CS361 Algorithm Lab 2

What to do

1. Implement the Radix sort algorithm and use it to sort roughly 10,000,000 numbers. I am providing a new data file (on Moodle).

```
/**
 * The main function that sorts the array passed in to index n using Radix Sort
 *
 * @param arrR The array that is being passed in and that you want sorted.
 * @param n The index to which you want to sort to.
 */
public void radixsort(int[] arrR, int n)
{

    int m = getMax(arrR, n);                          // Find the maximum number in order to know number of digits needed.

    // Do counting sort for every digit. Note that instead
    // of passing digit number, dig is passed. dig is 10^i
    // where i is current digit number
    for (int dig = 1; m/dig > 0; dig *= 10){
        countSort(arrR, n, dig);
        }
}

/* Much of the below code was helped and found on geeks for geeks https://www.geeksforgeeks.org/merge-sort/*/
```

Below is the counting sort alg. That was used for my radix sort.

```
/**
 * A function that uses counting sort to sort the array passed into arrC
 *
 * @param arrC An array of integers.
 * @param n The length in the array to which you would like to go to find a max value.
 * The max param for this value will be restricted to is the length of the array.
 * @param dig The digit that countSort will use for the sorting.
 */
private void countSort(int[] arrC, int n, int dig){
    int output[] = new int[n];                        // init. that output array to length n.
    int i;                                            // An index init. for the for loops.
    int count[] = new int[10];                        // init. the count array for base ten digits.
    Arrays.fill(count,0);                             // Fill the count array all zeros.

    for (i = 0; i < n; i++)                           // Store the count of occurrences in count[]
        count[ (arrC[i]/dig)%10 ]++;

    for (i = 1; i < 10; i++)                          // Change the values of count at index i so that it reflex
        count[i] += count[i - 1];                     // how many digits that are less than or equal to that digit.


    for (i = n - 1; i >= 0; i--)                      // Build the output array with the values of arrC
    {                                                 // divided by the digit we are using as an index to count[] that
        output[count[ (arrC[i]/dig)%10 ] - 1] = arrC[i];// will be used as an index for the output array that we will set to the arrC of index i.
        count[ (arrC[i]/dig)%10 ]--;                  // Decrement what the integer in count array at the index we used for the out put array.
    }



    for (i = 0; i < n; i++){                          // Copy the output array to arrC[], so that arr[] now
        arrC[i] = output[i];                          // contains sorted numbers according to current digit
        }
}
```

2. Implement the Bin sort algorithm and use it to sort roughly 10,000,000 numbers.

```
/* The bin sort algorithm was written with the help of looking at the code from
 * https://github.com/skoliver89/CS361-Lab3/blob/master/Lab3.java yet it was
 * modified by myself in order to fit other function and requierments to this lab.
 * All comments and documentation were add to show understanding and to clearify the
 * code and functionality.*/
```

*CS361Labs.java

```java
    /**
     * A signal method that implements bin sort algorithm.
     *
     * @param arrB An int. array that the bin sort alg. will be applied to.
     * @param lengthTo The length to which you want to apply the sort to.
     */
    public void binSort(int[] arrB, int lengthTo){
        int n = getMax(arrB, lengthTo);                 // Get the maximum value upto the index you want to go to.
        int[] bin = new int[n+1];                       // Set the bin array to be one greater than the max value that was just found.
        int i;// this is just an indexer for the next few loops.

        for(i=0;i<=n;i++){
            bin[i] = 0;                                 // init. the bins to zero.
        }

        for(i=0;i<=lengthTo - 1;i++){
            bin[arrB[i]]++;                             // Using increment the the index of bins for every value that is in arrB
        }                                               // up to the index that we are inspecting in arrB.

        int outIndex = 0;
        for(i=0;i<=n;i++){
            for(int j=0;j<bin[i];j++){                  // Go through the bins and find out how many numbers are in it.
                arrB[outIndex]=i;                       // Put the numbers from bin back into the array.
                outIndex++;
            }
        }
    }

    /* Much of the below code was helped and found on geeks for geeks https://www.geeksforgeeks.org/radix-sort/*/
```

Below is the method that allows me to retrieve the max value and us it in my bin sort.

```java
    /**
     * A get method to return the maximum number held in the array
     * pasted.
     *
     * @return The integer with the highest value in the array past.
     * @param arrR An array of integers.
     * @param n The length in the array to which you would like to go to find a max value.
     * The max param for this value will be restricted to is the length of the array.
     */
    private int getMax(int[] arrR, int n){
        int mx = arrR[0];
        for (int i = 1; i < n; i++)
            if (arrR[i] > mx)
                mx = arrR[i];
        return mx;
    }

    /**
```

3. Make sure the results are sorted for 1 and 2. Show the screen dump indicate the sorting algorithms are actually sorting correctly.

```java
        /************************************************************ RADIX SORT *

        int x = 1;
        for(int y = 1000; y <= arr.length; y  = y * 10){
        long radixSortTime = System.nanoTime();
        lab2RadixSort.radixsort(arr,y);
            System.out.println("Radix sort ran " + x + ": " + (System.nanoTime(
            x++;
        }

        if(lab2RadixSort.flgIsSorted(arr)){
            System.out.println("The array was sorted using radix sort.");
        }else{
            System.out.println("It didn't work.");
        }

        /************************************************************ RADIX SORT *

        /************************************************************ QUICK SORT *
```

Problems Javadoc Declaration Console

```
<terminated> CS361Labs [Java Application] C:\Program Files\Java\jdk1.8.0_131\bin\javaw.exe (May 3, 2018, 5:27:30
Radix sort ran 1: 651328
Radix sort ran 2: 3029337
Radix sort ran 3: 16623786
Radix sort ran 4: 101228499
Radix sort ran 5: 1090385866
The array was sorted using radix sort.
```

As we can see the flgIsSorted() method is use in an if statement so that the statement "The array was sorted using radix sort." Will print to the console only if the array is sorted. Otherwise the statement "It didn't work." Prints to the console. As we can see the proper statement is printed.

```
361         /*********************************************************  BIN SORT  ******************
362
363
364         int x = 1;
365         for(int y = 1000; y <= arr.length; y  = y * 10){
366         long binSortTime = System.nanoTime();
367         lab2BinSort.binSort(arr,y);
368             System.out.println("Bin sort ran " + x + ": " + (System.nanoTime() - binSortTime));
369             x++;
370         }
371
372         if(lab2BinSort.flgIsSorted(arr)){
373             System.out.println("The array was sorted using bin sort.");
374         }else{
375             System.out.println("It didn't work.");
376         }
377
378
379         /*********************************************************  BIN SORT  ******************
```

Problems Javadoc Declaration Console

```
<terminated> CS361Labs [Java Application] C:\Program Files\Java\jdk1.8.0_131\bin\javaw.exe (May 3, 2018, 5:35:33 AM)
Bin sort ran 1: 26292141
Bin sort ran 2: 22545399
Bin sort ran 3: 27129239
Bin sort ran 4: 46832581
Bin sort ran 5: 234057560
The array was sorted using bin sort.
```

As we can see the flgIsSorted() method is use in an if statement so that the statement "The array was sorted using bin sort." Will print to the console only if the array is sorted. Otherwise the statement "It didn't work." Prints to the console. As we can see the proper statement is printed.

4. Show the execution time comparison with your either quick sort or merge sort. Also make sure the result of your quick sort or merge sort is sorted.

1	First Run	Bin sort	Radix sort	Quick sort
2	1 to 1000	32450497	637358	454231
3	1 to 10000	28636922	3018010	1775767
4	1 to 100000	34032187	15717967	12404311
5	1 to 1000000	53449699	103563842	94535488
6	1 to 10000000	182235238	1047301160	2090318656
7				
8	Second Run	Bin sort	Radix sort	Quick sort
9	1 to 1000	16402524	635470	635847
10	1 to 10000	22839157	3062565	1641725
11	1 to 100000	26219268	15649625	10107104
12	1 to 1000000	45718338	100030433	95174356
13	1 to 10000000	172285207	1044499882	2081354491
14				
15	Third Run	Bin sort	Radix sort	Quick sort
16	1 to 1000	26629699	656236	478774
17	1 to 10000	22783652	3065208	1725170
18	1 to 100000	26044825	13571416	9882821
19	1 to 1000000	44671681	101422577	94384078
20	1 to 10000000	173543309	1045383423	2073495885

```
401         int x = 1;
402         for(int y = 1000; y <= arr.length; y  = y * 10){
403         long quickSortTime = System.nanoTime();
404         lab2QuickSort.auxQuickSort(arr, 0, y - 1);
405             System.out.println("Quick sort ran " + x + ": " + (System.na
406             x++;
407         }
408         if(lab2QuickSort.flgIsSorted(arr)){
409             System.out.println("The array was sorted using quick sort.")
410         }else{
411             System.out.println("It didn't work.");
412         }
413
414         /******************************************************* QUICK
415
416         /*************************************************** MERGE SO
417         /*
```

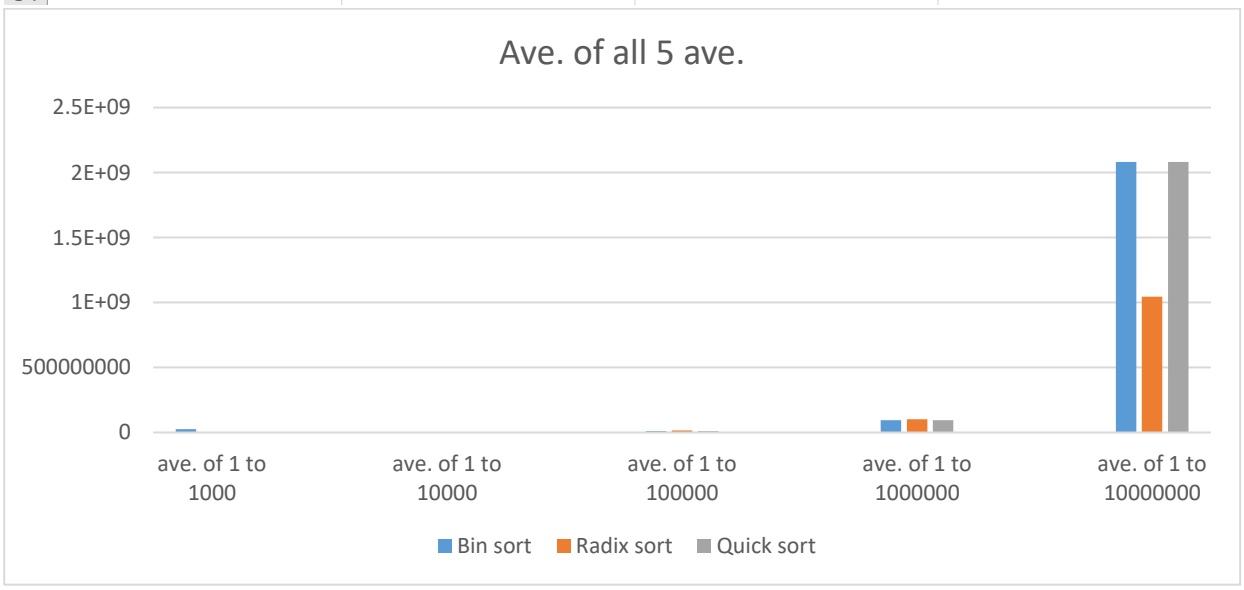
Problems Javadoc Declaration Console

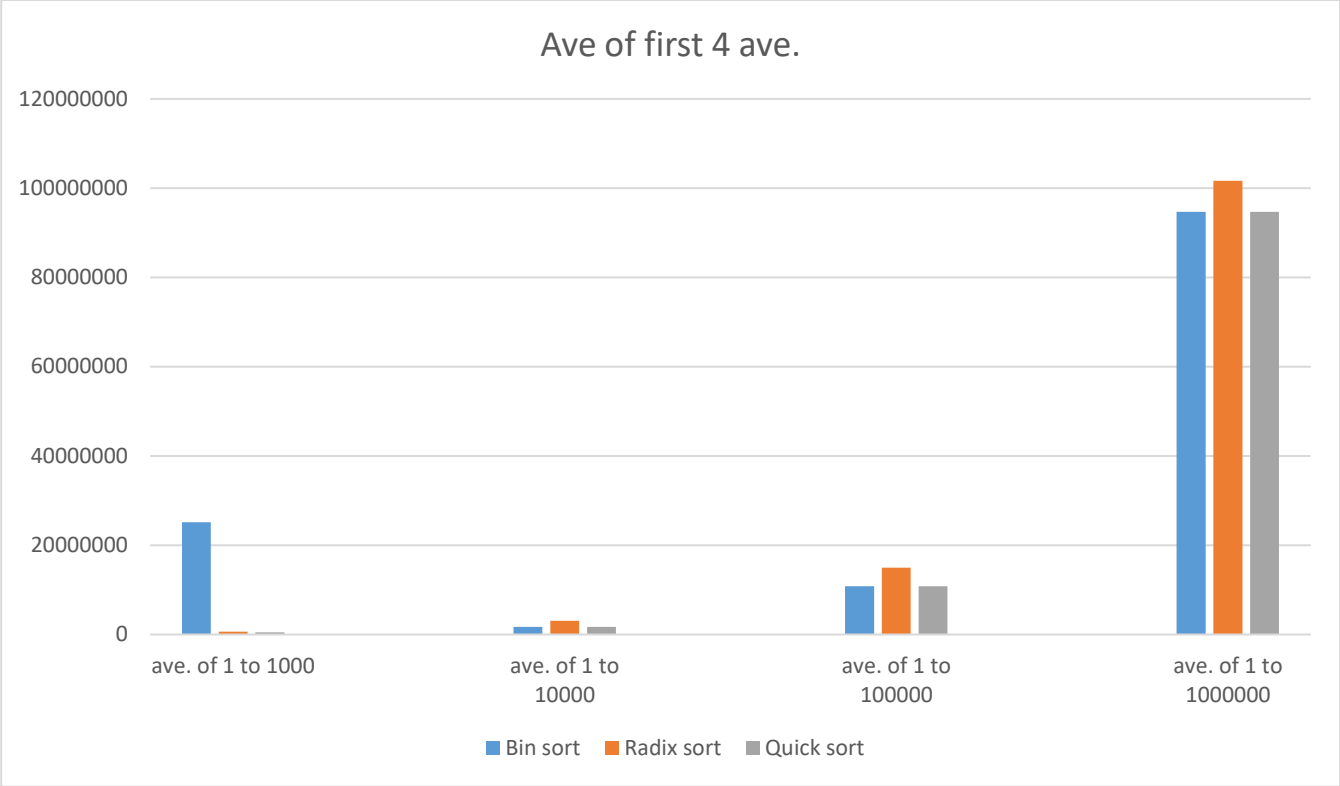
```
<terminated> CS361Labs [Java Application] C:\Program Files\Java\jdk1.8.0_131\bin\javaw.exe (May 3, 2018
Quick sort ran 1: 592426
Quick sort ran 2: 2151082
Quick sort ran 3: 11799048
Quick sort ran 4: 94921754
Quick sort ran 5: 2082709631
The array was sorted using quick sort.
```

As we can see the flgIsSorted() method is use in an if statement so that the statement "The array was sorted using quick sort." Will print to the console only if the array is sorted. Otherwise the statement "It didn't work." Prints to the console. As we can see the proper statement is printed.

5. Run your code for 1~3 three times, record the execution time in milliseconds for each run on each size, enter the milliseconds reading into an Excel spreadsheet, calculate the average execution time in milliseconds and provide your results in a table and/or as a line chart.

23				
24		Bin sort	Radix sort	Quick sort
25	ave. of 1 to 1000	25160906.67	643021.3333	522950.6667
26				
27	ave. of 1 to 10000	1714220.667	3048594.333	1714220.667
28				
29	ave. of 1 to 100000	10798078.67	14979669.33	10798078.67
30				
31	ave. of 1 to 1000000	94697974	101672284	94697974
32				
33	ave. of 1 to 10000000	2081723011	1045728155	2081723011
34				





6. Use your Lab 1 read method to from my data file. Then write **recursive** algorithm to list the largest 10 elements of the data you read, and listing them in decreasing order as the output. Again, start with 1,000 and increases at 10x until it needs to read more than 10 million numbers. Output the execution time of your approach.

```
/**
 * This method uses recursion to find and sort the top ten values in the array
 * passed and stores them in the array topTen which is a field held in this class.
 *
 * @param ray The array that you want to find the top ten integers form.
 * @param y The starting index of the array that we want to look to.
 * @param n The ending index of the array that we want to look to.
 */
public void step6(int[] ray,int y,int n){

    int temp = getIndexOfMax(ray,0,n);      // Get the max value's index from the array
                                            // in the array give the interval 0 to n and store the value in temp.
    topTen[y]= ray[temp];                   // sore the max values of the array and store it in the field array topTen.
    ray[temp] = ray[n];                     // Take where the max number is set it in to what the last value in the array is.
    ray[n] = topTen[y];                     // Now take the last value in the array to what the max value is.
    if(y<9){                                // Only make the recursive call 10 times.
        step6(ray,y+1,n-1);                 // Recursive call.
    }

}
```

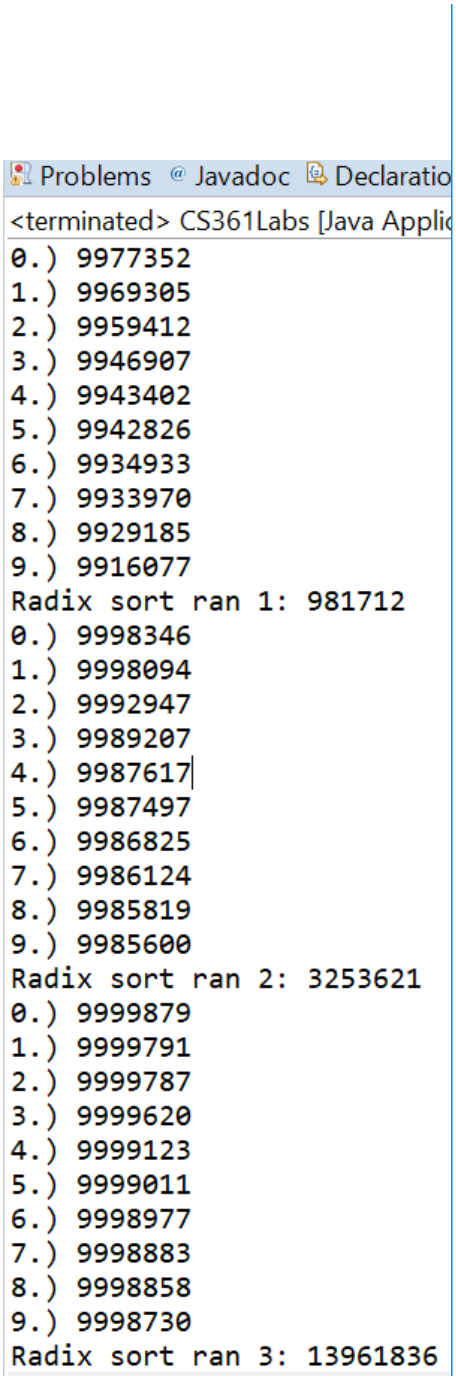
Above is the recursive method that I used to sort the top ten integers in descending order. Below is a helper method for the recursive method.

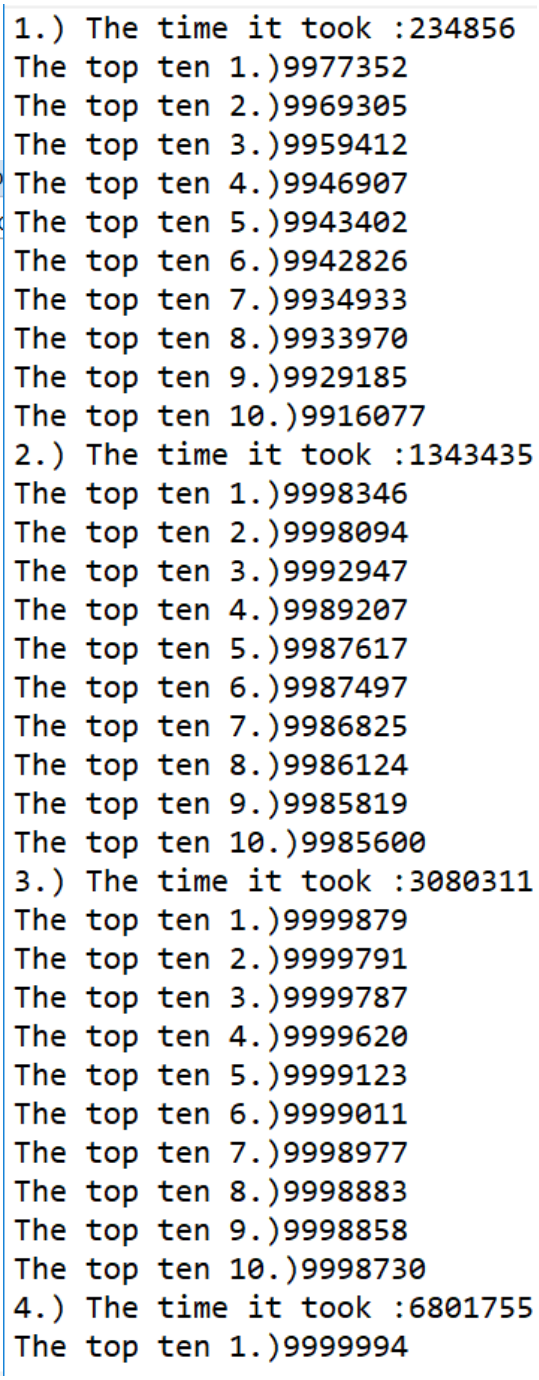
```
}
/**
 * The method will parse through the array and find the index of the max
 * value found in the array.
 *
 * @param arrRec the array that will be passed.
 * @param y The starting of the index of the array that we are looking at.
 * @param n The ending of the index of the array that we are looking at.
 * @return The index of the max value found in the array passed.
 */
public int getIndexOfMax(int[] arrRec, int y, int n) {
    int indexOfMax = 0;
    for (int i = y; i < n; i++)
        if (arrRec[i] > arrRec[indexOfMax]) {
            indexOfMax = i;
        }
    return indexOfMax;

}
```

```
454         /*****************************************************Recursive Alg.******************
455         int x = 1;
456         for(int y = 1000; y <= arr.length; y  = y * 10){
457             long recursiveTime = System.nanoTime();
458             lab2Recursive.step6(arr, 0, y - 1);
459             System.out.println(x + ".) The time it took :" + (System.nanoTime()-recursiveTime));
460             x++;
461             int w = 1;
462             for(int p:topTen){
463                 System.out.println("The top ten " + w + ".)" + p);
464                 w++;
465                 }
466             }
467
468         /*****************************************************Recursive Alg.******************
```

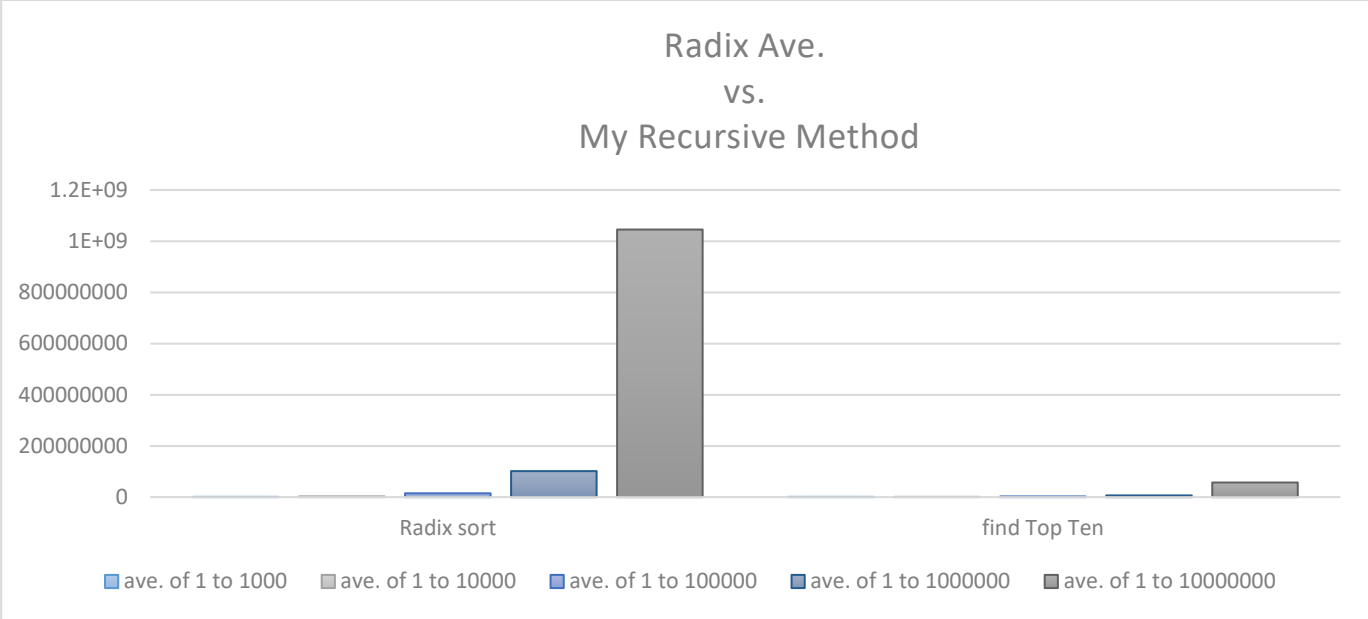
7. Test your result by calling one of your sorting algorithm to sort the data first and display largest numbers in decreasing order as the output. Output the execution time of your approach.



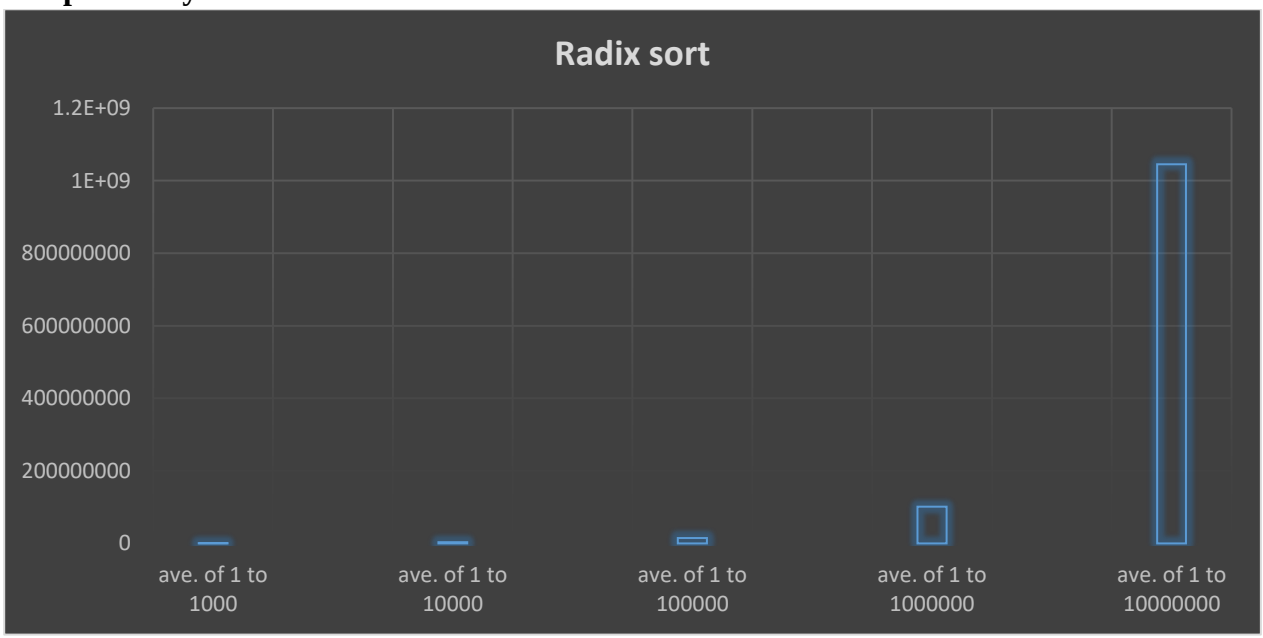


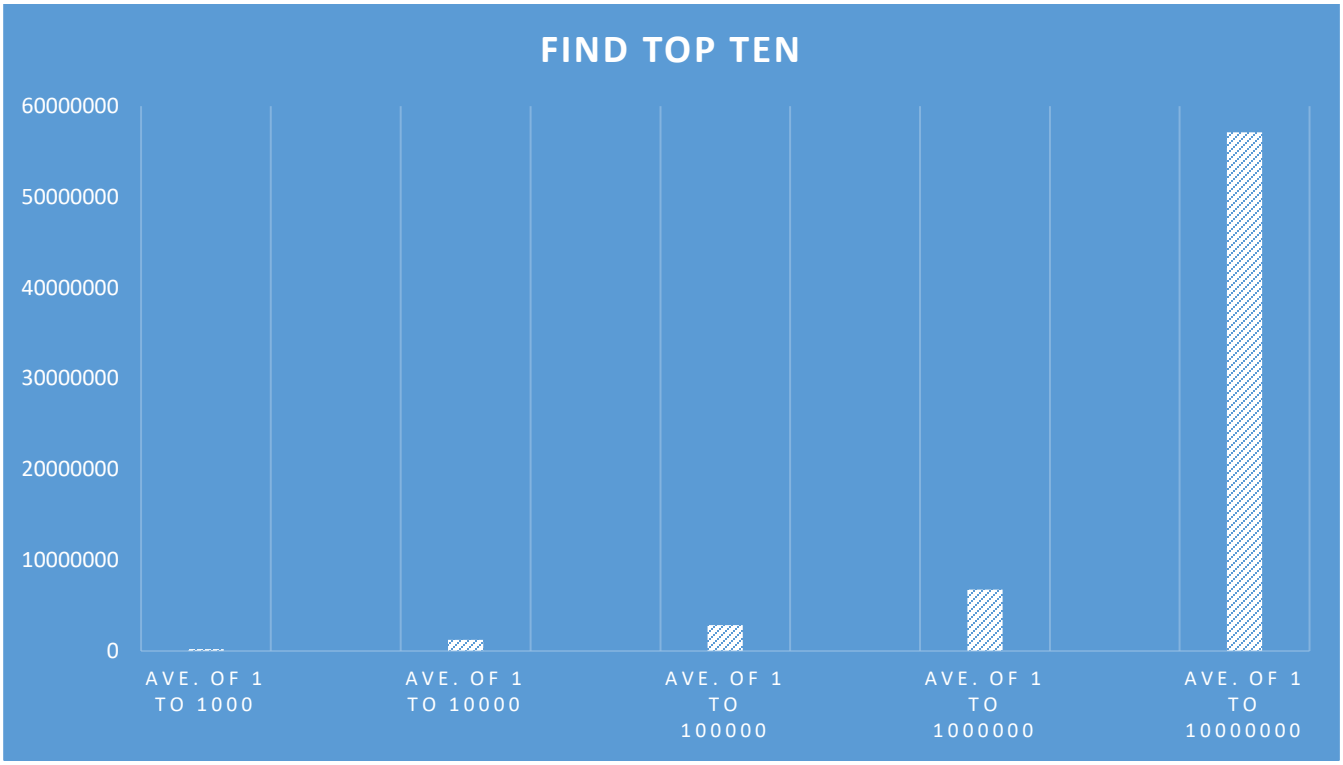
On the right is the print out of the recursive method that I wrote and executed to print out the top ten values in decreasing order. On the left is the same thing only radix sort is used to sort all of the integers in the array before printing the top ten values in decreasing order.

8. Run your code for part 6 and 7 three times, record the execution time in milliseconds for each run on each size, enter the milliseconds reading into an Excel spreadsheet, calculate the average execution time in milliseconds for each run on each size and display your results in both a table and as a line chart.



The above is them side by side. In order to have a better perspective I will show them independently.





9. Write a half to one-page report to explain your execution time observation and discuss the problem-solving approach you applied for step 6. Is it DP, greedy algorithm, or divide-and-conquer?

To solve the problem, I went through many executions and approaches that ended up with stack over flow errors, or I was solving the wrong problem as seen below in the commented-out code.

```
/**
 * The below is what I thought was what was wanted but it isn't.
 *
 * This is my recursive alg. to print out the top 10 integers of the array
 * that has been sorted.
 *
 * @param arrRe the array we are looking at
 * @param n The index that you start at.
 *//*
public void topTenDsc(int[] arrRe, int n){
    if(!(n%3==0 && n%10==0)){                          // Since I know what we are going to be passing in I can use this module arithmetic
        System.out.println("Recursive " + arrRe[n -1]);  // Print the largest numbers first before calling the method again.
        topTenDsc(arrRe, n - 1);                       // Recursive call.
    }
}
*/
/*
public int getRecu(int[] arrRec, int y, int n){

    return y;
}


public int findMax(int[] ray, int sizeArr, boolean isFirst){

    if(isFirst){
        topTen[0] = getMax(ray, sizeArr);
    }
    int lastMax;
    for(int i =1;i<10;i++){
        int temp1 = getMax(ray,sizeArr-1);
        if(topTen[i]<temp1&&temp1<topTen[i-1]){
            int temp2=topTen[i];
            topTen[i]=temp1;
```

Through all of them I was trying dynamic programming to store the values into an array

(topTen) of size 10. I was then reminded of the pivot strategy that we used for partition method, so I thought that I could use that kind of methodology to solve this problem. Only worrying about the ten values that I care or have been tasked to find. I also employed divide-and-conquer as I separated the max value out of the array that I was parsing each time. This made my problem smaller each time, and focused on the new problem, finding the next maximum value.

My execution time for my helper method to return the index of the max is O(n) then I would multiply that by 10 because we call that method 10 times in the step 6 main method + some of the steps that are taken in the execution of both the methods, which we will ignore as this is in respect to time complexity. So, the time complexity is O(n) time.