GeeksforGeeks
A computer science portal for geeks
Practice IDE Q&A GeeksQuiz

Pythagorean Triplet in an array

Given an array of integers, write a function that returns true if there is a triplet (a, b, c) that satisfies $a^2 + b^2 = c^2$.

Example:

```
Input: arr[] = {3, 1, 4, 6, 5}
Output: True
There is a Pythagorean triplet (3, 4, 5).

Input: arr[] = {10, 4, 6, 12, 5}
Output: False
There is no Pythagorean triplet.
```

Method 1 (Naive)

A simple solution is to run three loops, three loops pick three array elements and check if current three elements form a Pythagorean Triplet.

Below is C++ implementation of simple solution.

C++

```
// A C++ program that returns true if there is a Pythagorean
// Triplet in a given aray.
#include <iostream>
using namespace std;

// Returns true if there is Pythagorean triplet in ar[0..n-1]
bool isTriplet(int ar[], int n)
{
    for (int i=0; i<n; i++)
    {
       for (int j=i+1; j<n; j++)
       {
          for (int k=j+1; k<n; k++)
          {
            // Calculate square of array elements
            int x = ar[i]*ar[i], y = ar[j]*ar[j], z = ar[k]*ar[k];

            if (x == y + z || y == x + z || z == x + y)
```

```
                 return true;
          }
       }
    }

    // If we reach here, no triplet found
    return false;
}

/* Driver program to test above function */
int main()
{
    int ar[] = {3, 1, 4, 6, 5};
    int ar_size = sizeof(ar)/sizeof(ar[0]);
    isTriplet(ar, ar_size)? cout << "Yes": cout << "No";
    return 0;
}
```

Run on IDE

Java

```
// A Java program that returns true if there is a Pythagorean
// Triplet in a given aray.
import java.io.*;

class PythagoreanTriplet {

    // Returns true if there is Pythagorean triplet in ar[0..n-1]
    static boolean isTriplet(int ar[], int n)
    {
        for (int i=0; i<n; i++)
        {
            for (int j=i+1; j<n; j++)
            {
                for (int k=j+1; k<n; k++)
                {
                    // Calculate square of array elements
                    int x = ar[i]*ar[i], y = ar[j]*ar[j], z = ar[k]*ar[k];

                    if (x == y + z || y == x + z || z == x + y)
                        return true;
                }
            }
        }

        // If we reach here, no triplet found
        return false;
    }


    // Driver program to test above function
    public static void main(String[] args)
    {
        int ar[] = {3, 1, 4, 6, 5};
        int ar_size = ar.length;
        if(isTriplet(ar,ar_size)==true)
           System.out.println("Yes");
        else
           System.out.println("No");
    }
}
```

```
/* This code is contributed by Devesh Agrawal */
```

Run on IDE

Output:

```
Yes
```

Time Complexity of the above solution is O(n^3).

Method 2 (Use Sorting)

We can solve this in O(n^2) time by sorting the array first.

1) Do square of every element in input array. This step takes O(n) time.

2) Sort the squared array in increasing order. This step takes O(nLogn) time.

3) To find a triplet (a, b, c) such that a = b + c, do following.

1. Fix ‘a’ as last element of sorted array.
2. Now search for pair (b, c) in subarray between first element and ‘a’. A pair (b, c) with given sum can be found in O(n) time using meet in middle algorithm discussed in method 1 of this post.
3. If no pair found for current ‘a’, then move ‘a’ one position back and repeat step 3.b.

Below is C++ implementation of above algorithm.

C++

```
// A C++ program that returns true if there is a Pythagorean
// Triplet in a given array.
#include <iostream>
#include <algorithm>
using namespace std;

// Returns true if there is a triplet with following property
// A[i]*A[i] = A[j]*A[j] + A[k]*[k]
// Note that this function modifies given array
bool isTriplet(int arr[], int n)
{
    // Square array elements
    for (int i=0; i<n; i++)
        arr[i] = arr[i]*arr[i];

    // Sort array elements
    sort(arr, arr + n);

    // Now fix one element one by one and find the other two
    // elements
    for (int i = n-1; i >= 2; i--)
    {
        // To find the other two elements, start two index
        // variables from two corners of the array and move
        // them toward each other
        int l = 0; // index of the first element in arr[0..i-1]
        int r = i-1; // index of the last element in arr[0..i-1]
        while (l < r)
        {
            // A triplet found
            if (arr[l] + arr[r] == arr[i])
                return true;

            // Else either move 'l' or 'r'
            (arr[l] + arr[r] < arr[i])?  l++: r--;
        }
    }

    // If we reach here, then no triplet found
    return false;
}

/* Driver program to test above function */
int main()
{
    int arr[] = {3, 1, 4, 6, 5};
    int arr_size = sizeof(arr)/sizeof(arr[0]);
    isTriplet(arr, arr_size)? cout << "Yes": cout << "No";
    return 0;
}
```

Run on IDE

Java

```
// A Java program that returns true if there is a Pythagorean
// Triplet in a given aray.
import java.io.*;
import java.util.*;

class PythagoreanTriplet
{
    // Returns true if there is a triplet with following property
    // A[i]*A[i] = A[j]*A[j] + A[k]*[k]
    // Note that this function modifies given array
    static boolean isTriplet(int arr[], int n)
    {
        // Square array elements
        for (int i=0; i<n; i++)
            arr[i] = arr[i]*arr[i];

        // Sort array elements
        Arrays.sort(arr);

        // Now fix one element one by one and find the other two
        // elements
        for (int i = n-1; i >= 2; i--)
        {
            // To find the other two elements, start two index
            // variables from two corners of the array and move
            // them toward each other
            int l = 0; // index of the first element in arr[0..i-1]
            int r = i-1; // index of the last element in arr[0..i-1]
            while (l < r)
            {
                // A triplet found
                if (arr[l] + arr[r] == arr[i])
                    return true;

                // Else either move 'l' or 'r'
                if (arr[l] + arr[r] < arr[i])
                   l++;
                else
                   r--;
            }
        }

        // If we reach here, then no triplet found
        return false;
    }


    // Driver program to test above function
    public static void main(String[] args)
    {
        int arr[] = {3, 1, 4, 6, 5};
        int arr_size = arr.length;
        if (isTriplet(arr,arr_size)==true)
           System.out.println("Yes");
        else
           System.out.println("No");
    }
}
/*This code is contributed by Devesh Agrawal*/
```

Run on IDE

Output:

```
Yes
```

Time complexity of this method is O(n^2).

This article is contributed by **Harshit Gupta**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

35 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

3.1 Average Difficulty : **3.1/5.0**
Based on **8** vote(s)

□ Add to TODO List

□ Mark as DONE

Like Share 17 people like this. Be the first of your friends.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.