Nordeus Coding Challenge

QA puzzle

First pass through the code

import java.util.Arrays;

First thing I saw is unused import. It should be deleted.

int[] output = new int[size];

int max = array[0];

Passing through the code I have realized that the values of size and array are not checked. We can’t work with negative or zero values of size or null array.

if (array[i] > max)  
 max = array[i++];

As we can see every time we find another greatest number in array, we increment i which is not correct because in that case we will increment value i twice (after value assignment and after current iteration) and doing that we skip checking some numbers so max=array[i++]should be changed to max=array[i].

int[] count = new int[max];

We have to check if max==0 because making array with 0 numbers doesn’t make sense.

int k = 0;  
while (k < max) {  
 count[k] = 0;  
}

K is never incremented so the program is stuck in infinity loop here. Count[k] has to be changed to count[k++].

for (int i = 0; i > size; i++) {  
 count[array[i]]++;  
}

Condition i>size is never true. We should go through array and count every element which means that we should change > to <. At this point I realized that array count should be the size of max+1 because it should store the count of each element from 0 to max which means max+1 numbers and also because of that we don’t have to check if max==0 as I previously thought.

for (int i = 1; i <= max; i++) {  
 count[i] += count[--i];  
}

Irregular usage of --i because in this case we are stuck in infinity loop where i is constantly switching between 0 and 1. It should be fixed like this: count[i]+=count[i-1].

output[count[array[i]]] = array[i];

Count[array[i]] will be equivalent size for i=size-1 which is out of boundaries because the output array is the size of size so we should change value assignment to output[count[array[i]]-1].

After the initial pass and code corrections it looks like this:

class JobFair {

static void doSomething(int array[], int size) { // Array contains non-negative integers

if ( size<=0 || array==null )return;  
 int[] output = new int[size];  
  
 // Find the largest element of the array  
 int max = array[0];  
 for (int i = 1; i < size; i++) {  
 if (array[i] > max)  
 max = array[i];  
 }  
 max++;  
 int[] count = new int[max];  
  
 // Initialize count array with all zeros.  
 int k = 0;  
 while (k < max) {  
 count[k++] = 0;  
 }  
  
 // Store the count of each element  
 for (int i = 0; i < size; i++) {  
 count[array[i]]++;  
 }  
  
 // Store the cumulative count of each array  
 for (int i = 1; i < max; i++) {  
 count[i] += count[i-1];  
 }  
   
 // Find the index of each element of the original array in count array, and  
 // place the elements in output array  
 for (int i = size - 1; i >= 0; i--) {  
 output[count[array[i]]-1] = array[i];  
 count[array[i]]--;  
 }  
  
 // Copy elements into original array  
 for (int i = 0; i < size; i++) {  
 array[i] = output[i];  
 }  
}

}

Second pass

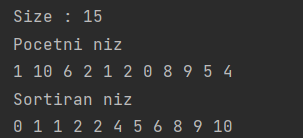
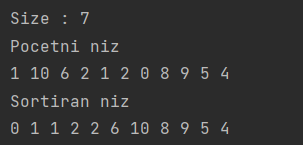
Parameter size can be different than array.length. It determines range that will be sorted starting from 0 so I added another check if size>array.length. In that case size gets length of array. I added another variable size2 so that function doesn’t change the value of original integer.

I have also deleted initialization of array with zeroes because it is implied in Java programming language.

Version 2 look like this:

static void doSomething(int array[], int size) { // Array contains non-negative integers  
 if ( size<=0 || array==null )return;  
 int size2=size;  
 if (size>array.length) size2=array.length;  
 int[] output = new int[size2];  
  
 // Find the largest element of the array  
 int max = array[0];  
 for (int i = 1; i < size2; i++) {  
 if (array[i] > max)  
 max = array[i];  
 }  
 max++;  
 int[] count = new int[max];  
  
 // Store the count of each element  
 for (int i = 0; i < size2; i++) {  
 count[array[i]]++;  
 }  
  
 // Store the cumulative count of each array  
 for (int i = 1; i < max; i++) {  
 count[i] += count[i-1];  
 }  
  
 // Find the index of each element of the original array in count array, and  
 // place the elements in output array  
 for (int i = size2 - 1; i >= 0; i--) {  
 output[count[array[i]]-1] = array[i];  
 count[array[i]]--;  
 }  
  
 // Copy elements into original array  
 for (int i = 0; i < size2; i++) {  
 array[i] = output[i];  
 }  
}

Tests



For test samples I choosed the version where function sorts the part of the array (first picture) and the version where function sorts whole array. I don’t think both of them are necessary. I think only first sample is enough because I considered borderline cases through code. This sort is not good for sparse arrays. For example, if we have i=[1,66777] array, function will go throught every number from 0 to 66777 and allocate a lot of memory just to sort two numbers. I think that sort should be optimazed with some chained list or binary tree for short arrays (next picture shows one of the problems in this case).

for (int i = 1; i < max; i++) {  
 count[i] += count[i-1];  
}

We can also change the range of observation. Instead of going through array from 0 to max we can find min and consider only numbers from min to max.

Instead of

for (int i = 1; i < max; i++) {  
 count[i] += count[i-1];  
}  
  
// Find the index of each element of the original array in count array, and  
// place the elements in output array  
for (int i = size2 - 1; i >= 0; i--) {  
 output[count[array[i]]-1] = array[i];  
 count[array[i]]--;  
}  
  
// Copy elements into original array  
for (int i = 0; i < size2; i++) {  
 array[i] = output[i];  
}

I also think that implementation could be simplified by deleting the cumulative count. We could add numbers to final array going from the min to max and adding them as many times as they occur in initial one like you can see on next picture:

k = 0;  
for (int i = 0; i < max; i++) {  
 for (int j=0;j<count[i];j++){  
 array[k++]=i;  
 }  
}

In this way we don’t need to go through array 3 times. We can finish sorting it in one pass and also it would be easier for hardware to assign array this way. I have also changed: if ( size<=0 || array==null )return; to if ( size<=1 || array==null )return;

Final version looks like this:

class JobFair {  
 static void doSomething(int array[], int size) { // Array contains non-negative integers  
 if ( size<=1 || array==null )return;  
 int size2=size;  
 if (size>array.length) size2=array.length;  
 int[] output = new int[size2];  
  
 // Find the largest element of the array  
 int max = array[0];  
 int min = array[0];  
 for (int i = 1; i < size2; i++) {  
 if (array[i] > max)  
 max = array[i];  
 if (array[i] < min)  
 min = array[i];  
 }  
 max++;  
 int[] count = new int[max];  
  
 // Store the count of each element  
 for (int i = 0; i < size2; i++) {  
 count[array[i]]++;  
 }  
  
 // Find the index of each element of the original array in count array, and  
 // place the elements in output array   
 int k = 0;  
 for (int i = min; i < max; i++) {  
 for (int j = 0; j < count[i]; j++) {  
 array[k++] = i;  
 }  
 }  
   
 }  
}