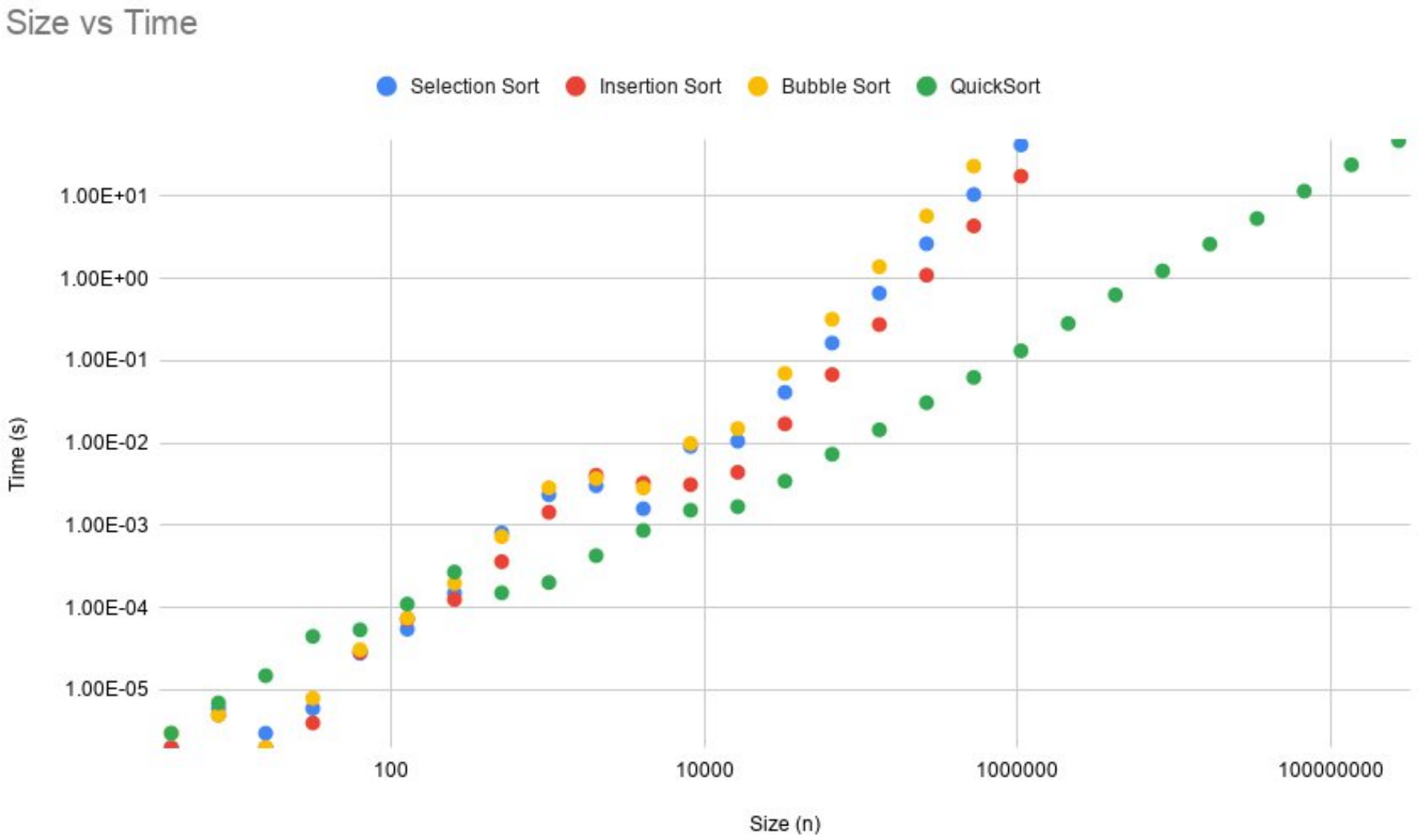
Assignment 01 - All Sorts of Sorts

Programming II / Tristan Goodell

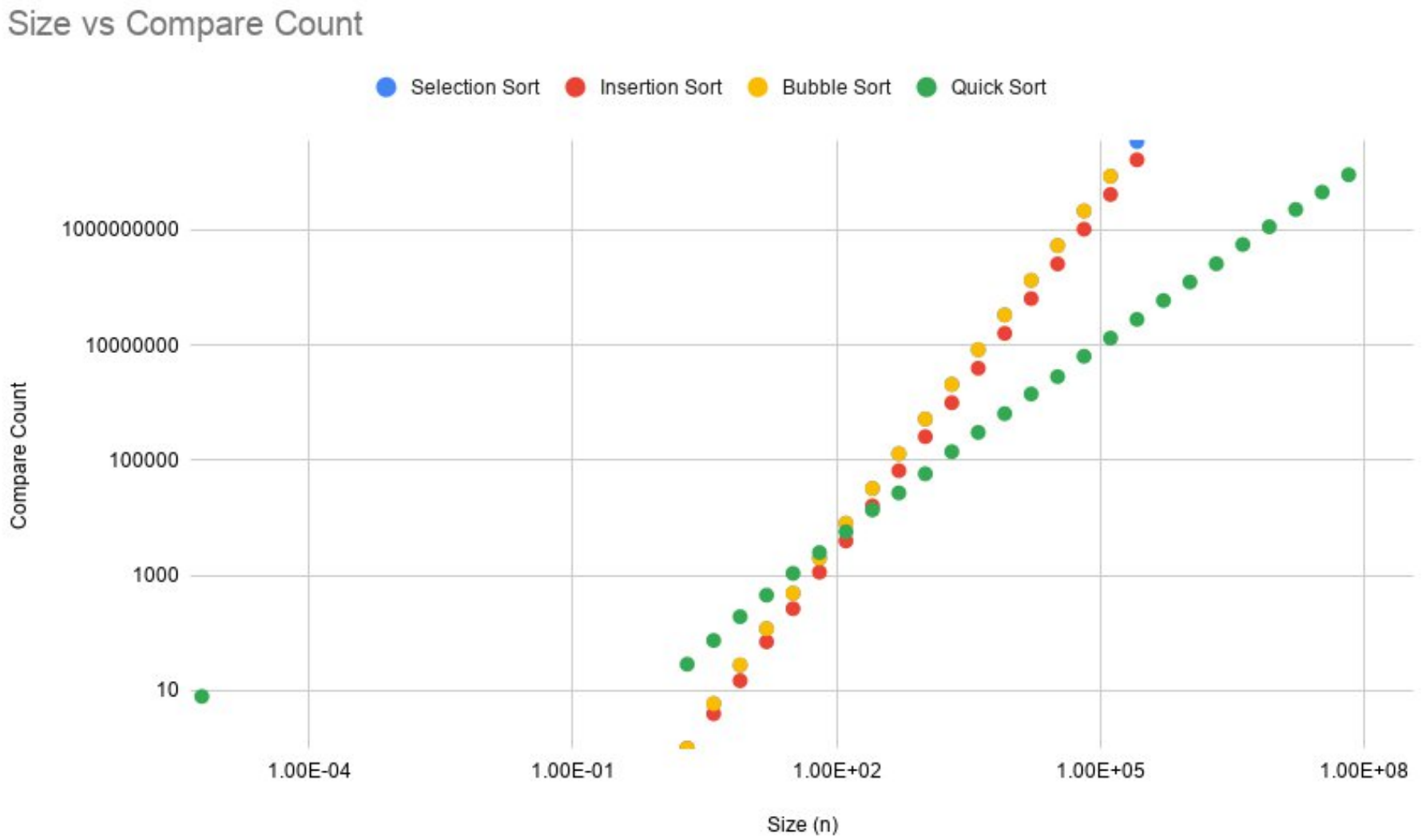
Time


Size vs Time
Selection Sort
Insertion Sort
Bubble Sort
QuickSort
1.00E+01
1.00E+00
1.00E-01
1.00E-02
1.00E-03
1.00E-04
1.00E-05
Time (s)
100
10000
1000000
100000000
Size (n)

Time / Discussion

- Although QuickSort is slower than the other sorts for lists smaller than $n=32$, it outpaces the other sorting techniques for large lists.
- Since QuickSort is faster than the other techniques, it can process lists up to size $n=268435456$ in less than a minute, compared to the other lists only being able to process lists up to size $n=262144$ in less than a minute.

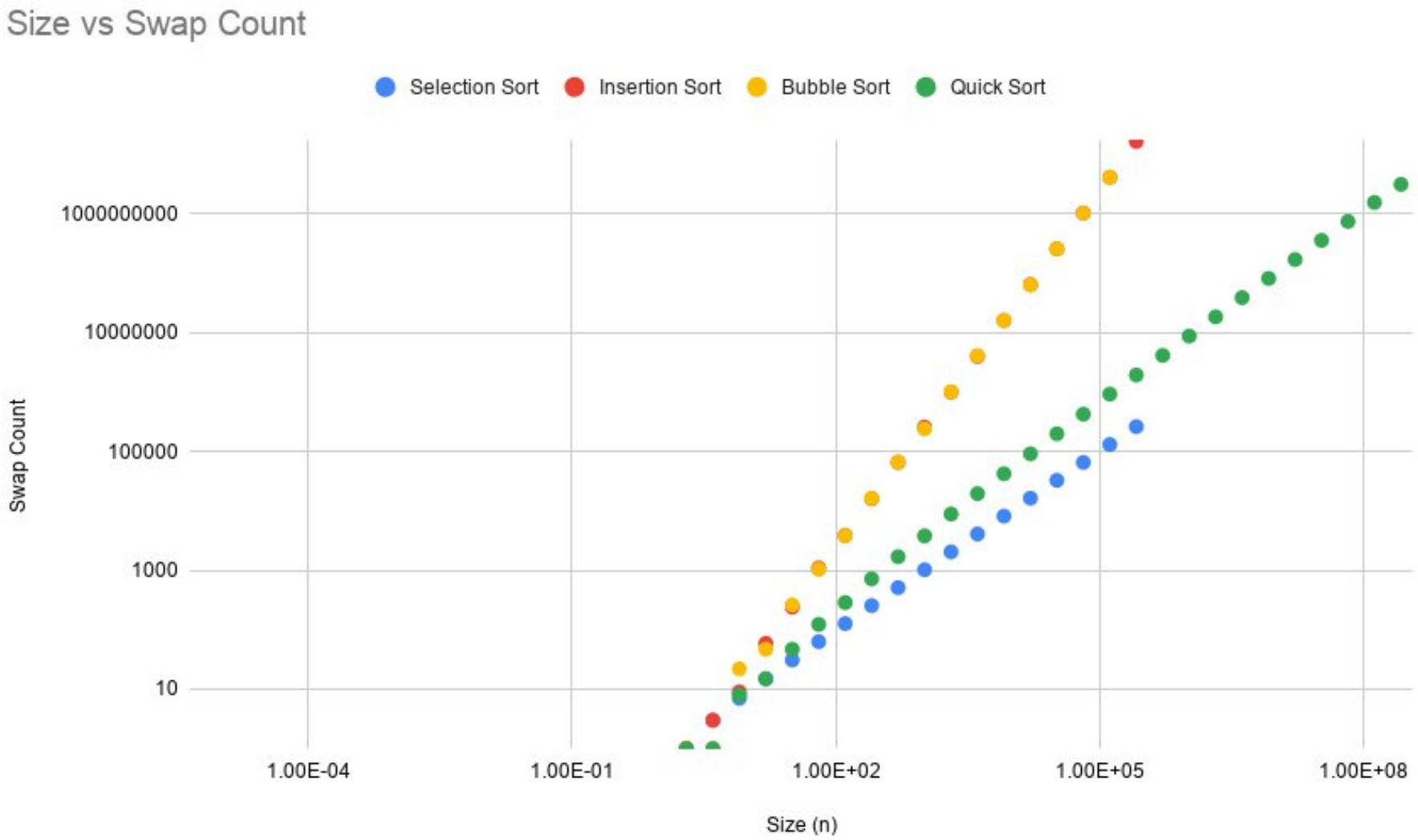
Compare Count



Compare Count / Discussion

- Once again, Quick Sort is slower than the other sorting techniques with lists with a length less than n=32. After that, Quick Sort becomes much more efficient and requires fewer compares to successfully sort.
- The number of Compare Counts for Insertion, Selection, & Bubble sorting is incredibly similar. This is probably explained by all three sorts comparing values just once per loop.

Swap Count


Size vs Swap Count
Selection Sort
Insertion Sort
Bubble Sort
Quick Sort
1000000000
10000000
100000
1000
10
Swap Count
1.00E-04
1.00E-01
1.00E+02
1.00E+05
1.00E+08
Size (n)

Swap Count / Discussion

- Surprisingly, Selection Sort outpaces Quick Sort with the fewest swaps for all lists size n>2.
- Insertion and Bubble Sort swap list items approximately the same number of times.
- It took approximately the same number of swaps for Quick Sort to sort a list $n=268435456$ as it took Insertion and Bubble Sort to sort a list $n=262144$.

Bubble Sort / Code

```
public void bubbleSort()
{
        for(int i=0;i<size;i++)
        {
                for(int j=1;j<=size-1-i;j++)
                {
                        if(!inOrder(get(j),get(j-1)))swap(j-1,j);
                }
        }
}
```

Insertion Sort / Code

```
public void insertionSort()
{
        for(int i=1;i<=size-1;i++)
        {
                int j=i;
                while(j>0 && inOrder(get(j-1),get(j)))
                {
                        swap(j,j-1);
                        j--;
                }

        }
}
```

Selection Sort / Code

```
public void selectionSort()
{
        for (int i=0;i<size-1;i++)
        {
                int minimum = i;
                for(int x=i+1;x<size;x++)
                {
                        if(inOrder(get(x),get(minimum)))
                        {
                                minimum=x;
                        }
                }
                swap(minimum,i);
        }
}
```

Quick Sort / Code

```
public int partition(int lo, int hi)
{
        int pivot=get(lo);
        int i=lo;
        int j=hi;
        while (true)
        {
                while(get(i)<pivot)
                {
                        i++;
                        compareCount++;
                }
                while(get(j)>pivot)
                {
                        j--;
                        compareCount++;
                }
                if(i>=j)
                {
                        compareCount++;
                        return j;
                }
                swap(i,j);
        }
}
```

```
public void quickSort()
{
        quickerSort(0,size-1);
}

public void quickerSort(int lo, int hi)
{
        if (lo>=hi)
        {
                return;
        }
        int pivot=partition(lo, hi);
        quickerSort(lo, pivot);
        quickerSort(pivot+1, hi);
}
```