Count distinct elements in an array

Given an unsorted array, count all distinct elements in it.

Examples:

```
Input :   arr[] = {10, 20, 20, 10, 30, 10}
Output : 3
There are three distinct elements 10, 20 and 30.

Input :   arr[] = {10, 20, 20, 10, 20}
Output : 2
```

A **simple solution** is to run two loops. For every element, check if it has appeared before. If yes, increment count of distinct elements.

```
// C++ program to count distinct elements
// in a given array
#include <iostream>
using namespace std;

int countDistinct(int arr[], int n)
{
    int res = 1;

    // Pick all elements one by one
    for (int i = 1; i < n; i++) {
        int j = 0;
        for (j = 0; j < i; j++)
            if (arr[i] == arr[j])
                break;

        // If not printed earlier, then print it
```

```
        if (i == j)
            res++;
    }
    return res;
}

// Driver program to test above function
int main()
{
    int arr[] = { 12, 10, 9, 45, 2, 10, 10, 45 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << countDistinct(arr, n);
    return 0;
}
```

Output

```
5
```

Time Complexity of above solution is O(n2). We can **Use Sorting** to solve the problem in O(nLogn) time. The idea is simple, first sort the array so that all occurrences of every element become consecutive. Once the occurrences become consecutive, we can traverse the sorted array and count distinct elements in O(n) time. Following is the implementation of the idea.

```
// C++ program to count all distinct elements
// in a given array
#include <algorithm>
#include <iostream>
using namespace std;

int countDistinct(int arr[], int n)
{
    // First sort the array so that all
    // occurrences become consecutive
    sort(arr, arr + n);

    // Traverse the sorted array
    int res = 0;
```

```
    for (int i = 0; i < n; i++) {

        // Move the index ahead while
        // there are duplicates
        while (i < n - 1 && arr[i] == arr[i + 1])
            i++;

        res++;
    }

    return res;
}

// Driver program to test above function
int main()
{
    int arr[] = { 6, 10, 5, 4, 9, 120, 4, 6, 10 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << countDistinct(arr, n);
    return 0;
}
```

Output

```
6
```

We can **Use Hashing** to solve this in O(n) time on average. The idea is to traverse the given array from left to right and keep track of visited elements in a hash set , as a set consists of only unique elements.

Following is the implementation of the idea.

```
/* CPP program to print all distinct elements
of a given array */
#include <bits/stdc++.h>
using namespace std;

// This function prints all distinct elements
int countDistinct(int arr[], int n)
{
```

```
    // Creates an empty hashset
    unordered_set<int> s;

    // Traverse the input array
    int res = 0;
    for (int i = 0; i < n; i++) {

        // If not present, then put it in
        // hashtable and increment result
        if (s.find(arr[i]) == s.end()) {
            s.insert(arr[i]);
            res++;
        }
    }

    return res;
}

// Driver program to test above function
int main()
{
    int arr[] = { 6, 10, 5, 4, 9, 120, 4, 6, 10 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << countDistinct(arr, n);
    return 0;
}
```

Output

```
6
```

Set STL approach:

Sets are a type of associative containers in which each element has to be unique, because the value of the element identifies it. The value of the element cannot be modified once it is added to the set, though it is possible to remove and add the modified value of that element. We exploit the very basic property of set STL that it stores only unique numbers.

Examples:

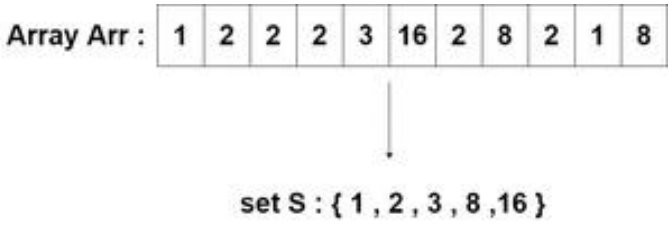
Arr = **[1,2,2,2,3,16,2,8,2,1,8]**

Distinct elements present = **[1,2,3,16,8]**

Total number of distinct elements in the array are **5**

Algorithm :

1. Insert all the elements into the set **S** one by one.



2. Store the total size **s** of the set using **set::size()**.

3.The total size **s** is the number of distinct elements present in the array.

```
#include <bits/stdc++.h>
using namespace std;
// function that accepts the array and it's size and returns
// the number of distince elements
int distinct(int* arr, int len)
{
    set<int> S; // declaring a set container using STL
    for (int i = 0; i < len; i++) {
        S.insert(arr[i]); // inserting all elements of the
                                            // array into set
        }
    int ans = S.size(); // calculating the size of the set
    return ans;
}
int main()
{
    int arr[] = { 12, 10, 9, 45, 2, 10, 10, 45};
    int l = sizeof(arr)/ sizeof(int); // calculating the size
    int dis_elements = distinct(arr, l); // calling the funct
    cout << dis_elements << endl;
```

```
    return 0;
}
```

Output

```
5
```