**Array comparator**

18188% *of* 588 *of* 773[veitsi](https://www.codewars.com/users/veitsi)

C#

* [Train Again](https://www.codewars.com/kata/array-comparator/train/csharp)
* [Next Kata](https://www.codewars.com/trainer/csharp)

Details

[Solutions](https://www.codewars.com/kata/array-comparator/solutions/csharp)

[Forks (2)](https://www.codewars.com/kata/array-comparator/forks/csharp)

[Discourse (16)](https://www.codewars.com/kata/array-comparator/discuss/csharp)

* Add to Collection
* |
* Share this kata:

You have two arrays in this kata, every array contain only unique elements. Your task is to calcualate number of elements in first array which also are in second array.

<https://www.codewars.com/kata/array-comparator/csharp>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

public static int MatchArrays(int[] v, int[] r)

{

Array.Sort(v);

Array.Sort(r);

int ans = 0;

for (int i = 0; i < r.Length; i++)

{

//Console.Write(r[i] + " ");

if (Array.BinarySearch(v, r[i]) >= 0)

{

ans++;

}

}

return ans;

}

static void Main(string[] args)

{

//int [] a = new int[] { 2, 34 };

//int[] b = new int[] { 7,8,34, 9 };

int[] a = new int[] { -1, -2, -3, -4 };

//int[] b = new int[] { -1, -2, -3, -4 };

// int[] a = new int[] { -4,-3,-2,-1 };

int[] b = new int[] { -4,-3,-2,-1 };

Console.WriteLine(MatchArrays(a, b));

//int[] arr = { -5,-4, -1};

//Console.WriteLine(binarySearch(arr, -1));

Console.ReadLine();

}

}

}

---------otra solución------------

using System.Collections.Generic;

using System.Linq;

public class Kata

{

public static int MatchArrays(int[] v, int[] r)

{

// We use a HashSet here because since a HashSet is composed of unique items and our inputs only have unique items

// This allows use to use `HashSet.Contains`, which is a constant O(1) operation, opposed to `Array.IndexOf`, which is a linear O(n) operation.

// With HashSet.Contains, our solution is done in O(n) time (~1 \* n), as opposed to O(n ^ 2) for Array.IndexOf (n \* n).

// Relevant links:

// Big O notation cheat sheet:

// http://bigocheatsheet.com/

// HashSet.Contains:

// https://msdn.microsoft.com/en-us/library/bb356440(v=vs.110).aspx

// Array.IndexOf:

// https://msdn.microsoft.com/en-us/library/7eddebat(v=vs.110).aspx

HashSet<int> items = new HashSet<int>(v);

// Return the count of items in the second array which are present in our set

return r.Count(item => items.Contains(item));

}

}