**Pythagorean Triplet**

*"Geometry is knowledge of the eternally existent."*

**Question:**

1. You are required to check if a given set of numbers is a valid Pythagorean triplet.
2. Take as input three numbers a, b and c.
3. Print true if they can form a Pythagorean triplet and false otherwise.

**Input format:**

a, an integer

b, an integer

c, an integer

**Output format:**

true if the numbers form a Pythagorean triplet and false otherwise.

**Constraints:**

1 <= a <= 10^9

1 <= b <= 10^9

1 <= c <= 10^9

A **Pythagorean triplet** consists of three positive integers a, b, and c, such that a2 + b2 = c2.A triangle whose sides form a Pythagorean triple is called a **Pythagorean triangle**, and is necessarily a right triangle.

**Solution Approach:**

There is a fairly straight-forward problem. We have to deduce whether the given numbers can form a right-angled triangle or not (Pythagorean triplet).

We find the greatest number out of the three numbers.

We sum the squared values of the smaller values and try equating it with the squared value of the greatest value.

If this equation is valid, then the given numbers are a Pythagorean triplet and we print "true", Else we print "false".

**Programming Implementation (In Java)**

import java.util.\*;

public class Main {

    public static void main(String[] args) {

        Scanner scn = new Scanner(System.in);

        int a = scn.nextInt();

        int b = scn.nextInt();

        int c = scn.nextInt();

        int max = a;

        if(b >= max)

            max = b;

        if(c >= max)

            max = c;

        if(max == a){

            System.out.println((b \* b + c \* c) == (a \* a));

        } else if(max == b){

            System.out.println((a \* a + c \* c) == (b \* b));

        } else {

            System.out.println((a \* a + b \* b) == (c \* c));

        }

    }

}

**Space & Time Complexity Analysis**

* What is the **time complexity** of the above solution ?

Finding maximum of 3 numbers, finding whether a2 + b2 = c2 or not, all are constant work, hence time complexity will be O(1).

* What is the **space complexity** of above solution ?

We are not using any data structure, hence it is taking constant space. Thus space complexity will be O(1).

**Asked in Companies:** Amazon