You can view this report online at : https://www.hackerrank.com/x/tests/574827/candidates/14068682/report

Full Name:	Temitope Akinsoto
Email:	takinsoto@gmail.com
Test Name:	**Lambda School - Web Whiteboard Fitness Assessment 4**
Taken On:	25 Apr 2020 07:06:49 PDT
Time Taken:	170 min 36 sec/ 210 min
Invited by:	Josh
Invited on:	23 Apr 2020 11:52:04 PDT

100%

175/175

scored in **Lambda School - Web Whiteboard Fitness Assessment 4** in 170 min 36 sec on 25 Apr 2020 07:06:49 PDT

Tags Score:

Algorithms	175/175
Arrays	50/50
Core CS	50/50
Data Structures	125/125
Easy	100/100
Hash Maps	75/75
Hash Tables	50/50
Medium	75/75

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

	Question Description	Time Taken	Score	Status
Q1	Filtering Signals > **Coding**	44 min 15 sec	50/ 50	!
Q2	Equalize Array > **Coding**	33 min 32 sec	50/ 50	✓
Q3	Frequency Queries > **Coding**	1 hour 32 min 15 sec	75/ 75	!

QUESTION 1

!

Needs Review

Score 50

Filtering Signals > Coding

Algorithms | Easy | Core CS

QUESTION DESCRIPTION

Sometimes it is necessary to filter a signal by frequency, e.g. to reduce noise outside of the expected frequency range. Filters can be stacked, allowing only the frequencies within the range allowed by all filters to get through. For example, three filters with ranges of *(10, 17)*, *(13, 15)* and *(13, 17)* will only allow signals to get through. For example, three filters with ranges of *(10, 17)*, *(13, 15)* and *(13, 17)* will only allow signals between *13* and *15* through. The only range that all filters overlap is *(13, 15).* Given *n* signals frequencies and a series of *m* filters that let through frequencies in the range *x* to *y*, inclusive, determine the number of signals that will get through the filters.

For example, given *n = 5* signals with *frequencies = [8, 15, 14, 16, 21]* and *m = 3 filtersRanges = [[10, 17], [13, 15], [13, 17]],* the *2* frequencies that will pass through all filters are *15* and *14.*

Function Description

Complete the *countSignals* function in the editor below. The function must return an integer that denotes the number of signals that pass through all filters.

countSignals has the following parameter(s):

*frequencies: a*n integer array, the frequencies of the signals sent through the filters

filterRanges: a 2D integer array, the lower and upper frequency bounds for each filters

Constraints

- $1 \le n \le 10^5$
- $1 \le frequencies[i] \le 10^9$
- $1 \le m \le 10^5$
- $1 \le filterRanges[j][k] \le 10^9$

▶ Input Format For Custom Testing

▼ Sample Case 0

Sample Input For Custom Testing

```
5
20
5
6
7
12
3
2
10 20
5 15
5 30
```

Sample Output

```
1
```

Explanation

The common pass-through range is *10* to *15,* so only frequency *12* passes through.

▶ Sample Case 1

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
/*
 * Complete the 'countSignals' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER_ARRAY frequencies
 *  2. 2D_INTEGER_ARRAY filterRanges
 */

function countSignals(frequencies, filterRanges) {
    // for this problem, we're simpy looping through the frequencies array
    // for this problem, we're simply looping through the frequencies array.
for each freqency,
    // if that frequency passes through all the signals, then we filter it
out

    // for a frequency to pass through each signal, its value must fall
within the range of
    // frequency signal ( both upper and lower limits inclusive)
    // Write your code here
    let range = [ filterRanges[0][0], filterRanges[0][1] ];
    for(let filter of filterRanges){
        if(filter[0] > range[0]){
            range[0] = filter[0]
        }
        if(filter[1] < range[1]){
            range[1] = filter[1]
        }
    }
    let count = 0;
    for(let value of frequencies){
        if(value >=range[0] && value <= range[1]) {
            count++
        }
    }
    return count

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.1651 sec	30.4 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.2039 sec	30.4 KB
Testcase 2	Easy	Sample case	✓ Success	1	0.1609 sec	30.4 KB
Testcase 3	Easy	Sample case	✓ Success	2	0.1543 sec	30.4 KB
Testcase 4	Easy	Sample case	✓ Success	2	0.134 sec	30.4 KB
Testcase 5	Easy	Sample case	✓ Success	2	0.169 sec	33.6 KB
Testcase 6	Medium	Sample case	✓ Success	2	0.2346 sec	35.4 KB
Testcase 7	Medium	Sample case	✓ Success	2	0.1667 sec	33.5 KB
Testcase 8	Medium	Sample case	✓ Success	3	0.1839 sec	36.8 KB
Testcase 9	Medium	Sample case	✓ Success	3	0.2326 sec	39.3 KB
Testcase 10	Medium	Sample case	✓ Success	3	0.1984 sec	40.9 KB
Testcase 11	Hard	Sample case	✓ Success	7	0.2744 sec	55.1 KB
Testcase 12	Hard	Sample case	✓ Success	7	0.3165 sec	60 KB
Testcase 13	Hard	Sample case	✓ Success	7	0.3439 sec	77.9 KB
Testcase 14	Hard	Sample case	✓ Success	7	0.3745 sec	86.5 KB

No Comments

QUESTION 2

✓ Correct Answer

Score 50

Equalize Array › Coding

Algorithms | Data Structures | Arrays | Hash Tables | Easy

QUESTION DESCRIPTION

Karl has an array of integers. He wants to reduce the array until all remaining elements are equal.

Determine the minimum number of elements to delete to reach his goal.

For example, if his array is `[1, 2, 2, 3]`, we see that he can delete the **2** elements **1** and **3** leaving `[2, 2]`. He could also delete both **2s** and either the **1** or the **3**, but that would take 3 deletions. The minimum number of deletions is 2.

Function Description

Complete the `equalizeArray` function in the editor below. It must return an integer that denotes the minimum number of deletions required.

`equalizeArray` has the following parameter:

- ***arr***: an array of integers

Input Format

The first line contains an integer ***n***, the number of elements in ***arr***.
The next line contains ***n*** space-separated integers ***arr[i]***.

Output Format

Print a single integer that denotes the minimum number of elements Karl must delete for all elements in the array to be equal.

Sample Input

```
5
3 3 2 1 3
```

Sample Output

```
2
```

Explanation

We're given an array `[3, 3, 2, 1, 3]`. If we delete `arr[2] = 2` and `arr[3] = 1`, the resulting array is `[3, 3, 3]`. All of the elements are thus equal. Deleting these ***2*** is minimal. Our only other options would be to delete ***4*** elements to get an array of either `[1]` or `[2]`.

CANDIDATE ANSWER

Language used: **Python 3**

```
#
# Complete the 'equalizeArray' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER_ARRAY arr as parameter.
#

def equalizeArray(arr):
    # Write your code here

    # For this problem, we simply want to get item of the arr with the
highest_count, then count
    # rest of the items whose freq is not the highest_count i.e number of
items to be deleted
```

```
    # return num_to_del

    char_count = {}
    num_to_del = 0
    highest_count = 0
    if len(arr) == 0:
        return char_count

    #count all the items in original array
    for item in arr:
        if item in char_count:
            char_count[item] += 1
        else:
            char_count[item] = 1

    # get the item or element with the highest count
    for val in char_count:
        if char_count[val] > highest_count:
            highest_count = char_count[val]

    # get the total number of items to be deleted from the array by
    # subtracting the item with the highest count from the total length of
the array

    num_to_del = len(arr) - highest_count

    return num_to_del
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.1608 sec	10.9 KB
Testcase 1	Easy	Sample case	✓ Success	6	0.1419 sec	11 KB
Testcase 2	Easy	Sample case	✓ Success	2	0.1885 sec	11 KB
Testcase 3	Easy	Sample case	✓ Success	6	0.1676 sec	10.6 KB
Testcase 4	Easy	Sample case	✓ Success	2	0.1497 sec	10.5 KB
Testcase 5	Easy	Sample case	✓ Success	6	0.1789 sec	10.9 KB
Testcase 6	Easy	Sample case	✓ Success	6	0.1416 sec	10.9 KB
Testcase 7	Easy	Sample case	✓ Success	6	0.1366 sec	10.7 KB
Testcase 8	Easy	Sample case	✓ Success	7	0.1599 sec	10.9 KB
Testcase 9	Easy	Sample case	✓ Success	7	0.1433 sec	10.8 KB
Testcase 10	Easy	Sample case	✓ Success	1	0.1351 sec	11 KB

No Comments

QUESTION 3

!

Needs Review

Score 75

Frequency Queries › Coding

Algorithms | Hash Maps | Data Structures | Medium

QUESTION DESCRIPTION

You are given *q* queries. Each query specifies an operation that needs to be performed on an (initially empty) collection of integers. Each query has one of three possible forms:

- `[1, x]` : Insert *x* into your collection.
- `[2, y]` : Delete a single occurrence of *y* from your collection. Note that we might get delete queries for elements that aren't in the collection.
- `[3, z]` : Check if any integer present in the collection occurs with a frequency of *z*. If such an

integer occurs with the specified frequency in the collection, this operation outputs `1`. If no such integer occurs with the specified frequency in the collection, this operation outputs `0`.

The queries are given in the form of a 2D array ***queries*** of size ***q*** where `queries[i][0]` contains the operation and `queries[i][1]` contains the operation's input value.

For example, given an array of queries such as the following:
`queries = [[1,1],[3,3],[2,2],[3,-1],[1,1],[1,1],[2,1],[1,2],[3,2]]`, the results of each operation are as follows:

```
Operation   Collection   Output   Rationale
[1,1]       [1]                   Add a 1 to the collection
[3,3]       [1]           0       Check for an element with frequency 3;
no such element
[2,2]       [1]                   Remove a 2 from the collection; no such
element to remove
[3,-1]      [1]           0       Check for an element with frequency -1;
no such element
[1,1]       [1,1]                 Add a 1 to the collection
[1,1]       [1,1,1]               Add a 1 to the collection
[2,1]       [1,1]                 Remove a 1 from the collection
[1,2]       [1,1,2]               Add a 2 to the collection; occurrences
of 2 is now 1
[3,2]       [1,1,2]       1       Check for an element with frequency 2; 1
satisfies this
```

Thus our function should return `[0,0,1]`.

Function Description

Complete the `frequencyQueries` function in the editor below. It must return an array of integers where each element is 1 if there is at least one element value in the collection with the specified number of occurrences, or 0 if no such element is present in the collection. The returned array should hold the output values in the order in which the queries occurred.

`frequencyQueries` has the following parameter:

- queries: a 2D array of integers

Input Format

The first line contains an integer ***q***, the number of queries in the 2D array.
The second line contains an integer specifying the number of columns in the 2D array; this number will always be 2 in this problem.
The following ***q*** lines contain 2 space-separated integers representing the 2 elements in each query.

Output Format

Return an array of integers consisting of all outputs of queries of type 3.

Sample Input 1

```
8
2
1 5
1 6
3 2
1 10
1 10
1 6
2 5
3 2
```

Sample Output 1

```
0
1
```

Explanation 1

For the first query of type 3, there is no integer whose frequency is 2. At this point the collection consists of `[5,6]`. So that query outputs 0. For the second query of type 3, there are two integers in the collection with a frequency of 2, 6 and 10. So that query outputs 1.

Sample Input 2

```
10
2
1 3
2 3
3 2
1 4
1 5
1 5
1 4
3 2
2 4
3 2
```

Sample Output 2

```
0
1
1
```

Explanation 2

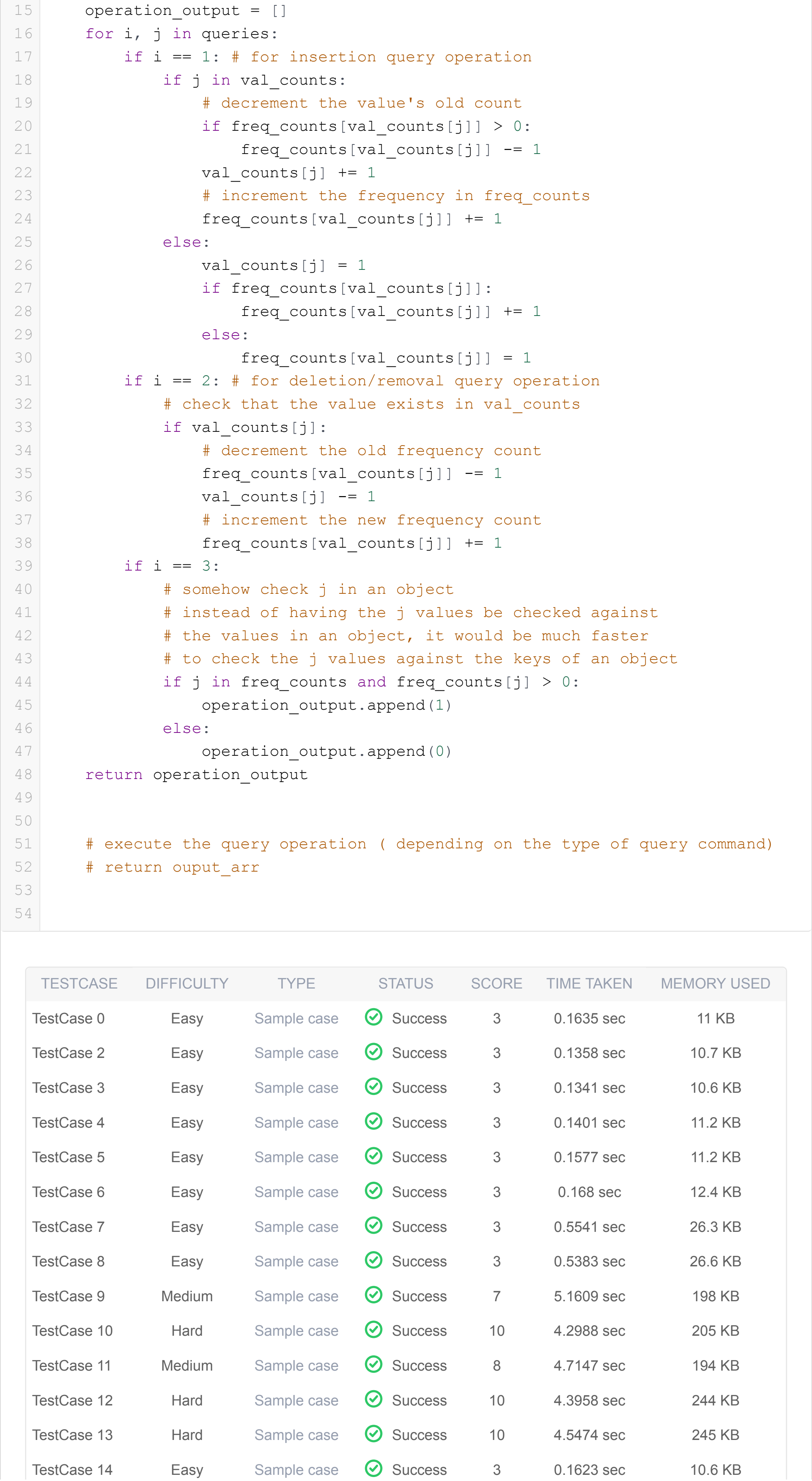
When the first query of type 3 occurs, the collection is empty, so this query outputs 0. By the time the second query of type 3 occurs, there are two integers in the collection, 4 and 5, that occur with a frequency of 2, so this query outputs 1. By the time the third query of type 3 occurs, there is one integer in the collection, 5, that occurs with a frequency of 2, so this query outputs 1.

CANDIDATE ANSWER

Language used: **Python 3**

```python
#
from collections import defaultdict
# Complete the 'frequencyQueries' function below.
#
# The function is expected to return an INTEGER_ARRAY.
# The function accepts 2D_INTEGER_ARRAY queries as parameter.
#

def frequencyQueries(queries):
    # Write your code here
    if len(queries) == 0:
        return False
    val_counts = defaultdict(int)
    freq_counts = defaultdict(int)
```



```
    operation_output = []
    for i, j in queries:
        if i == 1: # for insertion query operation
            if j in val_counts:
                # decrement the value's old count
                if freq_counts[val_counts[j]] > 0:
                    freq_counts[val_counts[j]] -= 1
                val_counts[j] += 1
                # increment the frequency in freq_counts
                freq_counts[val_counts[j]] += 1
            else:
                val_counts[j] = 1
                if freq_counts[val_counts[j]]:
                    freq_counts[val_counts[j]] += 1
                else:
                    freq_counts[val_counts[j]] = 1
        if i == 2: # for deletion/removal query operation
            # check that the value exists in val_counts
            if val_counts[j]:
                # decrement the old frequency count
                freq_counts[val_counts[j]] -= 1
                val_counts[j] -= 1
                # increment the new frequency count
                freq_counts[val_counts[j]] += 1
        if i == 3:
            # somehow check j in an object
            # instead of having the j values be checked against
            # the values in an object, it would be much faster
            # to check the j values against the keys of an object
            if j in freq_counts and freq_counts[j] > 0:
                operation_output.append(1)
            else:
                operation_output.append(0)
    return operation_output


    # execute the query operation ( depending on the type of query command)
    # return ouput_arr


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	3	0.1635 sec	11 KB
TestCase 2	Easy	Sample case	✓ Success	3	0.1358 sec	10.7 KB
TestCase 3	Easy	Sample case	✓ Success	3	0.1341 sec	10.6 KB
TestCase 4	Easy	Sample case	✓ Success	3	0.1401 sec	11.2 KB
TestCase 5	Easy	Sample case	✓ Success	3	0.1577 sec	11.2 KB
TestCase 6	Easy	Sample case	✓ Success	3	0.168 sec	12.4 KB
TestCase 7	Easy	Sample case	✓ Success	3	0.5541 sec	26.3 KB
TestCase 8	Easy	Sample case	✓ Success	3	0.5383 sec	26.6 KB
TestCase 9	Medium	Sample case	✓ Success	7	5.1609 sec	198 KB
TestCase 10	Hard	Sample case	✓ Success	10	4.2988 sec	205 KB
TestCase 11	Medium	Sample case	✓ Success	8	4.7147 sec	194 KB
TestCase 12	Hard	Sample case	✓ Success	10	4.3958 sec	244 KB
TestCase 13	Hard	Sample case	✓ Success	10	4.5474 sec	245 KB
TestCase 14	Easy	Sample case	✓ Success	3	0.1623 sec	10.6 KB

TestCase 15	Easy	Sample case	✓ Success	3	0.1531 sec	10.7 KB

No Comments

PDF generated at: 25 Apr 2020 16:59:10 UTC