Count all distinct pairs with difference equal to k

Given an integer array and a positive integer k, count all distinct pairs with difference equal to k.

Examples:

```
Input: arr[] = {1, 5, 3, 4, 2}, k = 3
Output: 2
There are 2 pairs with difference 3, the pairs are {1, 4} and {5, 2}

Input: arr[] = {8, 12, 16, 4, 0, 20}, k = 4
Output: 5
There are 5 pairs with difference 4, the pairs are {0, 4}, {4, 8},
{8, 12}, {12, 16} and {16, 20}
```

Method 1 (Simple)
A simple solution is to consider all pairs one by one and check difference between every pair. Following program implements the simple solution. We run two loops: the outer loop picks the first element of pair, the inner loop looks for the other element. This solution doesn't work if there are duplicates in array as the requirement is to count only distinct pairs.

```
/* A simple program to count pairs with difference k*/
#include<iostream>
using namespace std;

int countPairsWithDiffK(int arr[], int n, int k)
{
    int count = 0;

    // Pick all elements one by one
    for (int i = 0; i < n; i++)
```

Google™ Custom Search

GeeksforGeeks

53,519 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
    {
        // See if there is a pair of this picked element
        for (int j = i+1; j < n; j++)
            if (arr[i] - arr[j] == k || arr[j] - arr[i] == k )
                  count++;
    }
    return count;
}

// Driver program to test above function
int main()
{
    int arr[] =  {1, 5, 3, 4, 2};
    int n = sizeof(arr)/sizeof(arr[0]);
    int k = 3;
    cout << "Count of pairs with given diff is "
         << countPairsWithDiffK(arr, n, k);
    return 0;
}
```

Output:

```
Count of pairs with given diff is 2
```

Time Complexity of O(n^2)

Method 2 (Use Sorting)
We can find the count in O(nLogn) time using a O(nLogn) sorting algorithm like Merge Sort, Heap Sort, etc. Following are the detailed steps.

```
1) Initialize count as 0
2) Sort all numbers in increasing order.
3) Remove duplicates from array.
4) Do following for each element arr[i]
   a) Binary Search for arr[i] + k in subarray from i+1 to n-1.
   b) If arr[i] + k found, increment count.
5) Return count.
```

```
/* A sorting based program to count pairs with difference k*/
#include <iostream>
#include <algorithm>
using namespace std;
```

Popular Posts
All permutations of a given string
Memory Layout of C Programs
Understanding "extern" keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
/* Standard binary search function */
int binarySearch(int arr[], int low, int high, int x)
{
    if (high >= low)
    {
        int mid = low + (high - low)/2;
        if (x == arr[mid])
            return mid;
        if (x > arr[mid])
            return binarySearch(arr, (mid + 1), high, x);
        else
            return binarySearch(arr, low, (mid -1), x);
    }
    return -1;
}

/* Returns count of pairs with difference k in arr[] of size n. */
int countPairsWithDiffK(int arr[], int n, int k)
{
    int count = 0, i;
    sort(arr, arr+n);  // Sort array elements

    /* code to remove duplicates from arr[] */

    // Pick a first element point
    for (i = 0; i < n-1; i++)
        if (binarySearch(arr, i+1, n-1, arr[i] + k) != -1)
            count++;

    return count;
}
```

Output:

```
Count of pairs with given diff is 2
```

Time complexity: The first step (sorting) takes O(nLogn) time. The second step runs binary search n times, so the time complexity of second step is also O(nLogn). Therefore, overall time complexity is O(nLogn). The second step can be optimized to O(n), see this.

Method 3 (Use Self-balancing BST)
We can also a self-balancing BST like AVL tree or Red Black tree to solve this problem. Following is detailed algorithm.

```
1) Initialize count as 0.
2) Insert all elements of arr[] in an AVL tree. While inserting,
   ignore an element if already present in AVL tree.
3) Do following for each element arr[i].
   a) Search for arr[i] + k in AVL tree, if found then increment count.
   b) Search for arr[i] - k in AVL tree, if found then increment count.
   c) Remove arr[i] from AVL tree.
```

Time complexity of above solution is also O(nLogn) as search and delete operations take O(Logn) time for a self-balancing binary search tree.

Method 4 (Use Hashing)
We can also use hashing to achieve the average time complexity as O(n) for many cases.

```
1) Initialize count as 0.
2) Insert all distinct elements of arr[] in a hash map.  While inserting,
   ignore an element if already present in the hash map.
3) Do following for each element arr[i].
   a) Look for arr[i] + k in the hash map, if found then increment count.
   b) Look for arr[i] - k in the hash map, if found then increment count.
   c) Remove arr[i] from hash table.
```

A very simple case where hashing works in O(n) time is the case where range of values is very small. For example, in the following implementation, range of numbers is assumed to be 0 to 99999. A simple hashing technique to use values as index can be used.

```
/* An efficient program to count pairs with difference k when the rang
   numbers is small */
#define MAX 100000
int countPairsWithDiffK(int arr[], int n, int k)
{
    int count = 0;  // Initialize count

    // Initialize empty hashmap.
    bool hashmap[MAX] = {false};

    // Insert array elements to hashmap
    for (int i = 0; i < n; i++)
        hashmap[arr[i]] = true;
```

Recent Comments

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 3 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 28 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 30 minutes ago

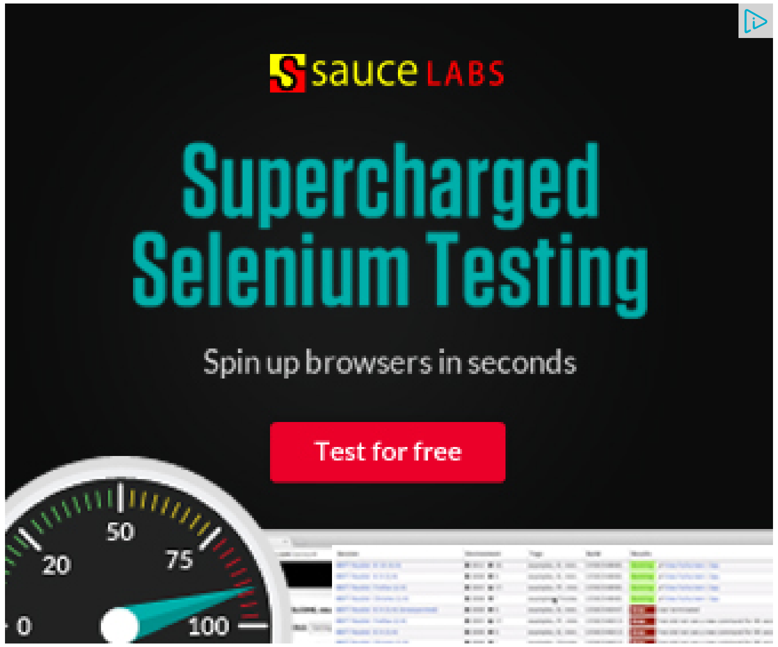
newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 54 minutes ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

Sanjay Agarwal You can also use the this method:...
Count trailing zeroes in factorial of a number · 1 hour ago

```
    for (int i = 0; i < n; i++)
    {
        int x = arr[i];
        if (x - k >= 0 && hashmap[x - k])
            count++;
        if (x + k < MAX && hashmap[x + k])
            count++;
        hashmap[x] = false;
    }
    return count;
}
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above



Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes

- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

11 Tweet 3 1

Writing code in comment? Please use **ideone.com** and share the link here.

24 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Debabrata • 21 days ago

1. sort the array
2. take 2 iterator say i and j when i =0 and j = i+1
3. repeat following steps from 4 to 7 until i<size ||="" j<size="" 4.="" if="" arr[i]=
{arr[i++]="" ,="" arr[j++]}="" 5.="" else="" if="" arr[i]="" -="" arr[j]="">= dist Then
6. else if arr[i] - arr[j] < dist Then j++

7 else i++ and j ++
8.End

Reply • Share ›

debabrata Debabrata • 21 days ago

I dont now where from those weird characters like ="" is coming and d
fix this .. otherwise it dosnt make any sense to post at here

Reply • Share ›

zzer • a month ago

in hashing method, we can optimize the process. for element arr[i], we put arr
and arr[i]+k has already been in hash table, if there exist in hash table we incre

By this way only one pass is needed.

Reply • Share ›

neelabhsingh • a month ago

@GeeksforGeeks , In method -4 (hash map)
2) Insert all distinct elements of arr[] in a hash map. While inserting,
ignore an element if already present in AVL tree. What is use of AVL tree in ha

Reply • Share ›

sethi → neelabhsingh • a month ago

It's a typo .
It should be "...if already present in Hashmap..."

Reply • Share ›

Kartik → sethi • a month ago

Thanks for pointing this out. We have corrected the typo.

Reply • Share ›

vijay kumar • 2 months ago

```
#include<stdio.h>

#include<conio.h>

int count[256];

int main()

{

int n=5,i,j=0,k=3;

int arr[10]={1,4,3,0,2,5,7,8,9,6};
```

{

count[arr[i]]++;

}

see more

Reply • Share ›

gkns • 3 months ago

in method 2, Can't we search for arr[i]+k and arr[i]-k and invalidate such an ele
n/2)?

Reply • Share ›

gourav pathak • 3 months ago

In Method 2 how can we reduce the complexity of 2nd step to O(N).....(In the li
single pair having difference k)

Reply • Share ›

rocker • 4 months ago

2nd Method can be made more efficient. There is a slight modification in 4th s

1) Initialize count as 0
2) Sort all numbers in increasing order.
3) Remove duplicates from array.
4) Do following for each element arr[i]
a) Binary Search for arr[i] + k in subarray from Low to n-1 (where Low = i+1 fo
b) If arr[i] + k found, increment count. Also update Low with the index of the ele
5) Return count.

1 • Reply • Share ›

gourav pathak rocker • 3 months ago

One more optimization could be done....if the number arr[i]+k is not fou
FLOOR(arr[i]+k)

^ | ˅ • Reply • Share ›

Deendayal • 4 months ago

GeeksforGeeks

2nd point of method-4 says

2) Insert all distinct elements of arr[] in a hash map. While inserting,
ignore an element if already present in AVL tree.

it should be

2) Insert all distinct elements of arr[] in a hash map. While inserting,
ignore an element if already present in Hash map.

thanks...

^ | ˅ • Reply • Share ›

Ish • 4 months ago

How about this solution(simple type) with a provision for ignoring duplicate pai

#include <stdio.h>

int check(int* arr,int n, int num){
int i = 0 ;
for(i;i<n;i++){ if(arr[i]="=num)" return="" 1;="" }="" return="" 0;="" }="" int="" dpa
int="" k=""){="" int="" count="0," checked[n]="" ,="" i="0" ,="" j,m="0" ;="" for(i=
if(check(checked,="" m,="" arr[i]="")="=0){" checked[m]="arr[i];" m++;="" for(j
if(abs(arr[i]-arr[j])="=k)" count++="" ;="" }="" }="" }="" return="" count="" ;="" }=
arr[]="{8," 12,="" 16,="" 4,="" 0,="" 20};="" int="" size="sizeof(arr)/sizeof(int);" p
return="" 0;="" }="">

Reply • Share ›

groomnestle • 4 months ago

The post is similar to http://www.geeksforgeeks.org/w... (Given an array A[] ar
sum as x), using hashtable (or map in STL) is a tidy solution.

Added some code to output the pairs:

```
int countPairsWithDiffK(int arr[], int n, int k)

{

int hashmap[20] = {0}; // assume number of elements are small



int count = 0;

int i;
```

see more

Reply • Share ›

Guest • 5 months ago

For hashing it your code wont work if the list of given integer numbers are not

#define MAX 100000

int countPairsWithDiffK(int arr[], int n, int k)

{

int count = 0; // Initialize count

// Initialize empty hashmap.

int hashmap[MAX] = {0};

// Insert array elements to hashmap

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

hashmap[arr[i]]++;

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

see more

• Reply • Share ›

smith Guest • 5 months ago

for (int i = 0; i < n; i++)
if(hashmap[arr[i]]==0)hashmap[arr[i]]++;
it will handle duplicate count alse.....

• Reply • Share ›

GeeksforGeeks Mod Guest • 5 months ago

Rezwan: Thanks for sharing your thoughts. The solution seems to be
unique pairs.

• Reply • Share ›

Pratik Sahoo GeeksforGeeks • 5 months ago

But the hashing technique would only be applicable if all elemen
sorting seems to a btr idea....and to remove duplicate after sort
technique...we can compare with the previous element and mo
time & O(1) space.

Kartik ➦ Pratik Sahoo • 5 months ago

As mentioned in the post, the given hashing technique w
0 to some max. For all other cases we can use a gener
programming languages.

• Reply • Share ›

Pratik Sahoo • 5 months ago

We hav to check if the pairs are distinct or not..so in sorting case.... after sorti
present more than once by comparing with its previous element ...if same we
just once.....

1 • Reply • Share ›

GeeksforGeeks Mod ➦ Pratik Sahoo • 5 months ago

Pratik, thanks for pointing this out. We have added a step to remove du
the array is sorted, we can remove duplicates in O(n) time. We have a
that it won't work for cases when array contains duplicates.

2 • Reply • Share ›

smith ➦ GeeksforGeeks • 5 months ago

In first method also,there is no criteria for handling duplicate pa
method to fix this??

1 • Reply • Share ›

Pratik Sahoo ➦ smith • 5 months ago

Yes we can do.... u can take a hash array and keep a c

int arr[100]; //original array

int count[100] = {0};//hash array

int i,size;

int count1=0;

for(i=0;i<size;i++) {="" if(count[arr[i]]="=0)" {="" arr[cour
count[arr[i]]="count[arr[i]]+1;" }="" count1="" will="" be="
duplicates;="" assuming="" all="" +ve="" elements="" in

^ | ⌄ • Reply • Share ›

smith ➦ Pratik Sahoo • 5 months ago

thanks pratik i got it.....

^ | ⌄ • Reply • Share ›

Subscribe   Add Disqus to your site