Check whether the sequence is a subsequence of another

Asked 3 years ago Active 2 years, 5 months ago Viewed 1k times

0

1

There are two sequences of numbers. Check whether the second sequence is a subsequence of the first sequence. The sequence B[0], … , B[m] is a subsequence of the sequence A[0], … , A[n], if there exist numbers 0≤ i_0< i_1< …< i_m≤n such that for all k from 0 to m one has B[k]=A[i_k].

Input:

An array of integers:

1. Quantity of the members of the first sequence.
2. Members of the first sequence.
3. Quantity of the members of the second sequence.
4. Members of the second sequence.

Output:

An integer:

· Either one if the second sequence is a subsequence of the first sequence,

· Or zero if not.

My solution:

```
class Program
{
    static void Main(string[] args)
    {
        var input = Console.ReadLine();
        var numbers = input.Split(' ').Select(x => int.Parse(x)).ToArray();
        int a = numbers[0];
```

```
            int[] arr1 = numbers.Skip(1).Take(a).ToArray();
            int b = numbers.Skip(a + 1).Take(1).ToArray()[0];
            int[] arr2 = numbers.Skip(a + 2).Take(b).ToArray();
            if(arr1.Length==0 || arr2.Length==0) Console.Write("0");
            else Console.Write(contains(arr1, arr2));
        }

        private static int contains(int[] arr1, int[] arr2)
        {
            int check = 1;
            int[] checkArray = arr1;
            for(int i=0;i<arr2.Length;i++)
            {
                if (!checkArray.Contains(arr2[i])) return 0;
                checkArray = checkArray.Skip(Array.FindIndex(checkArray, x =>
x==arr2[i])).ToArray();
            }
            return check;
        }
```

It runs on automatic tester, and returns correct result on every teste except a single one. I don't know what is input data for this test, and i can't find it out. What can cause such problem and what would be solution for it?

UPD: Examples from the description of the task Examples:

Input: 5 1 2 3 4 5 2 1 4

Output: 1

Input: 5 1 2 3 4 5 2 4 1

Output: 0

c#

Share Follow

edited Feb 3, 2019 at 11:58

asked Feb 2, 2019 at 18:30

Stressful Courtier

1 3

3 Answers

Active Oldest Score

0

This returns 1 : 3 1 2 3 3 1 1 3

1. First array length: 3
2. First array: 1 2 3
3. Second array length: 3
4. Second array: 1 1 3

The second array should not be a subsequence by your definition. I think the reason is `FindIndex` returns a 0 based index, so you `Skip` should add 1 to the returned value to skip the correct number.

You also don't need to do both a `Contains` and a `FindIndex`. `FindIndex` returns -1 if index is not there and that prevents unnecessarily iterations over the array. It also has an overload that takes the range to search in so you don't need to use `skip` And `take`. See the docs here

Share Follow edited Feb 2, 2019 at 19:23 answered Feb 2, 2019 at 18:54 Sanjid **85** 1 6

According to the description, elements in subsequence don't have to be in consecutive positions in the first array. Here are examples, provided with the description: Input: 5 1 2 3 4 5 2 1 4 Output: 1 Input: 5 1 2 3 4 5 2 4 1 Output: 0 – Stressful Courtier Feb 2, 2019 at 19:01

Ah my bad. I assumed subsequence meant in order. What did you mean by this bit "there exist numbers 0≤ i_0"? – Sanjid Feb 2, 2019 at 19:09

Updated the answer. – Sanjid Feb 2, 2019 at 19:24

Looks like i accidentally removed a part of text while editing the original post, now it should be fine – Stressful Courtier Feb 3, 2019 at 12:00

0

oh! i had the same problem)) i found solution! its correct version on c++

```
#include <iostream>
#include <vector>
#include <stdio.h>
#include <string>
#include <sstream>

using namespace std;
int main()
{
    int len, len2, t=0, i=0;

    std::string sentence;
    std::cout.flush();

    std::getline(std::cin,sentence);
    std::istringstream iss(sentence);

    std::vector<double> words;
    std::vector<double> SubSeq;

    vector<double>::iterator it;
    double word;
    while(iss >> word)
    {
        words.push_back(word);
    }
    len= static_cast<int>(words.front());
    it = words.begin();
    words.erase(it);
    words.shrink_to_fit();

    for( i=len; i<words.size(); i++ )
    {
           SubSeq.insert(SubSeq.end(),words[i]);
    }
    it = words.begin()+len;
    words.erase(it,words.end());
    words.shrink_to_fit();


    len2= static_cast<int>(SubSeq.front());
    it = SubSeq.begin();
    SubSeq.erase(it);
    SubSeq.shrink_to_fit();

    int j;
```

```
        for(i=0 , j=0; i<len && j<len2; i++ )
        {
            if( SubSeq[j]== words[i]){
                j++;
            }
        }

        if( j == len2)
            cout<<1;
        else
            cout<<0;
      return 0;
    }
```

i change this part

```
  /*
   i=0;
            for(it=words.begin(); it!=words.end(); it++ )
            {
                if( SubSeq[i]== *it){
                    i++;
            }
  */
```

and paste this

```
  for(i=0 , j=0; i<len && j<len2; i++ )
          {
              if( SubSeq[j]== words[i]){
                  j++;
              }
```

Share Follow

answered Feb 28, 2019 at 11:47

Vova Denys
34 2

The core logic of this problem is to look forward through both sequences. As you find an instance of the next element in the first

0

sequence in the second, you can advance in the first. The critical step that you're missing is that when you find a match, you need to "consume" the element of the second by incrementing its iterator.

```
checkArray = checkArray.Skip(Array.FindIndex(checkArray, x => x==arr2[i])).ToArray();
```

This line won't change the contents of `checkArray` when `FindIndex` returns 0, so you can keep matching on the first element. As a result, your algorithm will report that "2 2 2 2" is a substring of "1 2 3".

I'm generally a fan of the Linq extensions, but I think it really obfuscates the code here. I would just write a loop to process the arrays.

Share Follow

answered Sep 5, 2019 at 16:27

bmm6o
5,849 3 27 51