Group 10

By Ollie Willette and KD

Scores and Times

- Smaller data set:
  - 139 milliseconds
  - 133 milliseconds
  - 131 milliseconds
- Larger data set
  - 209 milliseconds
  - 219 milliseconds
  - 216 milliseconds

- Sum of medians was **349.00**
- Time complexity: O(n^2), O(mlogm)

?

Walkthrough

- HashMap
- TreeMap
- ArrayList

```
public static void sort(Integer[] toSort) {
        // Take the array. seperate it into groups of equvalante # of ones into
        // a hashmap<int, treemap<int, arraylist<int>>>. Then for each treemap<int, arraylist<int>>
        // add to the treemap with the key as LLRS, and the value to the associated array. Then sort each of those.
        HashMap<Integer,TreeMap<Integer, ArrayList<Integer>>> hm = new HashMap<>();
        for(int i = 0; i< toSort.length; i++) {
                int b = numBinaryOnes(toSort[i]);
                int LLRS = lengthLongestRepeatedSubstring(Integer.toBinaryString(toSort[i]));
                hm.putIfAbsent(b, new TreeMap<Integer,ArrayList<Integer>>());
                hm.get(b).putIfAbsent(LLRS, new ArrayList<Integer>());
                hm.get(b).get(LLRS).add(toSort[i]);
        }
        int i = 0; // index to put next value
        Iterator<TreeMap<Integer, ArrayList<Integer>>> it = hm.values().iterator();
        while(it.hasNext()) { // for every treemap in the hashmap
                Iterator<ArrayList<Integer>> itit = it.next().values().iterator(); // it.next().values().iterator() = treemap iterator
                while(itit.hasNext()) { // for every array in the treemap
                        ArrayList<Integer> a = itit.next();
                        a.sort(null);
                        for(int j = 0; j<a.size();j++) {// for every value in the arraylist<int>
                                toSort[i++]= a.get(j);
                        }
                }
        }
}
```

lengthLongestRepeatedSubstring()

```
public static int lengthLongestRepeatedSubstring(String binary) {
        if(binary.length() == 1) {
                return 0;
        }
        int l = 0;
        int h = 1;
        String substring = binary.substring(l, h);
        String afterstring = binary.substring(h);
        while(afterstring.contains(substring)||afterstring.length()>(h-l)) {
                if(afterstring.contains(substring)) {
                        substring = binary.substring(l, ++h);
                        afterstring = binary.substring(h);
                }else {
                        substring = binary.substring(++l,++h);
                        afterstring = binary.substring(h);
                }
        }

        return h-l-1;
```

External Code Source

- We implemented a method from Geeks for Geeks

```
public static int numBinaryOnes(int n){
        int count = 0;
        while (n > 0) {
            count += n & 1;
            n >>= 1;
        }
        return count;
}
```