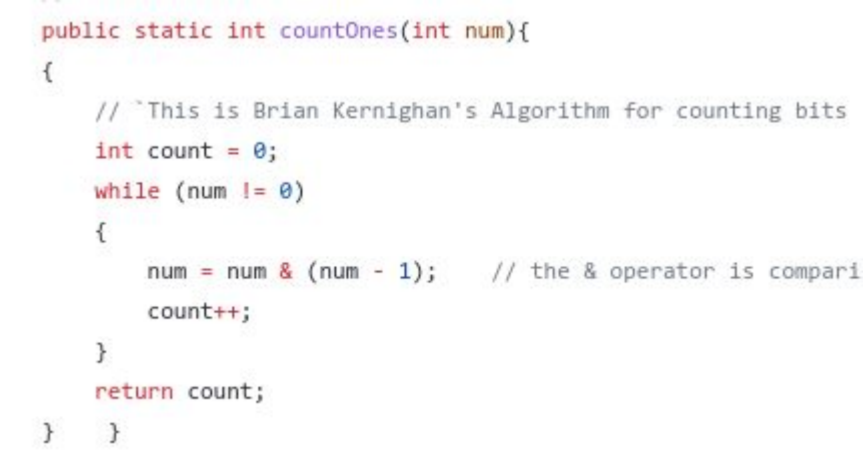
Group 1

Austin and Shawn

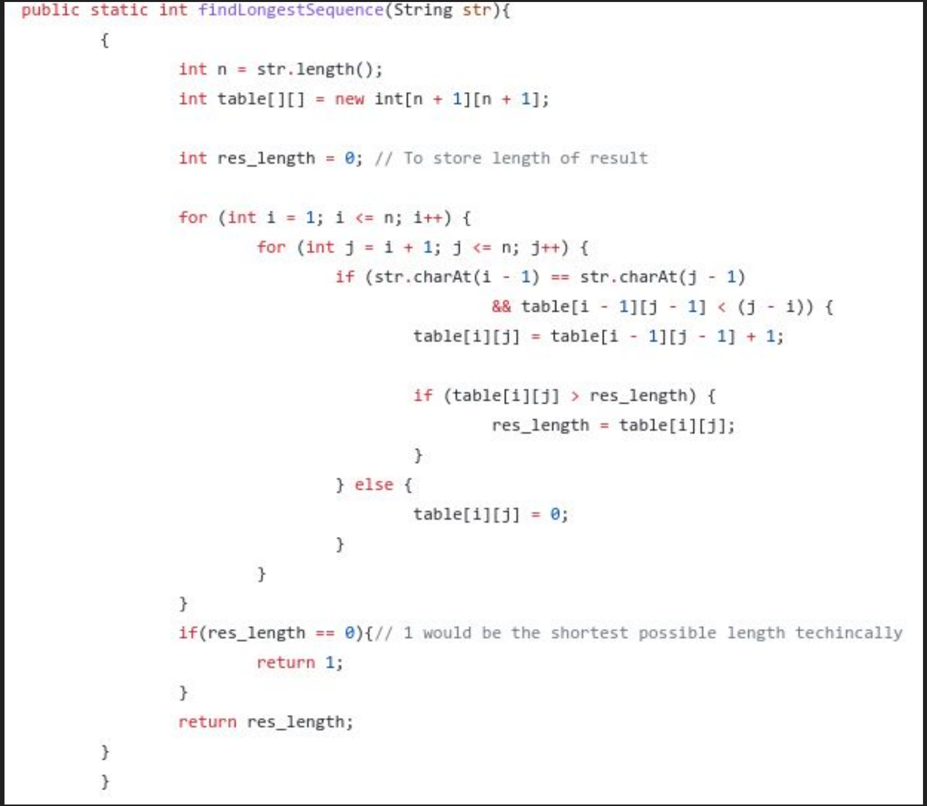
Overview

- Times: 2468 and 4449
- Final Score: 6917.0 7th place
- Main sorting algorithm
    - Mergesort with insertion sort
    - Data stored in a Integer array

Algorithm

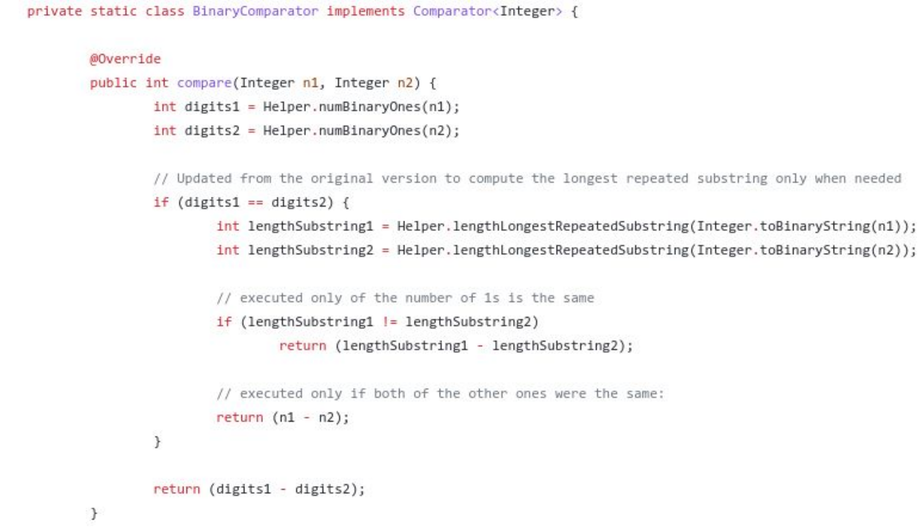


```
public static int countOnes(int num){
{
     // `This is Brian Kernighan's Algorithm for counting bits
     int count = 0;
     while (num != 0)
     {
          num = num & (num - 1);    // the & operator is compari
          count++;
     }
     return count;
}    }
```

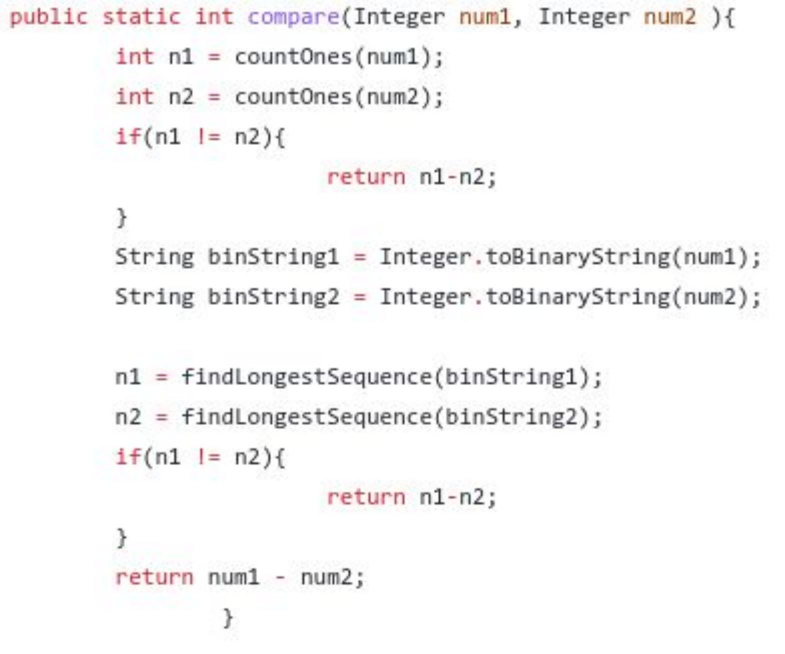


```
public static int findLongestSequence(String str){
        {
                int n = str.length();
                int table[][] = new int[n + 1][n + 1];

                int res_length = 0; // To store length of result

                for (int i = 1; i <= n; i++) {
                        for (int j = i + 1; j <= n; j++) {
                                if (str.charAt(i - 1) == str.charAt(j - 1)
                                                && table[i - 1][j - 1] < (j - i)) {
                                        table[i][j] = table[i - 1][j - 1] + 1;

                                        if (table[i][j] > res_length) {
                                                res_length = table[i][j];
                                        }
                                } else {
                                        table[i][j] = 0;
                                }
                        }
                }
                if(res_length == 0){// 1 would be the shortest possible length techincally
                        return 1;
                }
                return res_length;
        }
        }
```

Changes made from Group 0



```
private static class BinaryComparator implements Comparator<Integer> {

    @Override
    public int compare(Integer n1, Integer n2) {
        int digits1 = Helper.numBinaryOnes(n1);
        int digits2 = Helper.numBinaryOnes(n2);

        // Updated from the original version to compute the longest repeated substring only when needed
        if (digits1 == digits2) {
            int lengthSubstring1 = Helper.lengthLongestRepeatedSubstring(Integer.toBinaryString(n1));
            int lengthSubstring2 = Helper.lengthLongestRepeatedSubstring(Integer.toBinaryString(n2));

            // executed only of the number of 1s is the same
            if (lengthSubstring1 != lengthSubstring2)
                return (lengthSubstring1 - lengthSubstring2);

            // executed only if both of the other ones were the same:
            return (n1 - n2);
        }

        return (digits1 - digits2);
    }
```



```
public static int compare(Integer num1, Integer num2 ){
    int n1 = countOnes(num1);
    int n2 = countOnes(num2);
    if(n1 != n2){
            return n1-n2;
    }
    String binString1 = Integer.toBinaryString(num1);
    String binString2 = Integer.toBinaryString(num2);

    n1 = findLongestSequence(binString1);
    n2 = findLongestSequence(binString2);
    if(n1 != n2){
            return n1-n2;
    }
    return num1 - num2;
        }
```

Worst Case and Expected case

Time complexity:

- Mergesort: O(n*log(n)) --- not taking in account of insertion sort at end
- Longest substring: O(m^2)
- Count ones: O(log(m))

Possible Changes in the future

- More efficient Longest substring algorithm
- Improve upon actual sorting method
- Not use Integer data type for numbers