**Methodology**:
I generated the arrays by first picking the four different sizes I wanted to test. I settled with 20, 35, 50, and 100 since the quadratic sorting algorithms are best used for smaller data sets, but I also wanted to with large sizes that you can still reasonably visualize. Then I would let a function input random numbers into the arrays that will then get sorted via the three sorting algorithms. I will then take notes of each trial to calculate the average runtime for each algorithm.

**Results**:
Bubble sort:

| Array size | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Average comparisons |
|---|---|---|---|---|---|---|
| 20 | 107 | 115 | 118 | 135 | 90 | 113 |
| 35 | 289 | 335 | 273 | 316 | 319 | 306.4 |
| 50 | 642 | 661 | 765 | 664 | 655 | 677.4 |
| 100 | 2671 | 2568 | 2586 | 2781 | 2707 | 2662.6 |

Selection sort:

| Array size | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Average comparisons |
|---|---|---|---|---|---|---|
| 20 | 190 | 190 | 190 | 190 | 190 | 190 |
| 35 | 595 | 595 | 595 | 595 | 595 | 595 |
| 50 | 1225 | 1225 | 1225 | 1225 | 1225 | 1225 |
| 100 | 4950 | 4950 | 4950 | 4950 | 4950 | 4950 |

Insertion sort:

| Array size | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Average comparisons |
|---|---|---|---|---|---|---|
| 20 | 105 | 115 | 118 | 131 | 88 | 111.4 |
| 35 | 288 | 330 | 272 | 312 | 315 | 303.4 |
| 50 | 637 | 661 | 759 | 660 | 652 | 673.8 |
| 100 | 2669 | 2563 | 2584 | 2779 | 2699 | 2658.8 |

**Analysis**:

- No matter how elements are arranged, selection sort consistently makes the same number of comparisons for each array size. Selection sort makes about $(n^2/2)$ comparisons.
- Bubble sort and insertion sort tend to make about $(n^2/4)$ comparisons, although insertion sort is slightly faster on average.
- While selection sort on average makes about twice the number of comparisons as bubble and insertion sort, in general, all the quadratic sorting algorithms tend to match the Big O growth of $O(n^2)$.