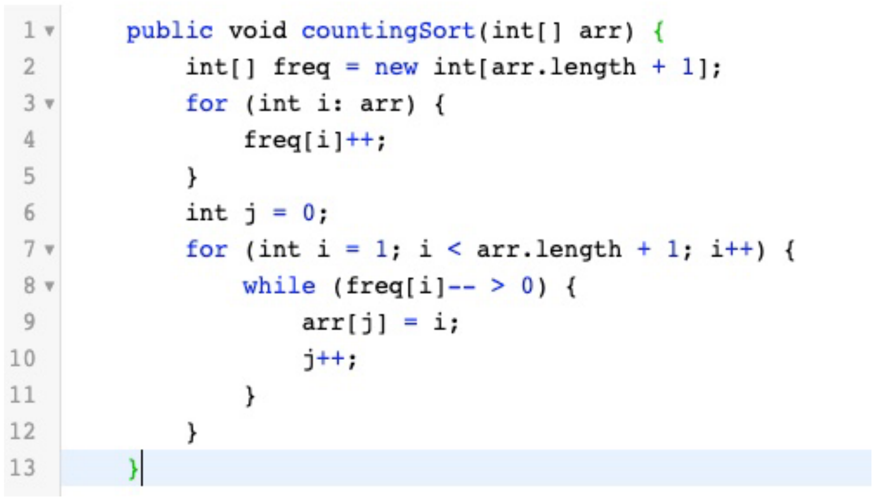
Q 11:

We can have an efficient sorting like O(n), because in the example array, which have many duplicated numbers, I will initiate a new array and sorted the numbers of frequency in the new array. then I'll iterate the the input array in a sorted order.

So, I'll get O(n) time complexity after sorting.

I'll show a code example as well.



```
public void countingSort(int[] arr) {
    int[] freq = new int[arr.length + 1];
    for (int i: arr) {
        freq[i]++;
    }
    int j = 0;
    for (int i = 1; i < arr.length + 1; i++) {
        while (freq[i]-- > 0) {
            arr[j] = i;
            j++;
        }
    }
}
```