Etan Soclof

1. Brute force algorithm - For each virus in array, compare to all other viruses in the array and count the number of equalities. If the number of equalities is greater than half length of the array, return index of element. Order of growth is $O(n^2)$.

2. Proposed Algorithm - Split the array into two halves and recursively find the most prevalent virus (MPV) of each half if one exists and how many of that virus is in the array. If an MPV exists for the first half of the array, compare that MPV with every virus in the second half of the array and count the amount of equalities. If the total number of MPV is greater than the size of the total array, return the MPV. If not, then if an MPV exists for the second half of the array, compare that MPV with every virus in the first half of the array and count the amount of equalities. If the total number of the MPV virus is greater than the size of the total array, return the MPV. In all other cases, return -1 as there is no MVP.

3.

```
if length of array is <= 3 {
      Check for MPV by counting equality of elements
      return MPV and number of MVP
}
else {
      Split array into two arrays - array1, array2 - and recursively
find MPV and number of MPV of each array {
            if there is an MPV in array1{
                  int amount = amount of MVP in array1
                  for all viruses in array2, check{
                        if virus = MPV of array1{
                              amount++
                        }
                  }
                  if amount > array.length/2{
                        return MPV and amount
                  }
            }
            if there is an MPV in array2{
                  int amount = amount of MVP in array2
                  for all viruses in array1{
                        if virus = MPV of array2{
                              amount++
                        }
                  }
                  if amount > array.length/2{
                        return MPV and amount
                  }
                  else{
                        return null
                  }
            }
      }
      return null
}
```

$$Q(n) = \begin{cases} 4 & \text{if } n \text{ is <= 3} \\ 2Q(n/2) + O(n) & \text{otherwise} \end{cases}$$

By the Master theorem, a = 2, b = 2, d = 1. Since a = b^d, Q(n) = O(nlogn).

Proof:
When splitting an array of size 2n to two subarrays of size n, two cases must be considered:

1. Neither subarray contain an SPV - In such a case, there will be no SPV for the array. This is because there is no virus that appears more than n/2 times in the first subarray and no virus that appears more than n/2 times in the second subarray. If each subarray have a particular virus that appears n/2 times in each subarray, this will amount to n times in the array which would not qualify for SPV.
2. One or both subarrays contains an SPV - In such a case, the only candidate for an SPV for the array is the SPVs of the two subarrays. If a particular virus appears n/2 + 1 or more times in a subarray, it might appear n/2 (or less) times in the other subarray which can amount to a total of n+1 appearances in the array.

Sketch

```
Created a MP class to store index of virus and number of time virus
appears in given array/subarray.

Set start = 0, end = array.length - 1

if (end - start <= 2){
      Check all permutations of the three elements to see if two are
equal or three are equal. If two are equal, return MP(index, 2) and if
three are equal, return MP(index, 3). Otherwise return null.
  }

int mid = (end - start)/2;
MP mp1 = mostPrevalent(viruses, checker, start, mid);
MP mp2 = mostPrevalent(viruses, checker, mid + 1, end);

if (mp1 != null) {
      Set amount = mp1.amount;
      for all elements i from mid+1 to end {
            if (array[mp1.index] == array[i]
                  amount++
      }
}
      if (amount is greater than size of end - start, i.e., the length
of combination of the two subarrays){
            return index and amount;
      }
  }
if (mp2 != null){
      repeat above for first subarray
 }

if no MPV was found, return null
```