Length of the largest subarray with contiguous elements | Set 2

Given an array of integers, find length of the longest subarray which contains numbers that can be arranged in a continuous sequence.

In the previous post, we have discussed a solution that assumes that elements in given array are distinct. Here we discuss a solution that works even if the input array has duplicates.

Examples:

```
Input:  arr[] = cc
Output: Length of the longest contiguous subarray is 4

Input:  arr[] = {10, 12, 12, 10, 10, 11, 10};
Output: Length of the longest contiguous subarray is 2
```

We strongly recommend to minimize the browser and try this yourself first.

The idea is similar to previous post. In the previous post, we checked whether maximum value minus minimum value is equal to ending index minus starting index or not. Since duplicate elements are allowed, we also need to check if the subarray contains duplicate elements or not. For example, the array {12, 14, 12} follows the first property, but numbers in it are not contiguous elements.
To check duplicate elements in a subarray, we create a hash set for every subarray and if we find an element already in hash, we don't consider the current subarray.

Following is Java implementation of the above idea.

```java
/* Java program to find length of the largest subarray which has
   all contiguous elements */
import java.util.*;

class Main
{
    // This function prints all distinct elements
    static int findLength(int arr[])
    {
        int n = arr.length;
        int max_len = 1; // Inialize result

        // One by one fix the starting points
        for (int i=0; i<n-1; i++)
        {
            // Create an empty hash set and add i'th element
            // to it.
            HashSet<Integer> set = new HashSet<>();
            set.add(arr[i]);

            // Initialize max and min in current subarray
            int mn = arr[i], mx = arr[i];

            // One by one fix ending points
            for (int j=i+1; j<n; j++)
            {
                // If current element is already in hash set, then
                // this subarray cannot contain contiguous elements
                if (set.contains(arr[j]))
                    break;

                // Else add curremt element to hash set and update
                // min, max if required.
                set.add(arr[j]);
                mn = Math.min(mn, arr[j]);
                mx = Math.max(mx, arr[j]);

                // We have already cheched for duplicates, now check
                // for other property and update max_len if needed
                if (mx-mn == j-i)
                    max_len = Math.max(max_len, mx-mn+1);
            }
        }
        return max_len; // Return result
    }

    // Driver method to test above method
    public static void main (String[] args)
    {
       int arr[] =  {10, 12, 12, 10, 10, 11, 10};
       System.out.println("Length of the longest contiguous subarray is " +
                           findLength(arr));
    }
}
```

Output:

```
Length of the longest contiguous subarray is 2
```

Time complexity of the above solution is O(n^2) under the assumption that hash set operations like add() and contains() work in O(1) time.