Problem 1

a)

The Goofy Algorithm will work if he is lucky enough, the best case scenario will happen if Goofy runs the Algo when it's already sorted.

The worse case might never happen which will result in a stack overflow bse the base case might never be achieved

The average case will be achieved at some point if Goofy runs the Algo and he gets the result after so many trials

b) Best case scenario will happen if Goofy gets an already sorted array as his input

c) Best case runtime is O(N) because it will loop through to check if the array is already sorted

d) The worst case scenario of this Algorithm is Infinite. This will might cause a stack overflow even before the algorithm gets you a result

e) Assuming you input an array of 9 elements

Average no of inversions = n(n-1)/4

= 9(9 - 1)/4 = 18 inversions on average

Assuming you got results on the second shuffle. This means that the algorithm went through the array atleast twice.

Total number of Comparisons = 9 * 2 = 18

Since for an algorithm to be inversion bound the number of comparisons should be equal or less the number of comparisons. Meaning that this Goofy's Algorithm is inversion bound

Problem 2

```
public int[] sortAlgo(int[] array){
        int countOnes = 0, countZeros = 0;
        for(int num : array){
            if(num == 0){
                countZeros++;
            }else if(num == 1){
                countOnes++;
            }
        }
        for(int i = 0; i <array.length; i++){
            if(i < countZeros) array[i] = 0;
            else if(i >= countOnes && i < (countZeros + countOnes))  array[i] = 1;
            else array[i] = 2;
        }
```

```
        return array;
    }
```

The above algorithm sorts the array in O(N) because the runtime is O(N + N) which becomes O(N)

Problem 3

```
package com.mum.edu;

import com.runtime.Sorter;
import java.util.Arrays;

public class BubbleSort1 extends Sorter {
   private int[] array2 = {2,4,1,9,0,10,1000,0,3,2};

   public BubbleSort1(){
       System.out.println("Before : "+Arrays.toString(array2));
       //sort();
       System.out.println("After  : "+Arrays.toString(sort(array2)));
   }


   private void swap(int[] array,int i, int j){
       int temp = array[i];
       array[i] = array[j];
       array[j] = temp;
   }
   public static void main(String[] args) {
       new BubbleSort1();
   }

   @Override
   public int[] sort(int[] arr) {
       int len = arr.length;
       for(int i = 0; i < len; i++){
           for (int j = 0; j < len - 1; j++){
               if(arr[j] > arr[j+1]){
                   swap(arr,j,j+1);
               }
           }
       }
       return arr;
   }
}

package com.mum.edu;
```

```java
import com.runtime.Sorter;

import java.util.Arrays;

public class BubbleSort2 extends Sorter {
   private int[] array2 = {2,4,1,9,0,10,1000,0,3,2};

   public BubbleSort2(){
       System.out.println("Before : "+Arrays.toString(array2));
       //sort();
       System.out.println("After  : "+Arrays.toString(sort(array2)));
   }


   private void swap(int[] array,int i, int j){
       int temp = array[i];
       array[i] = array[j];
       array[j] = temp;
   }
   public static void main(String[] args) {
       new BubbleSort2();
   }

   @Override
   public int[] sort(int[] arr) {
       int len = arr.length;
       for(int i = 0; i < len; i++){
           for (int j = 0; j < len - 1; j++){
               if(arr[j] > arr[j+1]){
                   swap(arr,j,j+1);
               }
           }
       }
       return arr;
   }
}
```

Problem 4

```
public int[] countOfZerosAndOnes(int[] arr){
    int index = binSearch(arr,0,arr.length-1);
    int countZeros = index + 1;
    int countOnes = arr.length - countZeros;
    return new int[]{countZeros,countOnes};
}

private int binSearch(int[] arr, int lower,int upper){
    int mid = (lower + upper)/2;
    if(arr[mid]==0 && arr[mid+1]==1) return mid;
    else if(arr[mid]==1 && arr[mid-1]==0) return mid-1;
    else if(arr[mid]==1 && arr[mid-1]==1) return binSearch(arr,lower, mid-1);
    else return binSearch(arr,mid+1,upper);
}
```