Longest Consecutive Subsequence

Difficulty Level : Medium • Last Updated : 19 Jul, 2022

Given an array of integers, find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers, the consecutive numbers can be in any order.

Examples:

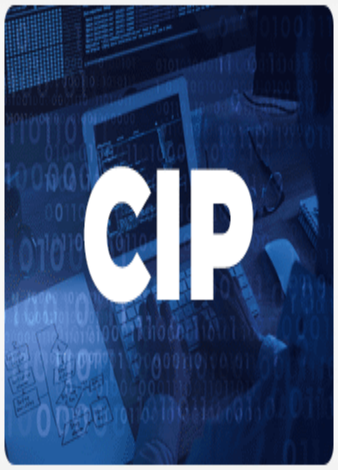
```
Input: arr[] = {1, 9, 3, 10, 4, 20, 2}
Output: 4
Explanation:
```

```
The subsequence 1, 3, 4, 2 is the longest
subsequence of consecutive elements

Input: arr[] = {36, 41, 56, 35, 44, 33, 34, 92, 43, 32, 42}
Output: 5
Explanation:
The subsequence 36, 35, 33, 34, 32 is the longest
subsequence of consecutive elements.
```

Recommended Practice
Longest consecutive subsequence
Try It!

Naive Approach: The idea is to first sort the array and find the longest subarray with consecutive elements.
After sorting the array and removing the multiple occurrences of elements, run a loop and keep a count and max (both initially zero). Run a loop from start to end and if the current element is not equal to the previous (element+1) then set the count to 1 else increase the count. Update max with a maximum of count and max.



C++

```
// C++ program to find longest
// contiguous subsequence
#include <bits/stdc++.h>
using namespace std;

// Returns length of the longest
// contiguous subsequence
int findLongestConseqSubseq(int arr[], int n)
{
    int ans = 0, count = 0;

    // sort the array
    sort(arr, arr + n);

    vector<int> v;
    v.push_back(arr[0]);

    //insert repeated elements only once in the vector
    for (int i = 1; i < n; i++)
    {
        if (arr[i] != arr[i - 1])
            v.push_back(arr[i]);
    }
    // find the maximum length
    // by traversing the array
    for (int i = 0; i < v.size(); i++)
    {

        // Check if the current element is equal
        // to previous element +1
        if (i > 0 && v[i] == v[i - 1] + 1)
            count++;
        // reset the count
        else
            count = 1;

        // update the maximum
        ans = max(ans, count);
    }
    return ans;
}

// Driver code
int main()
{
    int arr[] = { 1, 2, 2, 3 };
    int n = sizeof arr / sizeof arr[0];
    cout << "Length of the Longest contiguous subsequence "
            "is "
         << findLongestConseqSubseq(arr, n);
    return 0;
}
```

Java

```
// Java program to find longest
// contiguous subsequence
import java.io.*;
import java.util.*;

class GFG
{

    static int findLongestConseqSubseq(int arr[],
                                       int n)
    {

        // Sort the array
        Arrays.sort(arr);

        int ans = 0, count = 0;

        ArrayList<Integer> v = new ArrayList<Integer>();
        v.add(10);

        // Insert repeated elements
        // only once in the vector
        for (int i = 1; i < n; i++)
        {
            if (arr[i] != arr[i - 1])
                v.add(arr[i]);
        }
        // Find the maximum length
        // by traversing the array
        for (int i = 0; i < v.size(); i++)
        {

            // Check if the current element is
            // equal to previous element +1
            if (i > 0 &&v.get(i) == v.get(i - 1) + 1)
                count++;
            else
                count = 1;

            // Update the maximum
            ans = Math.max(ans, count);
        }
        return ans;
    }

    // Driver code
    public static void main(String[] args)
    {
        int arr[] = { 1, 9, 3, 10, 4, 20, 2 };
        int n = arr.length;

        System.out.println(
            "Length of the Longest "
            + "contiguous subsequence is "
            + findLongestConseqSubseq(arr, n));
    }
}

// This code is contributed by parascoding
```

Python3

```
# Python3 program to find longest
# contiguous subsequence

# Returns length of the longest
# contiguous subsequence
def findLongestConseqSubseq(arr, n):

    ans = 0
    count = 0

    # Sort the array
    arr.sort()

    v = []

    v.append(arr[0])

    # Insert repeated elements only
    # once in the vector
    for i in range(1, n):
        if (arr[i] != arr[i - 1]):
            v.append(arr[i])

    # Find the maximum length
    # by traversing the array
    for i in range(len(v)):

        # Check if the current element is
        # equal to previous element +1
        if (i > 0 and v[i] == v[i - 1] + 1):
            count += 1

        # Reset the count
        else:
            count = 1

        # Update the maximum
        ans = max(ans, count)

    return ans

# Driver code
arr = [ 1, 2, 2, 3 ]
n = len(arr)

print("Length of the Longest contiguous subsequence is",
       findLongestConseqSubseq(arr, n))

# This code is contributed by avanitrachhadiya2155
```

C#

```
// C# program to find longest
// contiguous subsequence
using System;
using System.Collections.Generic;

class GFG{

static int findLongestConseqSubseq(int[] arr,
                                   int n)
{

    // Sort the array
    Array.Sort(arr);

    int ans = 0, count = 0;

    List<int> v = new List<int>();
    v.Add(10);

    // Insert repeated elements
    // only once in the vector
    for(int i = 1; i < n; i++)
    {
        if (arr[i] != arr[i - 1])
            v.Add(arr[i]);
    }

    // Find the maximum length
    // by traversing the array
    for(int i = 0; i < v.Count; i++)
    {

        // Check if the current element is
        // equal to previous element +1
        if (i > 0 && v[i] == v[i - 1] + 1)
            count++;
        else
            count = 1;

        // Update the maximum
        ans = Math.Max(ans, count);
    }
    return ans;
}
// Driver code
static void Main()
{
    int[] arr = { 1, 9, 3, 10, 4, 20, 2 };
    int n = arr.Length;

    Console.WriteLine("Length of the Longest " +
                      "contiguous subsequence is " +
                      findLongestConseqSubseq(arr, n));
}
}

// This code is contributed by divyeshrabadiya07
```

Javascript

```
<script>

    // JavaScript program to find longest
    // contiguous subsequence

    // Returns length of the longest
    // contiguous subsequence
    function findLongestConseqSubseq(arr, n) {
        let ans = 0, count = 0;

        // sort the array
        arr.sort(function (a, b) { return a - b; })

        var v = [];
        v.push(arr[0]);

        //insert repeated elements only once in the vector
        for (let i = 1; i < n; i++) {
            if (arr[i] != arr[i - 1])
                v.push(arr[i]);
        }
        // find the maximum length
        // by traversing the array
        for (let i = 0; i < v.length; i++) {

            // Check if the current element is equal
            // to previous element +1
            if (i > 0 && v[i] == v[i - 1] + 1)
```

```
                count++;
            // reset the count
            else
                count = 1;

            // update the maximum
            ans = Math.max(ans, count);
        }
        return ans;
    }

    // Driver code

    let arr = [1, 2, 2, 3];
    let n = arr.length;
    document.write(
    "Length of the Longest contiguous subsequence is "
    +findLongestConseqSubseq(arr, n)
    );

 // This code is contributed by Potta Lokesh

</script>
```

Output

```
Length of the Longest contiguous subsequence is 3
```

Complexity Analysis:

- **Time complexity:** O(nLogn).
  Time to sort the array is O(nlogn).
- **Auxiliary space :** O(n).
  As extra space is needed for storing in vector v.

Thanks to Hao.W for suggesting the above solution.

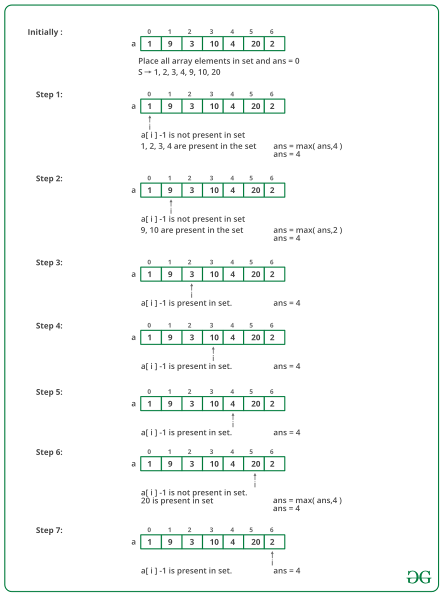
Efficient solution:

This problem can be solved in O(n) time using an **Efficient Solution**. The idea is to use Hashing. We first insert all elements in a Set. Then check all the possible starts of consecutive subsequences.

Algorithm:

1. Create an empty hash.
2. Insert all array elements to hash.
3. Do following for every element arr[i]
4. Check if this element is the starting point of a subsequence. To check this, simply look for arr[i] – 1 in the hash, if not found, then this is the first element a subsequence.
5. If this element is the first element, then count the number of elements in the consecutive starting with this element. Iterate from arr[i] + 1 till the last element that can be found.
6. If the count is more than the previous longest subsequence found, then update this.

Below image is a dry run of the above approach:



Initially :
a 1 9 3 10 4 20 2
Place all array elements in set and ans = 0
S → 1, 2, 3, 4, 9, 10, 20
Step 1:
a[i] -1 is not present in set
1, 2, 3, 4 are present in the set
ans = max(ans,4)
ans = 4
Step 2:
a[i] -1 is not present in set
9, 10 are present in the set
ans = max(ans,2)
ans = 4
Step 3:
a[i] -1 is present in set.
ans = 4
Step 4:
a[i] -1 is present in set.
ans = 4
Step 5:
a[i] -1 is present in set.
ans = 4
Step 6:
a[i] -1 is not present in set.
20 is present in set
ans = max(ans,4)
ans = 4
Step 7:
a[i] -1 is present in set.
ans = 4

Below is the implementation of the above approach:

C++

```
// C++ program to find longest
// contiguous subsequence
```

```
#include <bits/stdc++.h>
using namespace std;

// Returns length of the longest
// contiguous subsequence
int findLongestConseqSubseq(int arr[], int n)
{
    unordered_set<int> S;
    int ans = 0;

    // Hash all the array elements
    for (int i = 0; i < n; i++)
        S.insert(arr[i]);

    // check each possible sequence from
    // the start then update optimal length
    for (int i = 0; i < n; i++)
    {
        // if current element is the starting
        // element of a sequence
        if (S.find(arr[i] - 1) == S.end())
        {
            // Then check for next elements
            // in the sequence
            int j = arr[i];
            while (S.find(j) != S.end())
                j++;

            // update  optimal length if
            // this length is more
            ans = max(ans, j - arr[i]);
        }
    }
    return ans;
}

// Driver code
int main()
{
    int arr[] = { 1, 9, 3, 10, 4, 20, 2 };
    int n = sizeof arr / sizeof arr[0];
    cout << "Length of the Longest contiguous subsequence "
            "is "
         << findLongestConseqSubseq(arr, n);
    return 0;
}
```

Java

```
// Java program to find longest
// consecutive subsequence
import java.io.*;
import java.util.*;

class ArrayElements {
    // Returns length of the longest
    // consecutive subsequence
    static int findLongestConseqSubseq(int arr[], int n)
    {
        HashSet<Integer> S = new HashSet<Integer>();
        int ans = 0;

        // Hash all the array elements
        for (int i = 0; i < n; ++i)
            S.add(arr[i]);

        // check each possible sequence from the start
        // then update optimal length
        for (int i = 0; i < n; ++i)
        {
            // if current element is the starting
            // element of a sequence
            if (!S.contains(arr[i] - 1))
            {
                // Then check for next elements
                // in the sequence
                int j = arr[i];
                while (S.contains(j))
                    j++;

                // update  optimal length if this
                // length is more
                if (ans < j - arr[i])
                    ans = j - arr[i];
            }
        }
        return ans;
    }
    // Driver Code
    public static void main(String args[])
    {
        int arr[] = { 1, 9, 3, 10, 4, 20, 2 };
        int n = arr.length;
        System.out.println(
            "Length of the Longest consecutive subsequence is "
            + findLongestConseqSubseq(arr, n));
    }
}
// This code is contributed by Aakash Hasija
```

Python3

```
# Python program to find longest contiguous subsequence


def findLongestConseqSubseq(arr, n):

    s = set()
    ans = 0

    # Hash all the array elements
    for ele in arr:
        s.add(ele)

    # check each possible sequence from the start
    # then update optimal length
    for i in range(n):

         # if current element is the starting
        # element of a sequence
        if (arr[i]-1) not in s:

            # Then check for next elements in the
            # sequence
            j = arr[i]
            while(j in s):
                j += 1

            # update  optimal length if this length
            # is more
            ans = max(ans, j-arr[i])
    return ans


# Driver code
if __name__ == '__main__':
    n = 7
    arr = [1, 9, 3, 10, 4, 20, 2]
    print ("Length of the Longest contiguous subsequence is ",findLongest

# Contributed by: Harshit Sidhwa
```

C#

```
using System;
using System.Collections.Generic;

// C# program to find longest consecutive subsequence

public class ArrayElements {
    // Returns length of the
    // longest consecutive subsequence
    public static int findLongestConseqSubseq(int[] arr,
                                              int n)
    {
        HashSet<int> S = new HashSet<int>();
        int ans = 0;

        // Hash all the array elements
        for (int i = 0; i < n; ++i) {
            S.Add(arr[i]);
        }

        // check each possible sequence from the start
        // then update optimal length
        for (int i = 0; i < n; ++i)
        {
            // if current element is the starting
            // element of a sequence
            if (!S.Contains(arr[i] - 1))
            {
                // Then check for next elements in the
                // sequence
                int j = arr[i];
                while (S.Contains(j))
                {
                    j++;
                }

                // update  optimal length if this length
                // is more
                if (ans < j - arr[i])
                {
                    ans = j - arr[i];
                }
            }
        }
        return ans;
    }

    // Driver code
    public static void Main(string[] args)
    {
        int[] arr = new int[] { 1, 9, 3, 10, 4, 20, 2 };
        int n = arr.Length;
        Console.WriteLine(
            "Length of the Longest consecutive subsequence is "
            + findLongestConseqSubseq(arr, n));
    }
}

// This code is contributed by Shrikant13
```

Javascript

```
<script>
// Javascript program to find longest
// contiguous subsequence


// Returns length of the longest
// contiguous subsequence
function findLongestConseqSubseq(arr, n) {
    let S = new Set();
    let ans = 0;

    // Hash all the array elements
    for (let i = 0; i < n; i++)
```

```
            S.add(arr[i]);

        // check each possible sequence from
        // the start then update optimal length
        for (let i = 0; i < n; i++)
        {

            // if current element is the starting
            // element of a sequence
            if (!S.has(arr[i] - 1))
            {

                // Then check for next elements
                // in the sequence
                let j = arr[i];
                while (S.has(j))
                    j++;

                // update optimal length if
                // this length is more
                ans = Math.max(ans, j - arr[i]);
            }
        }
        return ans;
    }

    // Driver code
    let arr = [1, 9, 3, 10, 4, 20, 2];
    let n = arr.length;
    document.write("Length of the Longest contiguous subsequence is "
        + findLongestConseqSubseq(arr, n));

        // This code is contributed by gfgking.
    </script>
```

Output

```
Length of the Longest contiguous subsequence is 4
```

Complexity Analysis:

- **Time complexity:** O(n).
  Only one traversal is needed and the time complexity is O(n) under

the assumption that hash insert and search take O(1) time.

- **Auxiliary space:** O(n).

  To store every element in hashmap O(n) space is needed

  Thanks to Gaurav Ahirwar for the above solution.

Another Solution:

This problem can be solved in **O(N log N)** time with another **Method**, this time the Idea is to use Priority Queue.

Algorithm:

1. Create a Priority Queue to store the element
2. Store the first element in a variable
3. Remove it from the Priority Queue
4. Check the difference between this removed first element and the new peek element
5. If the difference is equal to 1 increase count by 1 and repeats step 2 and step 3
6. If the difference is greater than 1 set counter to 1 and repeat step 2 and step 3
7. if the difference is equal to 0 repeat step 2 and 3
8. if counter greater than the previous maximum then store counter to maximum
9. Continue step 4 to 7 until we reach the end of the Priority Queue
10. Return the maximum value

C++

```
// CPP program for the above approach
#include <bits/stdc++.h>
using namespace std;

int findLongestConseqSubseq(int arr[], int N)
{
    priority_queue<int, vector<int>, greater<int> > pq;
    for (int i = 0; i < N; i++) {
        // adding element from
        // array to PriorityQueue
        pq.push(arr[i]);
    }

    // Storing the first element
    // of the Priority Queue
    // This first element is also
    // the smallest element
    int prev = pq.top();
    pq.pop();

    // Taking a counter variable with value 1
    int c = 1;

    // Storing value of max as 1
    // as there will always be
    // one element
    int max = 1;
    while (!pq.empty()) {

        // check if current peek
        // element minus previous
        // element is greater than
        // 1 This is done because
        // if it's greater than 1
        // then the sequence
        // doesn't start or is broken here
        if (pq.top() - prev > 1) {

            // Store the value of counter to 1
            // As new sequence may begin
            c = 1;

            // Update the previous position with the
            // current peek And remove it
            prev = pq.top();
            pq.pop();
        }

        // Check if the previous
        //  element and peek are same
        else if (pq.top() - prev == 0) {
            // Update the previous position with the
            // current peek And remove it
            prev = pq.top();
            pq.pop();
        }

        // If the difference
        // between previous element and peek is 1
        else {

            // Update the counter
            // These are consecutive elements
            c++;

            // Update the previous position
            //  with the current peek And remove it
            prev = pq.top();
            pq.pop();
        }

        // Check if current longest
        // subsequence is the greatest
        if (max < c) {

            // Store the current subsequence count as
            // max
            max = c;
        }
    }
    return max;
}

// Driver Code
int main()
{
    int arr[] = { 1, 9, 3, 10, 4, 20, 2 };
    int n = 7;

    cout << "Length of the Longest consecutive subsequence "
            "is "
         << findLongestConseqSubseq(arr, n);
    return 0;
}
// this code is contributed by Manu Pathria
```

Java

```java
// Java Program to find longest consecutive
// subsequence This Program uses Priority Queue
import java.io.*;
import java.util.PriorityQueue;
public class Longset_Sub
{
    // return the length of the longest
    // subsequence of consecutive integers
    static int findLongestConseqSubseq(int arr[], int N)
    {

        PriorityQueue<Integer> pq
            = new PriorityQueue<Integer>();
        for (int i = 0; i < N; i++)
        {
            // adding element from
            // array to PriorityQueue
            pq.add(arr[i]);
        }

        // Storing the first element
        // of the Priority Queue
        // This first element is also
        // the smallest element
        int prev = pq.poll();

        // Taking a counter variable with value 1
        int c = 1;

        // Storing value of max as 1
        // as there will always be
        // one element
        int max = 1;

        for (int i = 1; i < N; i++)
        {
            // check if current peek
            // element minus previous
            // element is greater than
            // 1 This is done because
            // if it's greater than 1
            // then the sequence
```

```
                // doesn't start or is broken here
                if (pq.peek() - prev > 1)
                {
                    // Store the value of counter to 1
                    // As new sequence may begin
                    c = 1;

                    // Update the previous position with the
                    // current peek And remove it
                    prev = pq.poll();
                }

                // Check if the previous
                //  element and peek are same
                else if (pq.peek() - prev == 0)
                {
                    // Update the previous position with the
                    // current peek And remove it
                    prev = pq.poll();
                }
                // if the difference
                // between previous element and peek is 1
                else
                {
                    // Update the counter
                    // These are consecutive elements
                    c++;

                    // Update the previous position
                    //  with the current peek And remove it
                    prev = pq.poll();
                }

                // Check if current longest
                // subsequence is the greatest
                if (max < c)
                {
                    // Store the current subsequence count as
                    // max
                    max = c;
                }
            }

            return max;
        }
```

```
    // Driver Code
    public static void main(String args[])
        throws IOException
    {
        int arr[] = { 1, 9, 3, 10, 4, 20, 2 };
        int n = arr.length;
        System.out.println(
            "Length of the Longest consecutive subsequence is "
            + findLongestConseqSubseq(arr, n));
    }
}
// This code is contributed by Sudipa Sarkar
```

Python3

```
# Python program for the above approach
import bisect

def findLongestConseqSubseq(arr,N):
    pq = []
    for i in range(N):

        # adding element from
        # array to PriorityQueue
        bisect.insort(pq,arr[i])

    # Storing the first element
    # of the Priority Queue
    # This first element is also
    # the smallest element
    prev = pq[0]
    pq.pop(0)

    # Taking a counter variable with value 1
    c=1

    # Storing value of max as 1
    # as there will always be
    # one element
    max = 1
    while(len(pq)):
        # check if current peek
        # element minus previous
```

```
            # element is greater than
            # 1 This is done because
            # if it's greater than 1
            # then the sequence
            # doesn't start or is broken here
            if(pq[0] - prev > 1):
                # Store the value of counter to 1
                # As new sequence may begin
                c = 1

                # Update the previous position with the
                # current peek And remove it
                prev = pq[0]
                pq.pop(0)

            # Check if the previous
            # element and peek are same
            elif(pq[0] - prev == 0):
                # Update the previous position with the
                # current peek And remove it
                prev = pq[0]
                pq.pop(0)

            # If the difference
            # between previous element and peek is 1
            else:
                # Update the counter
                # These are consecutive elements
                c = c +1
                # Update the previous position
                # with the current peek And remove it
                prev = pq[0]
                pq.pop(0)

            # Check if current longest
            # subsequence is the greatest
            if(max < c):
                # Store the current subsequence count as
                # max
                max = c
        return max

# Driver Code
arr = [1, 9, 3, 10, 4, 20, 2]
n = 7
```

```
print("Length of the Longest consecutive subsequence is {}".format(findLo


# This code is contributed by Pushpesh Raj
```

C#

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

class GFG
{

  // return the length of the longest
  // subsequence of consecutive integers
  static int findLongestConseqSubseq(int[] arr, int N)
  {

    List<int> pq = new List<int>();
    for (int i = 0; i < N; i++)
    {

      // adding element from
      // array to PriorityQueue
      pq.Add(arr[i]);
      pq.Sort();
    }

    // Storing the first element
    // of the Priority Queue
    // This first element is also
    // the smallest element
    int prev = pq[0];

    // Taking a counter variable with value 1
    int c = 1;

    // Storing value of max as 1
    // as there will always be
    // one element
    int max = 1;
    for (int i = 1; i < N; i++)
    {

      // check if current peek
      // element minus previous
      // element is greater than
      // 1 This is done because
      // if it's greater than 1
      // then the sequence
      // doesn't start or is broken here
      if (pq[0] - prev > 1)
      {
        // Store the value of counter to 1
        // As new sequence may begin
        c = 1;

        // Update the previous position with the
        // current peek And remove it
        prev = pq[0];
        pq.RemoveAt(0);
      }

      // Check if the previous
      //  element and peek are same
      else if (pq[0] - prev == 0)
      {

        // Update the previous position with the
        // current peek And remove it
        prev = pq[0];
        pq.RemoveAt(0);
      }

      // if the difference
      // between previous element and peek is 1
      else
      {

        // Update the counter
        // These are consecutive elements
        c++;

        // Update the previous position
        //  with the current peek And remove it
        prev = pq[0];
```

```
          pq.RemoveAt(0);
        }

        // Check if current longest
        // subsequence is the greatest
        if (max < c)
        {

          // Store the current subsequence count as
          // max
          max = c;
        }
      }

      return max;
    }

    // Driver Code
    public static void Main()
    {
      int[] arr = { 1, 9, 3, 10, 4, 20, 2 };
      int n = arr.Length;
      Console.WriteLine(
        "Length of the Longest consecutive subsequence is "
        + findLongestConseqSubseq(arr, n));
    }
  }

// This code is contributed by code_hunt.
```

Output

```
Length of the Longest consecutive subsequence is 4
```

Time Complexity : O(n*logn)

Auxiliary Space: O(n)

This article is contributed by **Aarti_Rathi**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the

GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Liked 154

Previous

Check if an array can be divided into pairs whose sum is divisible by k

Next

Longest Increasing consecutive subsequence

RECOMMENDED ARTICLES

Page : 1 2 3

01 Longest subsequence with consecutive English alphabets
17, Jun 21

05 Longest Increasing consecutive subsequence
13, May 18

06 Longest subsequence such that every element in the subsequence is formed by

02 Longest subsequence such that

no 3 consecutive characters are same
14, Dec 21

multiplying previous element with a prime
12, Jul 19

03 Longest Increasing consecutive subsequence | Set-2
02, Feb 22

04 Printing longest Increasing consecutive subsequence
13, May 18

07 Longest Subsequence with absolute difference of pairs as at least Subsequence's maximum
24, Feb 22

08 Maximize product of digit sum of consecutive pairs in a subsequence of length K
05, May 20

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : shrikanth13, andrew1234, parascoding, agrawalvaibhaw96, akashgoraya1, avanitrachhadiya2155, divyeshrabadiya07, sudipasarkar999, manupathria, revathi727, lokeshpotta20, gfgking, CoderSaty, amartyaghoshgfg, code_hunt, pushpeshrajdx01, codewithmini, mitalibhola94, dhruvbis06jj

Article Tags : Amazon, Linkedin, subsequence, Walmart, Zoho, Arrays, Hash, Sorting

Practice Tags : Zoho, Amazon, Walmart, Linkedin, Arrays, Hash, Sorting

Improve Article Report Issue

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

Load Comments

NodeJS