Thursday, December 16, 2021 1:07 AM

Count Customers Who Did Not Get A Computer

Difficulty: EASY

A

Contributed By
Arshit Babariya
Avg. time to solve
15 min
Success Rate
85%

Problem Statement

Suggest Edit

Mr. X runs an internet cafe which has 'K' computers. His internet cafe has 'N' customers who can come anytime throughout the day. Initially, all the 'K' computers are available for customer use. When a customer enters the cafe he first checks whether any available computer is there. If he finds one he starts using it and it is marked unavailable. When he leaves the cafe that computer is again marked as available. If the customer is not able to find any available computer he leaves the cafe immediately.
You are given an integer array 'ARR' in which each value occurs exactly 2 times, the index of the first occurrence of any value denotes the arrival time of the customer while the second occurrence denotes the departing time of the customer. Your task is to find the no. of customers who were not able to find any available computer and had to leave the cafe immediately.

For Example :

Consider the sequence of customers as [1, 2, 3, 2, 3, 1] for N = 3 and K = 2.

The procedure takes place as follows :

1) At the start, Customer 1 comes and finds an available computer and starts using it and now the number of available computers is reduced by 1.

2) Customer 2 comes and he is also able to find an available computer and he starts using the computer. Now all the computers are unavailable.

3) Customer 3 comes and goes back immediately as there are no computers available currently.

4) Customer 2 leaves the cafe making 1 computer available.

5) As customer 3 has already left no new computers are made available.

6) Customer 1 leaves the cafe and all the computers are again available.

As only Customer 3 was unable to find any available computers therefore the answer is 1 for this case.

Input Format :

The first line of the input contains an integer 'T', denoting the number of test cases.

The first line of each test case contains two space-separated integers 'N' and 'K' denoting the number of customers and the number of computers available in the cafe respectively.

The second line of each test case '2*N' space-separated integers denoting the array 'ARR'.

Output Format :

For each test case print the number of customers who left the cafe without using a computer. Print the output of each test case in a new line.

Note :
You do not need to print anything, it has already been taken care of. Just implement the given function.

Constraints :
1 <= T <= 10
1 <= N <= 10^5
1 <= K <= 10^5
1 <= ARR[i] <= N

Time Limit: 1 sec

Sample Input 1:
```
2
2 2
1 2 2 1
3 2
1 3 2 1 2 3
```
Sample Output 1:
```
0
1
```
Explanation For Sample Input 1:
For the first test case:
1) Initially, Customer 1 comes and finds an available computer and starts using it. Now only 1 computer is available.

2) Then Customer 2 comes and takes the only available computer. Now all computers are unavailable.

3) Customer 1 leaves the cafe making 1 computer available.

4) Customer 2 leaves the cafe making all the computers available.

As all customers were able to find an available computer and nobody left without using a computer. Hence, the answer is 0 in this case.

For the second test case :
The procedure takes place as follows :
1) At the start, Customer 1 comes and finds an available computer and starts using it and now the number of available computers is reduced by 1.

2) Customer 3 comes and he is also able to find an available computer and he starts using the computer. Now all the computers are unavailable.

3) Customer 2 comes and goes back immediately as there are no computers available currently.

4) Customer 1 leaves the cafe making 1 computer available.

5) As customer 2 has already left no new computers are made available.

6) Customer 3 leaves the cafe and all the computers are again available.

As only Customer 2 was unable to find any available computers therefore the answer is 1 for this case.

Sample Input 2:
```
2
4 1
1 2 3 4 4 3 2 1
2 2
1 2 1 2
```
Sample Output 2:
```
3
0
```

From <https://www.codingninjas.com/codestudio/problems/count-customers-who-did-not-get-a-computer_1115775?topList=love-babbar-dsa-sheet-problems&leftPanelTab=0>

SOLUTION

Solution using status array

The idea is to store the status of each customer in an extra array and use two variables **availableComputers** and **customersWhoLeft** that stores the number of available computers and the number of customers who left because of no available computers respectively.

The status of a customer can have 3 possible values.

- **Status = 0,** if the customer has not visited till now.
- **Status = 1,** if the customer was able to find an available computer and is currently using it.
- **Status = -1,** if the customer has left the cafe.

Steps :

- If the **number of computers** is greater than or equal to the **number of customers**, then we will return 0 as in this case all customers will be able to find an available computer irrespective of the sequence with which they come and leave.
- Initialize an array **customerStatus** having **N+1** elements to store the status of each customer. Initialize all values with 0. As initially, no customer has visited the cafe. We are using **N+1** elements as we are considering 1-indexing.
- Set **availableComputers** as **K** and **customersWhoLeft** as 0.

- Iterate from **i = 0** to **2 * N - 1**
- If **customerStatus [ARR[i]]** is equal to 0 then
- If **availableComputers** is greater than 0 then decrease **availableComputers** by 1 and set **customerStatus[ARR[i]]** as 1.
- Otherwise, increase **customersWhoLeft** by 1 and set **customerStatus[ARR[i]]** as -1. We are setting the status as -1 here as the customer will leave the cafe immediately in this case.
- Otherwise, If **customerStatus [ARR[i]]** is equal to 1 then increase **availableComputers** by 1 and set **customerStatus [ARR[i]]** as -1. In this case, the customer has used the computer and has left the cafe after using it.
- Otherwise, If **customerStatus [ARR[i]]** is equal to -1 then we do not have to do anything as the customer has already left the cafe.
- Return **customersWhoLeft** as the final answer.

Time Complexity

O(N), where N denotes the number of customers.

As we are doing only one traversal of the input array having 2 * N elements. Hence, the overall Time Complexity is O(N).

Space Complexity

O(N), where N denotes the number of customers.

The number of elements in the **customerStatus** array is N. Hence, the overall Space Complexity is O(N).

From <https://www.codingninjas.com/codestudio/problems/count-customers-who-did-not-get-a-computer_1115775?topList=love-babbar-dsa-sheet-problems&leftPanelTab=2>

```
/*
   Time Complexity: O(N)
   Space Complexity: O(N)

   Where N denotes the number of customers.
*/

import java.util.ArrayList;

public class Solution {
      public static int countCustomers(ArrayList<Integer> arr, int k)

   {

      //   Finding the number of customers
      int n = (arr.size()) / 2;

      //   Checking if the number of computers are more than or equal to the number of customers


      if (k >= n)
```

```
{
return 0;
}

//    Defining the customer status array


int[] customerStatus = new int[n + 1];

//    Defining the variables


int customersWhoLeft = 0, availableComputers = k;

//    Iterating through the input array



for (int i = 0; i < 2 * n; i++)
{

//    If the customer status is 0 then we will check whether any computers are available


if (customerStatus[arr.get(i)] == 0)
   {

//    If computers are available then we will allot one computer to the customer otherwise we will increase
customersWhoLeft by 1


if (availableComputers > 0)
      {
availableComputers--;
customerStatus[arr.get(i)] = 1;
}
      else
      {
customersWhoLeft++;
customerStatus[arr.get(i)] = -1;
}
}

//    If the customer status is 1 then we will make 1 more computer available

else if (customerStatus[arr.get(i)] == 1)
   {
availableComputers++;
customerStatus[arr.get(i)] = -1;
}
}
```

```
        return customersWhoLeft;
        }
}
```

Function to find Number of customers who could not get a computer

- Difficulty Level : Medium
- Last Updated : 20 Jul, 2021

Write a function “runCustomerSimulation” that takes following two inputs

a) An integer ‘n’: total number of computers in a cafe and a string:

b) A sequence of uppercase letters ‘seq’: Letters in the sequence occur in pairs. The first occurrence indicates the arrival of a customer; the second indicates the departure of that same customer.

A customer will be serviced if there is an unoccupied computer. No letter will occur more than two times.

Customers who leave without using a computer always depart before customers who are currently using the computers. There are at most 20 computers per cafe.

For each set of input the function should output a number telling how many customers, if any walked away without using a computer. Return 0 if all the customers were able to use a computer.

runCustomerSimulation (2, “ABBAJJKZKZ”) should return 0

runCustomerSimulation (3, “GACCBDDBAGEE”) should return 1 as ‘D’ was not able to get any computer

runCustomerSimulation (3, “GACCBGDDBAEE”) should return 0

runCustomerSimulation (1, “ABCBCA”) should return 2 as ‘B’ and ‘C’ were not able to get any computer.

runCustomerSimulation(1, “ABCBCADEED”) should return 3 as ‘B’, ‘C’ and ‘E’ were not able to get any computer.

Source: Fiberlink (maas360) Interview

We strongly recommend to minimize your browser and try this yourself first.

Below are simple steps to find number of customers who could not get any computer.

1) Initialize result as 0.

2) Traverse the given sequence. While traversing, keep track of occupied computers (this can be done by keeping track of characters which have appeared only once and a computer was available when they appeared). At any point, if count of occupied computers is equal to ‘n’, and there is a new customer, increment result by 1.

The important thing is to keep track of existing customers in cafe in a way that can indicate whether the customer has got a computer or not. Note that in sequence “ABCBCADEED”, customer ‘B’ did not get a seat, but still in cafe as a new customer ‘C’ is next in sequence.

Below are implementations of above idea.

```java
// JAVA program to find number of
//customers who couldn't get a resource.
class GFG
{

static int MAX_CHAR = 26;

// n is number of computers in cafe.
// 'seq' is given sequence of customer entry, exit events
static int runCustomerSimulation(int n, char []seq)
{
    // seen[i] = 0, indicates that customer 'i' is not in cafe
    // seen[1] = 1, indicates that customer 'i' is in cafe but
    //         computer is not assigned yet.
    // seen[2] = 2, indicates that customer 'i' is in cafe and
    //         has occupied a computer.
    char []seen = new char[MAX_CHAR];

    // Initialize result which is number of customers who could
    // not get any computer.
    int res = 0;

    int occupied = 0; // To keep track of occupied computers

    // Traverse the input sequence
    for (int i=0; i< seq.length; i++)
    {
        // Find index of current character in seen[0...25]
        int ind = seq[i] - 'A';

        // If First occurrence of 'seq[i]'
        if (seen[ind] == 0)
        {
            // set the current character as seen
            seen[ind] = 1;

            // If number of occupied computers is less than
            // n, then assign a computer to new customer
            if (occupied < n)
            {
                occupied++;

                // Set the current character as occupying a computer
                seen[ind] = 2;
            }

            // Else this customer cannot get a computer,
            // increment result
            else
                res++;
        }

        // If this is second occurrence of 'seq[i]'
        else
        {
```

```
        // Decrement occupied only if this customer
        // was using a computer
        if (seen[ind] == 2)
            occupied--;
        seen[ind] = 0;
        }
    }
    return res;
}

// Driver program
public static void main(String[] args)
{
    System.out.println(runCustomerSimulation(2, "ABBAJJKZKZ".toCharArray()));
    System.out.println(runCustomerSimulation(3, "GACCBDDBAGEE".toCharArray()));
    System.out.println(runCustomerSimulation(3, "GACCBGDDBAEE".toCharArray()));
    System.out.println(runCustomerSimulation(1, "ABCBCA".toCharArray()));
    System.out.println(runCustomerSimulation(1, "ABCBCADEED".toCharArray()));
}
}

// This code is contributed by Princi Singh
```

Output:

```
0
1
0
2
3
```

Time complexity of above solution is O(n) and extra space required is O(CHAR_MAX) where CHAR_MAX is total number of possible characters in given sequence.

This article is contributed by **Lokesh**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

From <https://www.geeksforgeeks.org/function-to-find-number-of-customers-who-could-not-get-a-computer/>

From <https://www.geeksforgeeks.org/function-to-find-number-of-customers-who-could-not-get-a-computer/>