Count the number of possible triangles

Given an unsorted array of positive integers. Find the number of triangles that can be formed with three different array elements as three sides of triangles. For a triangle to be possible from 3 values, the sum of any two values (or sides) must be greater than the third value (or third side).
For example, if the input array is {4, 6, 3, 7}, the output should be 3. There are three triangles possible {3, 4, 6}, {4, 6, 7} and {3, 6, 7}. Note that {3, 4, 7} is not a possible triangle.
As another example, consider the array {10, 21, 22, 100, 101, 200, 300}. There can be 6 possible triangles: {10, 21, 22}, {21, 100, 101}, {22, 100, 101}, {10, 100, 101}, {100, 101, 200} and {101, 200, 300}

Method 1 (Brute force)
The brute force method is to run three loops and keep track of the number of triangles possible so far. The three loops select three different values from array, the innermost loop checks for the triangle property (the sum of any two sides must be greater than the value of third side).

Time Complexity: O(N^3) where N is the size of input array.

Method 2 (Tricky and Efficient)
Let a, b and c be three sides. The below condition must hold for a triangle (Sum of two sides is greater than the third side)
i) a + b > c
ii) b + c > a
iii) a + c > b

Following are steps to count triangle.

1. Sort the array in non-decreasing order.

2. Initialize two pointers ‘i’ and ‘j’ to first and second elements respectively, and initialize count of triangles as 0.

3. Fix ‘i’ and ‘j’ and find the rightmost index ‘k’ (or largest ‘arr[k]‘) such that ‘arr[i] + arr[j] > arr[k]‘. The number of triangles that can be formed with ‘arr[i]‘ and ‘arr[j]‘ as two sides is ‘k – j’. Add ‘k – j’ to count of triangles.

Let us consider ‘arr[i]‘ as ‘a’, ‘arr[j]‘ as b and all elements between ‘arr[j+1]‘ and ‘arr[k]‘ as ‘c’. The above mentioned conditions (ii) and (iii) are satisfied because ‘arr[i] < arr[j] < arr[k]'. And we check for condition (i) when we pick 'k'

4. Increment ‘j’ to fix the second element again.

Note that in step 3, we can use the previous value of ‘k’. The reason is simple, if we know that the value of ‘arr[i] + arr[j-1]‘ is greater than ‘arr[k]‘, then we can say ‘arr[i] + arr[j]‘ will also be greater than ‘arr[k]‘, because the array is sorted in increasing order.

5. If ‘j’ has reached end, then increment ‘i’. Initialize ‘j’ as ‘i + 1′, ‘k’ as ‘i+2′ and repeat the steps 3 and 4.

Following is implementation of the above approach.

```
// Program to count number of triangles that can be formed from given
#include <stdio.h>
#include <stdlib.h>

/* Following function is needed for library function qsort(). Refer
   http://www.cplusplus.com/reference/clibrary/cstdlib/qsort/ */
int comp(const void* a, const void* b)
{  return *(int*)a > *(int*)b ; }

// Function to count all possible triangles with arr[] elements
int findNumberOfTriangles(int arr[], int n)
{
    // Sort the array elements in non-decreasing order
    qsort(arr, n, sizeof( arr[0] ), comp);

    // Initialize count of triangles
    int count = 0;

    // Fix the first element.  We need to run till n-3 as the other tw
    // selected from arr[i+1...n-1]
    for (int i = 0; i < n-2; ++i)
```


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

```
    {
        // Initialize index of the rightmost third element
        int k = i+2;

        // Fix the second element
        for (int j = i+1; j < n; ++j)
        {
            // Find the rightmost element which is smaller than the su
            // of two fixed elements
            // The important thing to note here is, we use the previou
            // value of k. If value of arr[i] + arr[j-1] was greater t
            // then arr[i] + arr[j] must be greater than k, because th
            // array is sorted.
            while (k < n && arr[i] + arr[j] > arr[k])
               ++k;

            // Total number of possible triangles that can be formed
            // with the two fixed elements is k - j - 1.  The two fixe
            // elements are arr[i] and arr[j].  All elements between a
            // to arr[k-1] can form a triangle with arr[i] and arr[j].
            // One is subtracted from k because k is incremented one e
            // in above while loop.
            // k will always be greater than j. If j becomes equal to
            // above loop will increment k, because arr[k] + arr[i] is
            // greater than arr[k]
            count += k - j - 1;
        }
    }

    return count;
}

// Driver program to test above functionarr[j+1]
int main()
{
    int arr[] =   {10, 21, 22, 100, 101, 200, 300};
    int size = sizeof( arr ) / sizeof( arr[0] );

    printf("Total number of triangles possible is %d ",
           findNumberOfTriangles( arr, size ) );

    return 0;
}
```

Output:

```
Total number of triangles possible is 6
```

Time Complexity: O(n^2). The time complexity looks more because of 3 nested loops. If we take a closer look at the algorithm, we observe that k is initialized only once in the outermost loop. The innermost loop executes at most O(n) time for every iteration of outer most loop, because k starts from i+2 and goes upto n for all values of j. Therefore, the time complexity is O(n^2).

Source: http://stackoverflow.com/questions/8110538/total-number-of-possible-triangles-from-n-numbers

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

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 9 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 12 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 37 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 39 minutes ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

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

- 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

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

42 Comments GeeksforGeeks

Sort by Newest

Gautam Jha • 6 months ago

@geeksforgeeks
can u explain the complexity for this input arr[]= {14,15,16,17,18,19,20,21,22,23,24,25,26,27,28};
i think it will b n^3.....science each time u initialize the k as 2 next of i .why no and then do some calculation regarding previous two index of i and j and incre will b happen in o(n^2) complexity (if i m wrong then plz let me know..)

AlienOnEarth Gautam Jha • 13 days ago

Yeah. Even I think its o(n^3).

Sri AlienOnEarth • 7 days ago

its o(n^2). since n* (O(n)+O(n)) .

AlienOnEarth Sri • 6 days ago

Got it. Thanks for the explanation

Reply • Share ›

denial • 10 months ago

@geeksforgeeks
Please correct the text (not code) written in algorithm above. In step 4, it is wri
the previous value of ‘k’" , but I think we are using k in step 3. So correct this lir

Reply • Share ›

GeeksforGeeks Mod denial • 13 days ago

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

Reply • Share ›

aspire • 10 months ago

@GeeksforGeeks : I would like to suggest an O(n^2*logn) algorithm.
This algorithm runs two loops and then a binary search to find the first such el
sum of the 2 loop elements.

```
 #include<stdio.h>
#include<stdlib.h>

int comp(const void *a,const void *b)
{
    return (*(int*)a-*(int*)b);
}

int binaryLastLesser(int a[],int l,int h,int sum)
{
    int m = (l+h)/2;
    if(l>h)
        return -1;
    else
```

see more

3 ^ | v • Reply • Share ›

wgpshashank ➚ aspire • 3 months ago

This is not correct , you need to find two such indexes e.g. 1st and last between them.

^ | v • Reply • Share ›

pefullarton • 11 months ago

Shouldn't the middle loop for j
....
// Fix the second element
for (int j = i+1; j < n; ++j) <<<<<<<<<<<<
{
....

should be
for (int j = i+1; j < k; ++j)
of course we need to initialize k before entering the loop.

Am i wrong????

^ | v • Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

^ | v • Reply • Share ›

Prateek Sharma • a year ago

Python Code.....

```
def numberOfTriangles(a):
    arraylist =[]
```

```
    count =0
    for i in range(len(a)-2):
        for j in range(i+2,len(a)):
            arraylist.append([a[i],a[j-1],a[j]])
    for i in arraylist:
        if i[0]+i[1]>i[2] and i[0]+i[2]>i[1] and i[1]+i[2]>i[0]:
            count = count+1
    print "number of triangles possible:"+str(count)
def main():
    array =[10,21,22,100,101,200,300]
    numberOfTriangles(array)
if __name__ == '__main__':
    main()
```

 

anonymous Prateek Sharma • 7 months ago

Try your code with array = [10,21,22,23]....
It doesn't work....
because you aren't considering (10,21,23) in your arraylist...

Silva • a year ago

I think an easier way of seeing this problem is generating all subsets of the arr
if they form a triangle.

It's easy to do this with recursion.

[sourcecode language="C#"]
private static int DoCountTriangles(int[] array, int arrayIndex, int[] sides, int side
{
//cannot form a triangle (not enouth elements left)

```
if (array.Length - arrayIndex < 3 - sideIndex)
{
return 0;
}

//we have a set of three elemtsn: a triangle can possibly be formed
if (sideIndex == 3)
{
//return 1 is it's a triagle (and break the recursion anyway)
if (IsTriangle(sides[0], sides[1], sides[2]))
```

see more

• Reply • Share ›

Rik Dey • a year ago

Minhazul, the algo seems to be correct... it needs no more comparison in your last value and without doing any check/comparison for the array elements it di the correct result

• Reply • Share ›

Jaykrishna Babu • a year ago

i guess it is right man!! check the 4th step in the algo...and the for loop for j run the possible combinations of i, i+1 with the other sides, the next step j is increi other possible values...so it must be able to find the combinations which u hav

• Reply • Share ›

Sk Minhazul Islam • a year ago

This is wrong..

suppose we have an array..

{40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50}.

Then this algo won't find the combination of {40, 42, 43}, {40, 42, 44}, {40, 42, 48}, {40, 42, 49} and {40, 42, 50}.

because the the variable "k" already has been incremented.

I don&#039t thing worst case scenario can optimized to O(n^2),

Because worst case scenario is when we have consecutive numbers just like

Reply • Share ›

rahul sundar • a year ago

The triangle count k-j-1 after finding k is wrong. It should be n-k-1, as all the ele

```
/* Paste your code here (You may delete these lines if not writing co
```

Reply • Share ›

Kartik ➚ rahul sundar • a year ago

@Rahul: Could you please provide a sample input for which the given

Reply • Share ›

Balasubramanian • a year ago

In the problem statement, it is given that {3,4,7} cannot form a triangle. But if w for forming a triangle is satisfied(7+4>3). So, won't it also be one of the solutio

Reply • Share ›

pefullarton ➚ Balasubramanian • 11 months ago

If you look closely at the algorithm, we are sorting the input array first. S become {3,4,7}. Get it????

```
/* Paste your code here (You may delete these lines if not wri
```

Reply • Share ›

LK • 2 years ago

Let a, b and c be three sides. The below condition must hold for a triangle (Su
side)
i) a + b > c
ii) b + c > a
iii) a + c > b

INSTEAD

For a triangle "sum of two smaller sides must be greater than the third one".

Eg: 4 3 1 is not a triangle because 1 + 3 (smaller sides) = 4 (not greater)

  • Reply • Share ›

wgpshashank • 2 years ago

more simpler :D

```
import java.util.*;

class Solution
{
public static void main(String ar[])
{
int a[]={10, 21, 22, 100, 101, 200, 300};
Arrays.sort(a);
int count=0;
for (int i = 0; i < a.length-2; i++)
{
for(int j=i+1;ja[j+1])
count++;
```

```
}
System.out.println(count);
}

}
```

• Reply • Share ›

atul007 • 2 years ago

outer loop should run till n-2 (i loop) , to avoid unnecessary iteration.

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

kartik ↗ atul007 • 2 years ago

@atul007: Thanks for suggesting the optimization. We have changed t

• Reply • Share ›

Nick ↗ kartik • 10 months ago

Dont u think j should be from i+1 to n-2 ?

• Reply • Share ›

somesh • 2 years ago

I could not understand how is it O(n^2)
you are iterating i from 1:n
j from 1:n
and k from i+2:n
then how is it o(n^2) not n^3??
please explain in detail

```
/* Paste your code here (You may delete these lines if not writing c
```

HLS.Nirma • 2 years ago

Very good code.

Thank you very much.

• Reply • Share ›

rup • 2 years ago

[sourcecode language="C++"]
/* #include<iostream>
using namespace std;
#include<algorithm>

int noOfTriangle(int arr[],int length)
{
int j=length-2;//initializing 2nd last element
static int count=0;
int i=j-1;//last third element
// cout<<length;
while(i>=0)
{
/* as the array is sorted obviously sum of j&j+1 will be greater then i so we just
array which are greater then the difference of j+1 & j */
if((arr[j+1]-arr[j])<arr[i])
{
count++;

see more

• Reply • Share ›

rup • 2 years ago

all possible triangle in {10, 21, 22, 100, 101, 200, 300}as u said are {10, 21, 22
100, 101}, {100, 101, 200} and {101, 200, 300}
I think these should also be the possible triangle
{10,22,100},{10,21,100},{10,100,200},{10,100,300}....
plz tell me if i'm wrong

Reply • Share ›

amor ➚ rup • 2 years ago

No,it is not because we have to check all the three condition of triangle
{10,22,100},{10,21,100} and other set are not satified those properties.

Reply • Share ›

Kartik ➚ rup • 2 years ago

{10,22,100},{10,21,100},{10,100,200},{10,100,300} are not possible, be

Reply • Share ›

shashank • 2 years ago

Okay instead of using a linear search to find k why cant we use a binary searc
helpful if the array is long.Although it will not change the complexity as the choi
the complexity O(n^2)
but still an optimization in terms of total execution time.

```
#include <iostream>
using namespace std;
int binary_search(int arr[],int start,int end,int find)
{
int mid=(start+end)/2;
if(start==end&&arr[mid]!=find)return start;
else if(arr[mid]==find)return mid;
```

```
else if(arr[mid]>find)return binary_search(arr,start,mid,find);
else if(arr[mid]<find)return binary_search(arr,mid+1,end,find);
}
```

see more

• Reply • Share ›

Shashank Kumar • 2 years ago

I think I have found a more optimal solution(dont know whether it will change th
the given algorithm u used a linear search to find out the third index(k ,here it is
array is sorted so why not use binary search,it might be handy when the array

Here is the code.

```
#include <iostream>
using namespace std;
int binary_search(int arr[],int start,int end,int find)
{
int mid=(start+end)/2;
if(start==end&&arr[mid]!=find)return start;
else if(arr[mid]==find)return mid;
else if(arr[mid]>find)return binary_search(arr,start,mid,find);
else if(arr[mid]<find)return binary_search(arr,mid+1,end,find);
}
```

see more

• Reply • Share ›

amor • 2 years ago

```
#include
#include
using namespace std;
int Number_Of_Triangle(int arr[],int len)
{
int pos,count=0;
sort(&arr[0],&arr[len-1]);
for(int i=0;i0 && arr[i+1]>0)
{
pos=i;
if(arr[i]+arr[i+1]>arr[i+2])
{
count++;
cout<<"("<<arr[i]<<"&#039"<<arr[i+1]<<"&#039"<<arr[i+2]<<")" <=0)
{
//check for -ve numbers
if(arr[pos]>0 && arr[i+1]>0)
{
```

see more

^ | v • Reply • Share ›

amor • 2 years ago

The one way to count the number of triangles is:
1) Sort the array in O(nlogn).
2)i) Now check arr[i]+arr[i+1]>arr[i+2] then count++ i.e. it is the triangle
ii) now for middle element situation like {10,20,30,100,101}
100+30>101,100+20>101 ,100+10>101
so we take one variable pos to track the previous elements like 20,10 etc,
iii)count the number of triangle each time the triangle property satisified.

```
#include
using namespace std;
int Number_Of_Triangle(int arr[],int len)
{
int pos,count=0;
sort(&arr[0],&arr[len-1]);
for(int i=0;i0 && arr[i+1]>0)
{
pos=i;
```

see more

Reply • Share ›

mukesh • 2 years ago

can u tell me all possible triangle in {10, 21, 22, 100, 101, 200, 300};..

Reply • Share ›

Kartik ➚ mukesh • 2 years ago

{10, 21, 22}, {21, 100, 101}, {22, 100, 101}, {100, 101, 200} and {101, 2

Reply • Share ›

Kartik ➚ Kartik • 2 years ago

Apologies, I missed the 6th triangle {10, 100, 101}. Have added

Reply • Share ›

anonym • 2 years ago

There is a typo in the statement - "For a triangle to be possible from 3 values,
must be smaller(SHOULD BE GREATER) than the third value (or third side)."

```
/* Paste your code here (You may delete these lines if not writing co
```

-

Reply • Share ›

GeeksforGeeks anonym • 2 years ago

Thanks for pointing this out. We have corrected the typo. Keep it up!

Reply • Share ›

Jenish • 2 years ago

In above example, isnt {21,100,101} also valid triangle?

```
/* Paste your code here (You may delete these lines if not writing co
```

Reply • Share ›

GeeksforGeeks Jenish • 2 years ago

Thanks for pointing this out. Instead of {21, 100, 101}, there was {21, 2
it now. So total number of triangles is still 6. Keep it up.

Reply • Share ›

Subscribe Add Disqus to your site