5. (A). The best-case for the BubbleSort is when the input array is already sorted in increasing order. Because if the input array is in increasing order already there will not be any swap operations will be made.

The worst-case scenario is when the given array is in decreasing order in that case there would be a swap operation in every comparison.

(B). Both the best and the worst cases of the given BubbleSort is $O(N^2)$ because even in the best scenario where no swap happens there will be a N^2 comparison operations will be conduct.

(C). We can stop the loop if there is not a single swap operation occurred. If an already sorted list is given then the loop sorting will immediately after a single iteration.

```
public int[] BubbleSort1(int[] arr) {
    int len = arr.length;
    for(int i = 0; i < len; ++i) {
        boolean swapped = false;
        for(int j = 0; j < len - 1; ++j) {
            if(arr[j] > arr[j + 1]) {
                int temp = arr[j + 1];
                arr[j + 1] = arr[j];
                arr[j] = temp;
                swapped = true;
            }
        }
        if(swapped == false) {
            break;
        }
    }
    return arr;
}
```

(D). Let I(i) be the statement the elements arr[n-i-1]..arr[n-1] are in final sorted order.

Base case: I(0): arr[n-1] is an already in a sorted order.

Induction: Lets assume I(i) is true and prove I(i+1) is true. Then arr[n-i-1]..arr[n-1] will be already sorted and we need to put arr[n-i] in a correct position. Since the second operation starts from the beginning of the list and compares the current element to its next element, we always keeps the higher values in the next position. In other words, if the correct position of the arr[n-i] is x (n-i-1 < x < n-1) then until x-th position we will keep the value and shifting smaller values forward.

(E). Almost same as the insertion sort. Suppose arr is an input array, I < j < n and x=arr[i]>arr[j]=y. At some point we need to change order of the x and y to get the sorted array. So, there definitely will occur a comparison between x and y. In other words, Bubble sort is an Inversion-Bound sorting algorithm.

6. Let us define 2 new integer variables named with countZero, and countOne. Using these 2 variables we can count how many 0's, and 1's in the given array. After that we can just change the first countZero elements of the array with 0's, next countOne elements of the array with 1's, and rest of the elements of the array with 2's in a single iteration.

There is 2 loops but not a single nested loop since the running time of the algorithm is O(N), also the there is O(1) additional space is used.

```
public int[] problem6(int[] arr){
    int countZero, countOne;
    countZero = 0;
    countOne = 0;

    for(int i = 0; i < arr.length; i++) {        // first loop
        if(arr[i] == 0) {countZero ++;}
        if(arr[i] == 1) {countOne ++;}
    }

    for(int i = 0; i < arr.length; i++) {        // second loop
        if(i < countZero) {
            arr[i] = 0;
        } else {
            if (i < countZero + countOne) {
                arr[i] = 1;
            } else {
                arr[i] = 2;
            }
        }
    }
    return arr;
}
```