GeeksforGeeks

Array Matrix Strings Hashing Linked List Stack Queue Binary Tree Binary Search

Check if an array can be split into subsets of K consecutive elements

Difficulty Level : Medium • Last Updated : 21 May, 2021

Given an array **arr[]** and integer **K**, the task is to split the array into subsets of size **K**, such that each subset consists of **K** consecutive elements.

Examples:

> ***Input:*** *arr[] = {1, 2, 3, 6, 2, 3, 4, 7, 8}, K = 3*
> ***Output:*** *true*
> ***Explanation:***
> *The given array of length 9 can be split into 3 subsets {1, 2, 3}, {2, 3, 4} and {6, 7, 8} such that each subset consists of 3 consecutive elements.*
>
> ***Input:*** *arr[] = [1, 2, 3, 4, 5], K = 4*
> ***Output:*** *false*
> ***Explanation:***
> *The given array of length 5 cannot be split into subsets of 4.*

Recommended: Please try your approach on ***{IDE}*** first, before moving on to the solution.

Approach

Follow the steps to solve the problem:

- Store the frequencies of all array elements in a HashMap
- Traverse the **HashMap**.
- For every element present in the **HashMap**, check if all its occurrences can be grouped in a subsets with its **next (K – 1) consecutive elements** or not. If so, reduce the frequencies of the elements included in the subsets accordingly in the **HashMap** and proceed forward.
- If any element is found which cannot be grouped into a subset of K consecutive elements, print False. Otherwise print True.

Below is the implementation of the above approach:

C++

```
// C++ Program to implement the
// above approach
#include <bits/stdc++.h>
using namespace std;

// Function to check if a given array can
// be split into subsets of K consecutive
// elements
bool groupInKConsecutive(vector<int>& arr,
                         int K)
{
    // Stores the frequencies of
    // array elements
    }

    // Traverse the map
    for (auto c : count) {
        int cur = c.first;
        int n = c.second;

        // Check if all its occurrences can
        // be grouped into K subsets
        if (n > 0) {

            // Traverse next K elements
            for (int i = 1; i < K; ++i) {

                // If the element is not
                // present in the array
                if (!count.count(cur + i)) {
                    return false;
                }

                count[cur + i] -= n;

                // If it cannot be split into
                // required number of subsets
                if (count[cur + i] < 0)
                    return false;
            }
        }
    }

    return true;
}

// Driver Code
int main()
{
    vector<int> arr = { 1, 2, 3, 6, 2,
                        3, 4, 7, 8 };
    int k = 3;
    if (groupInKConsecutive(arr, k)) {
        cout << "True";
```

```
}
```

Java

```
// Java Program to implement the
// above approach
import java.util.*;
class GFG{

// Function to check if a given array can
// be split into subsets of K consecutive
// elements
static boolean groupInKConsecutive(int[] arr,
                                    int K)
{
    // Stores the frequencies of
    // array elements
    HashMap<Integer,
              Integer> count = new HashMap<Integer,
                                           Integer>();

    for (int h : arr)
    {
        if(count.containsKey(h))
            count.put(h, count.get(h) + 1);
        else
            count.put(h, 1);
    }

    // Traverse the map
    for (Map.Entry<Integer,
                    Integer> c : count.entrySet())
    {
        int cur = c.getKey();
        int n = c.getValue();

        // Check if all its occurrences can
        // be grouped into K subsets
        if (n > 0)
        {
                // If the element is not
                // present in the array
                if (!count.containsKey(cur + i))
                {
                    return false;
                }

                count.put(cur + i, count.get(cur + i) - n);

                // If it cannot be split into
                // required number of subsets
                if (count.get(cur + i) < 0)
                    return false;
            }
        }
    }
    return true;
}

// Driver Code
public static void main(String[] args)
{
    int[] arr = { 1, 2, 3, 6, 2,
                     3, 4, 7, 8 };
    int k = 3;
    if (groupInKConsecutive(arr, k))
    {
        System.out.print("True");
    }
    else
    {
        System.out.print("False");
    }
}
}

// This code contributed by sapnasingh4991
```

Python3

```
from collections import defaultdict

# Function to check if a given array can
# be split into subsets of K consecutive
# elements
def groupInKConsecutive(arr, K):

    # Stores the frequencies of
    # array elements
    count = defaultdict(int)

    for h in arr:
        count[h] += 1

    # Traverse the map
    for key, value in count.items():
        cur = key
        n = value

        # Check if all its occurrences can
        # be grouped into K subsets
        if (n > 0):

            # Traverse next K elements
            for i in range(1, K):

                # If the element is not
                # present in the array
                if ((cur + i) not in count):
                    return False

                count[cur + i] -= n

                # If it cannot be split into
                # required number of subsets
                if (count[cur + i] < 0):
                    return False

    return True

# Driver Code
```

```
    k = 3

    if (groupInKConsecutive(arr, k)):
        print("True")
    else:
        print("False")

# This code is contributed by chitranayal
```

C#

```
// C# program to implement the
// above approach
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

class GFG{

// Function to check if a given array can
// be split into subsets of K consecutive
// elements
static bool groupInKConsecutive(int[] arr,
                                int K)
{

    // Stores the frequencies of
    // array elements
    Dictionary<int,
               int> count = new Dictionary<int,
                                           int>();

    foreach(int h in arr)
    {
        if (count.ContainsKey(h))
            count[h]++;
        else
            count[h] = 1;
    }
```

```
            int cur = c;
            int n = count;

            // Check if all its occurrences can
            // be grouped into K subsets
            if (n > 0)
            {

                // Traverse next K elements
                for(int i = 1; i < K; ++i)
                {

                    // If the element is not
                    // present in the array
                    if (!count.ContainsKey(cur + i))
                    {
                        return false;
                    }

                    count[cur + i] -= n;

                    // If it cannot be split into
                    // required number of subsets
                    if (count[cur + i] < 0)
                        return false;
                }
            }
        }
        return true;
    }

    // Driver Code
    public static void Main(string[] args)
    {
        int[] arr = { 1, 2, 3, 6, 2,
                      3, 4, 7, 8 };
        int k = 3;
        if (groupInKConsecutive(arr, k))
        {
            Console.Write("True");
        }
```

```
}
}

// This code is contributed by rutvik_56
```

Javascript

```
<script>

// Javascript Program to implement the
// above approach

// Function to check if a given array can
// be split into subsets of K consecutive
// elements
function groupInKConsecutive(arr, K)
{
    // Stores the frequencies of
    // array elements
    var count = new Map();

    arr.forEach(element => {
        if(count.has(element))
            count.set(element, count.get(element)+1)
        else
            count.set(element, 1)

    });

    // Traverse the map

    count.forEach((value, key) => {

        var cur = key;
        var n = value;

        // Check if all its occurrences can
        // be grouped into K subsets
        if (n > 0) {
                // present in the array
                if (!count.has(cur + i)) {
                    return false;
                }

                count.set(cur + i, count.get(cur+i)-n);

                // If it cannot be split into
                // required number of subsets
                if (count.get(cur + i) < 0)
                    return false;
            }
        }
    });

    return true;
}

// Driver Code
var arr = [1, 2, 3, 6, 2,
                    3, 4, 7, 8];
var k = 3;
if (groupInKConsecutive(arr, k)) {
    document.write( "True");
}
else {
    document.write( "False");
}


</script>
```

Output:

```
True
```

Time Complexity: *O(N*log(N))*

Liked 2

Previous

Length of longest subsequence whose XOR value is odd

Next

Count of binary strings of length N having equal count of 0's and 1's

RECOMMENDED ARTICLES

Page : 1 2 3

01 Maximum number of subsets an array can be split into such that product of their minimums with size of subsets is at least K
21, Apr 21

02 Minimum length of a rod that can be split into N equal parts that can further be split into given number of equal parts
16, May 21

03 Split array into minimum number of subsets such that elements of all pairs are present in different subsets at least once
04, Jan 21

04 Check if an array can be split into K consecutive non-overlapping subarrays of length M consisting of single distinct element
12, Feb 21

05 Maximize count of subsets into which the given array can be split such that it satisfies the given condition
03, Feb 21

06 Split an Array A[] into Subsets having equal Sum and sizes equal to elements of Array B[]
21, Aug 20

07 Check if it is possible to split given Array into K odd-sum subsets
19, Jun 20

08 Check if it’s possible to split the Array into strictly increasing subsets of size at least K
05, Jul 21

Article Contributed By :

coder001
@coder001

Easy Normal Medium Hard Expert

Improved By : ukasp, sapnasingh4991, rutvik_56, famously

Article Tags : cpp-map, frequency-counting, subset, Arrays, Hash, Searching, Sorting

Practice Tags : Arrays, Searching, Hash, Sorting, subset

Improve Article

Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Career

In Media

Contact Us

Privacy Policy

Copyright Policy

Learn

Algorithms

Structures

SDE Cheat Sheet

Machine learning

CS Subjects

Video Tutorials

Courses

News

Top News

Technology

Work & Career

Business

Languages

Python

Java

CPP

Golang

Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Bootstrap

ReactJS

NodeJS

Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship