Check if array contains exact same sequence as other array

Asked 5 years, 4 months ago Active 5 years, 4 months ago Viewed 2k times

3

1

I have a question, I need to check if some array is part of greater array, it would be rather easy but I need to check if the greater array contains exact same sequence. For example

```
int[] greaterArray = {8, 3, 4, 5, 9, 12, 6 ... n - elements}
int[] lesserArray = {3, 4, 5}
```

Now I need to know if lesser array is part of this array but with same sequence so It it contains 3, 4, 5 next to each other in greater array.

I tried:

```
var exists = greaterArray.Intersect(lesserArray).Any();
```

But it return me information if any element of lesser array exists in greater array, not exact sequence. Any ideas?

c# arrays

Share Follow

edited Oct 26, 2016 at 9:12

User_Targaryen

3,812 3 24 47

asked Oct 26, 2016 at 8:44

Wojciech Szabowicz

2,780 2 31 67

1 stackoverflow.com/documentation/c%23/1429/arrays/16892/... – Jaydip Jadhav Oct 26, 2016 at 8:47

2 @JaydipJ That answer does not refer to subsets. – Steve Oct 26, 2016 at 8:57

This is just ans to what you stated in your Question Header *Check if array contains exact same sequence as other array* – Jaydip Jadhav Oct 26, 2016 at 8:59

Not my question header. Still, the fact remains; it doesn't answer the question. – Steve Oct 26, 2016 at 9:00

Absolutely not @JaydipJ. He's asking about subsequence not the same array – Tinwor Oct 26, 2016 at 9:06

5 Answers

Active | Oldest | Score

3

```
int[] greaterArray = {8, 3, 4, 5, 9, 12, 6};
int[] lesserArray = { 3, 4, 5 };
bool sequenceFound = false;

for (int i = 0; i <= greaterArray.Length - lesserArray.Length; i++)
{
    if (greaterArray.Skip(i).Take(lesserArray.Length).SequenceEqual(lesserArray))
    {
        sequenceFound = true;
        break;
    }
}

if (sequenceFound)
{
    //sequence found
}
else
{
    //sequence not found
}
```

Use the above code. It takes multiple sub-sequences from `greaterArray` of length equal to the length of `lesserArray` and matches it with `lesserArray` .

Share Follow

edited Oct 26, 2016 at 9:28

answered Oct 26, 2016 at 8:53

Akshey Bhat

7,785 1 18 20

3 You might want to replace the hardcoded 3 in `Take(3)` with `lesserArray.Length` . – heijp06 Oct 26, 2016 at 9:00

stackoverflow.com/questions/18037625/... – vivek nuna Oct 26, 2016 at 9:25

A bit more generic and without the use of LINQ:

```
int[] greaterArray = {8, 2, 4, 5, 9, 12, 3, 4, 5};
int[] lesserArray = {3, 4, 5};
for (int i = 0; i <= greaterArray.Length - lesserArray.Length; i++)
            {
                var sub = greaterArray.SubArray(i, lesserArray.Length);
                if (Enumerable.SequenceEqual(sub, lesserArray))
                {
                    Console.WriteLine("Equals!");
                }
            }
```

And this utils to get the SubArray:

```
public static T[] SubArray<T>(this T[] data, int index, int length)
        {
            T[] result = new T[length];
            Array.Copy(data, index, result, 0, length);
            return result;
        }
```

Share Follow

answered Oct 26, 2016 at 9:01

Tinwor

7,105 6 31 55

This should do your work

```
int[] grtarr = { 8, 3, 4, 5, 9, 12, 6 };
int[] lsarr = { 3, 4, 5 };

List<int> lstGrtArr = grtarr.ToList();
List<int> lstLsrArr = lsarr.ToList();

bool sequenceMatch = false;
```

```
for (int i = 0; i < grtarr.Count(); i++)
{
    if (lstGrtArr.Where(x => lstGrtArr.IndexOf(x) >=
i).Take(lstLsrArr.Count()).SequenceEqual(lstLsrArr))
    {
        sequenceMatch = true;
        break;
    }
}

if(sequenceMatch)
{
    //Do Something
}
```

Share Follow

answered Oct 26, 2016 at 9:15

Amey Kamat
181 2 12

1 stackoverflow.com/questions/18037625/... – vivek nuna Oct 26, 2016 at 9:25

0

```
static bool isPrefix(int[] source, int start_pos, int[] prefix)
{
    bool result = start_pos + prefix.Length <= source.Length;
    for (int i = 0; result && i < prefix.Length; ++i, ++start_pos)
        result = source[start_pos] == prefix[i];
    return result;
}
static bool Contains(int[] source, int[] prefix)
{
    bool result = false;
    for (int i = 0; !result && i < source.Length; ++i)
        result = source[i] == prefix[0] ? isPrefix(source, i, prefix) : false;
    return result;
}
```

Share Follow

edited Oct 26, 2016 at 9:48

answered Oct 26, 2016 at 9:42

LmTinyToon
4,161 3 22 45

-1

Use this piece of code:

```
public bool ArraysEqual<T>(T[] a1, T[] a2)
{
    if (ReferenceEquals(a1,a2))
        return true;

    if (a1 == null || a2 == null)
        return false;

    if (a1.Length != a2.Length)
        return false;

    EqualityComparer<T> comparer = EqualityComparer<T>.Default;
    for (int i = 0; i < a1.Length; i++)
    {
        if (!comparer.Equals(a1[i], a2[i])) return false;
    }
    return true;
}
```

Or if you want to use Linq and don't care too much about performance, the easiest thing is:

```
var arrays_are_the_same = Enumerable.SequenceEqual(a1, a2);
```

Share Follow

answered Oct 26, 2016 at 8:48

Dawid Rutkowski

2,543 1 29 34

How does performance differ between your 2 solutions? Its not like your `ArraysEqual<T>` contains some speedup magic (not my -1 though). – grek40 Oct 26, 2016 at 8:51

This doesn't answer the question. The OP is asking if the `lesserArray` is a subset of `greaterArray` with the elements being in the same sequence order. – Steve Oct 26, 2016 at 8:52