Comp3506hw2

Mohammad Faiz Ather

September 2020

1 Introduction

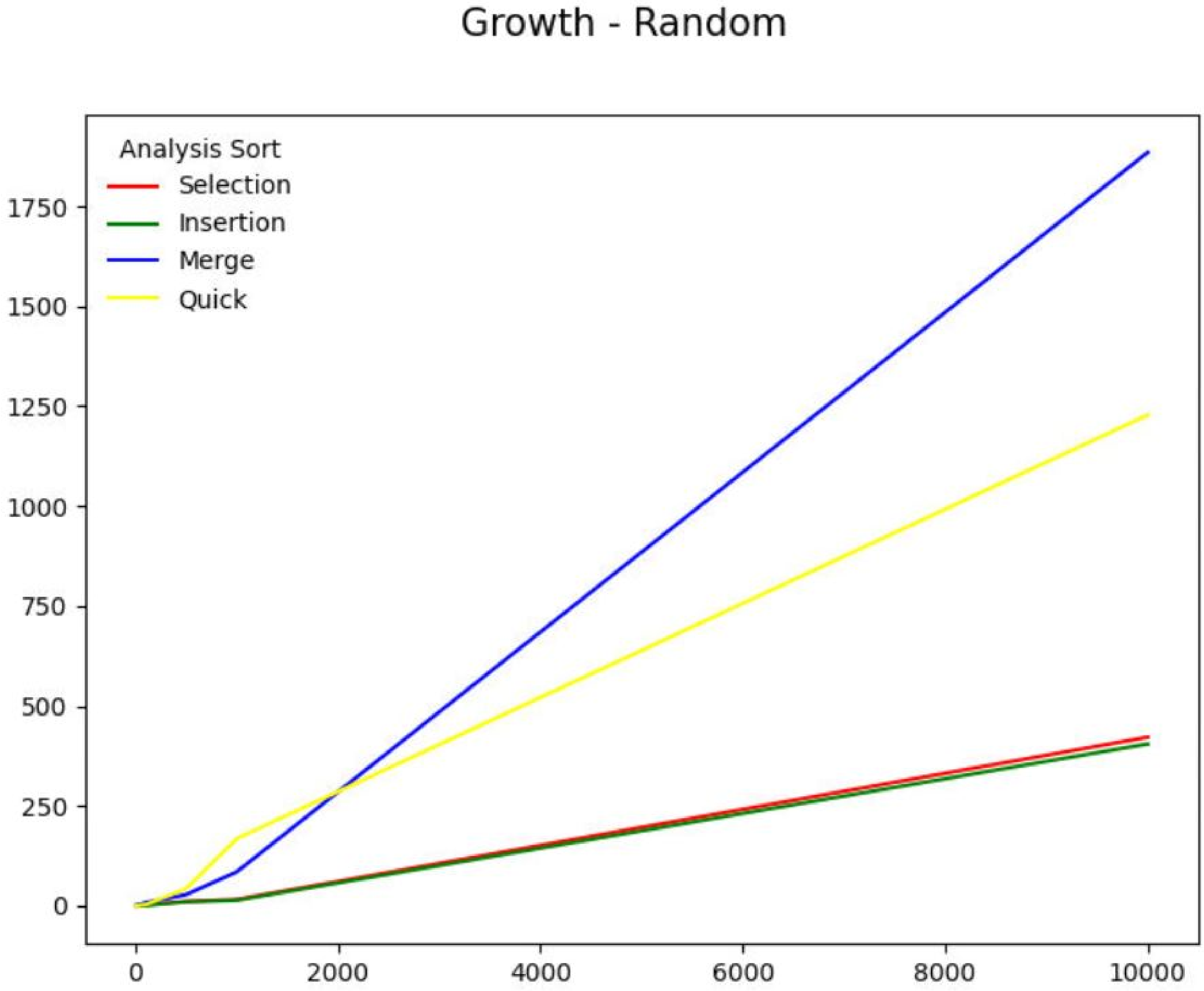
1. Merge Sort is $nlog(n)$ so for max $10,000 * log(10,000)$
2. Quick Sort for worst is n^2 average is $nlog(n)$.
3. Selection Sort is n^2
4. Insertion sort best n and worst n^2

   time.java

```java
public static void timeSort() {
    Random rd = new Random();
    creating Random objectInteger[] sizes = {5, 10, 50, 100, 500, 1000, 10000;
    for (int s : sizes) {
        Integer[] arr = new Integer[s];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = rd.nextInt();// storing random integers in an array
        }
        //SortingAlgorithms.selectionSort(arr, true);
        long start = System.currentTimeMillis();
        SortingAlgorithms.quickSort(arr, false);
        long end = System.currentTimeMillis();
        System.out.println("size of " + s + " for q " +(end - start) + "ms");
    }
}
```

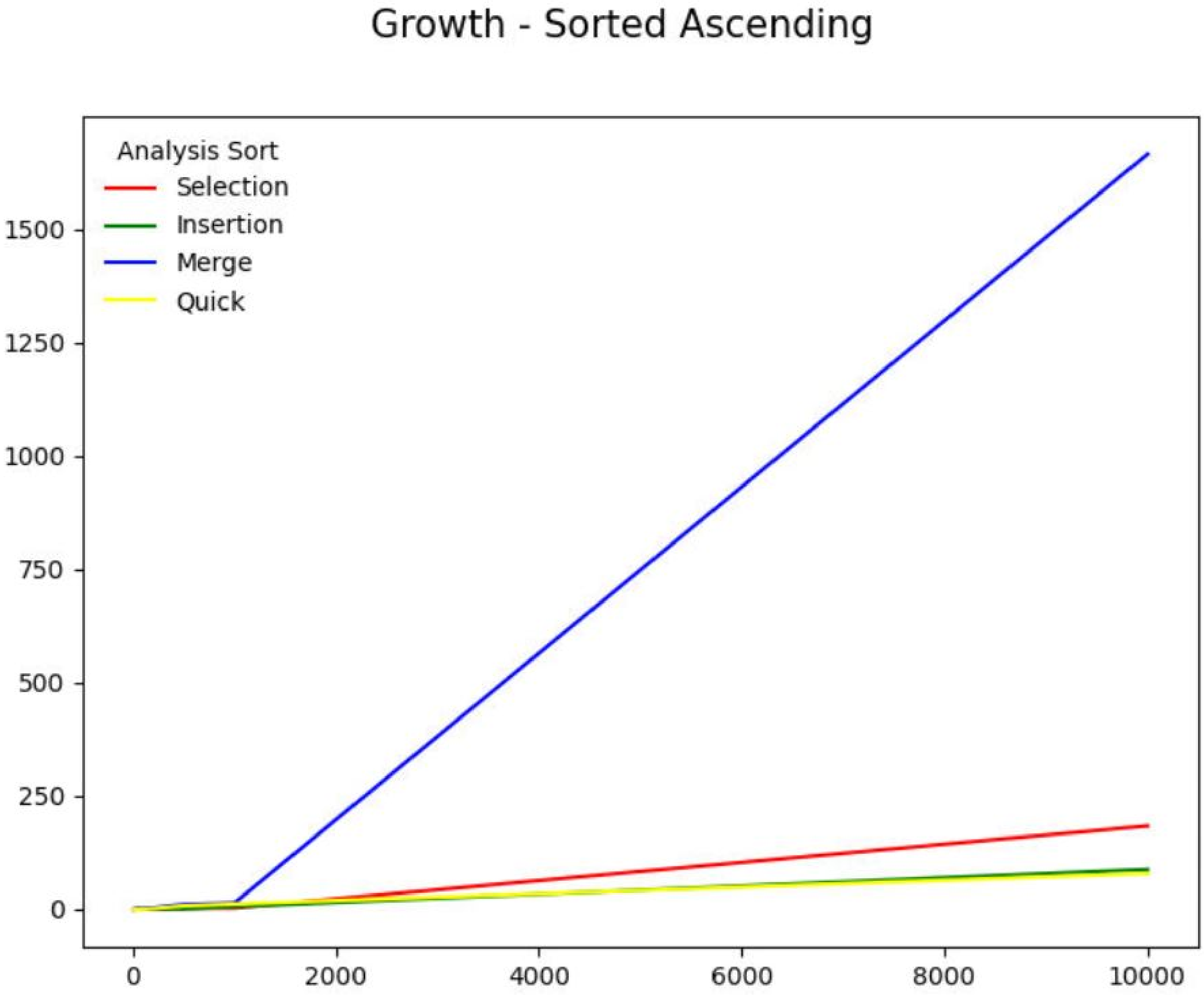
   Maybe if the number of items is increased by a lot to 50,000 or 100,000 and more then we will see merge sort and quick sort out perform the iterative algorithms

2 Graph 1 - Random



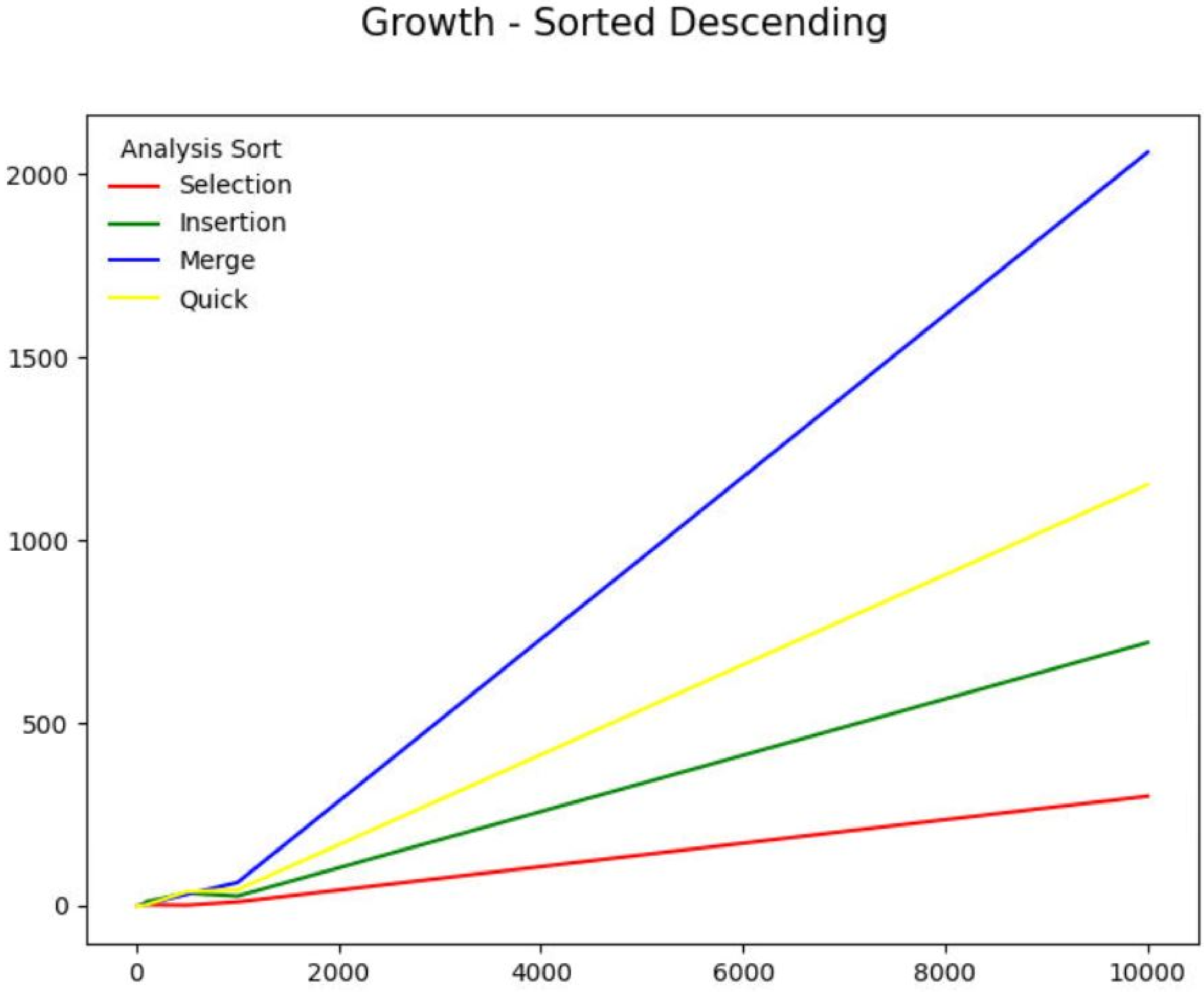
The recursive algorithms grow more compared to the iterative ones, perhaps because of the function stack overhead for the JVM to maintain.

3 Graph 2 - Ascending



Program is not going inside the loops because hence faster. Already sorted so matches expectations but again merge sort just shoots up at n = 10,000.

4 Graph 3 - Descending



The same trend is observed as Random. I expected Merge Sort to perform better and I am almost certain my implementation of mergesort is not correct I will have to check the code again.