size of Sequence [800], Average comparisons: 34456.
Size of Sequence [1600], Average comparisons: 73910.
Size of Sequence [3200], Average comparisons: 152930.
Size of Sequence [6400], Average comparisons: 312036.

Insertion Sort Averages
Size of Sequence [100], Average comparisons: 2373.
Size of Sequence [200], Average comparisons: 9984.
Size of Sequence [400], Average comparisons: 38819.
Size of Sequence [800], Average comparisons: 160032.
Size of Sequence [1600], Average comparisons: 630731.
Size of Sequence [3200], Average comparisons: 2547893.
Size of Sequence [6400], Average comparisons: 10123667.

Merge Sort Averages
Size of Sequence [100], Average comparisons: 540.
Size of Sequence [200], Average comparisons: 1277.
Size of Sequence [400], Average comparisons: 2968.
Size of Sequence [800], Average comparisons: 6706.
Size of Sequence [1600], Average comparisons: 15020.
Size of Sequence [3200], Average comparisons: 33243.
Size of Sequence [6400], Average comparisons: 72856.

Quick Sort Averages
Size of Sequence [100], Average comparisons: 191.
Size of Sequence [200], Average comparisons: 446.
Size of Sequence [400], Average comparisons: 1056.
Size of Sequence [800], Average comparisons: 2388.
Size of Sequence [1600], Average comparisons: 5407.
Size of Sequence [3200], Average comparisons: 12323.
Size of Sequence [6400], Average comparisons: 27303.
```

Discussion:

When the sequences[n, n-1, … 1] and [1, 2, …, n] were recorded I found that insertion sort and selection sort did much better then O(n^2), however it was a very obvious that insertion sort and selection sort took much more time/comparisons. When computing sequences of size 6400 and using insertion/selection sort it often took minutes compared to merge/quick sort working in seconds. When it comes to merge/quick sort, sorts that run in linear time (O(n)), the comparisons made were more than n. The fasted sort was by far quicksort running even sequences of 6400 in seconds, merge sort was definitely an improve upon insertion/selection sort but still having more comparisons the quicksort. Selection sort took significant comparison often at least 10 X n, however insertion sort was clearly the least efficient especially when it came to large sizes.

The averages for each sort did not agree with our analysis in class. The sorting algorithm that was closest to its analysis in class was Quick Sort that on average runs in O(n) but quick sort still often took at least 2 X n comparisons, however merge sort also was much more then O(n) and runs in that time. The other two algorithms ran significantly faster than the average analysis of O(n^2).