CS/ECE 374 Spring 2017 — Renheng Ruan (rruan2)
Homework 4 Problem 3 — Lanxiao Bai (lbai5)

(a) Describe an algorithm to determine in $O(n)$ time whether an arbitrary array $A[1..n]$ contains more than $n/4$ copies of any value.

(b) Describe and analyze an algorithm to determine, given an arbitrary array $A[1..n]$ and an integer k, whether A contains more than k copies of any value. Express the running time of your algorithm as a function of both n and k.

Do not use hashing, or radix sort, or any other method that depends on the precise input values.

Solution:

(a) The first thing in our algorithm is want to create a temporary array of size 3 to store elements and the counts. This step takes $O(4)$ time.

```
INITIALIZE():
  for i ← 0 to 3
      temp[i].count = 0
```

We treat each number as a piece in Tetris (a tile-matching puzzle video game). Each number in the original array will falls down in the temporary array. The same number will be stacked on the same column. And the times the number appears is counted. If the temporary array is full and there is a new element needed to be added. Then we remove the bottom row from stacks of elements (decrease count of every element by 1 in the temporary array at same time) and ignore this element. This step takes $O(4n)$ time.

```
TetrisSort(A[s..n], n):
  for i ← 0 to n
      for j ← 0 to 3
          if temp[j].element = A[i] (A[i] is already in array temp[])
              temp[j].count ← temp[j].count + 1
              break
      if j = 3 (A[i] is not in array temp[])
          for l ← 0 to 3(there is space in last row of temp[])
              if temp[l].count = 0
                  temp[l].element ← A[i]
                  temp[l].count ← 1
                  break
          if l = 3(there is no space in last row of temp[])
              for l ← 0 to 3
                  temp[l].count ← temp − 1
```

At last, we go through the every element stored in temporary array to check if it actually has count more than $n/4$. This step takes $O(4n)$ time.
So the overall running time is $O(n) + O(4n) + O(4n) = O(9n)$ which is also O(n).

```
COMPARE():
  for i ← 0 to 3
      for j ← 0 to n
          if A[j] = temp[i].element
              count ← count + 1
  if count > n/4
      print temp[i].element and count
```

(b) At this part, we want determine the array A contains more than k copies of any value or not. Using the same algorithm in part (a), we calculated it $n/4$ as k which $n/k = 4$, and 4 is the coefficient of the running time in part (a).
So in this case, the running time is $O(n/k) + O(n^2/k) + O(n/k)$ which is $O(n^2/k)$.

CITE[1]

■

[1]Idea comes from http://www.geeksforgeeks.org/given-an-array-of-of-size-n-finds-all-the-elements-that-appear-more-than-nk-times/