Daily Coding Problem #282

Problem

This problem was asked by Netflix.

Given an array of integers, determine whether it contains a Pythagorean triplet. Recall that a Pythogorean triplet `(a, b, c)` is defined by the equation $a^2 + b^2 = c^2$.

Solution

One simple solution involves looping through the array three times, trying each possible combination of three numbers to see if the Pythogorean property holds true. We can make this a little faster by precomputing the squares of each number.

```
def triplet(array):
    array = [x ** 2 for x in array]
    for a in array:
        for b in array:
            for c in array:
                if a + b == c or a + c == b or b + c == a:
                    return True

    return False
```

However, with this algorithm we cannot get around the fact that using three loops corresponds to $O(N^3)$ time complexity, where N is the total number of integers.

A better way to go about this is to first sort the squared array. Without loss of generality, we can assume that $a < b < c$, and so the elements of our triplet appear in that order in our new array. Given this information, we can apply the following algorithm.

First, let us assume that c is the last element in the array. Then the lowest possible value of a will be the first element, and the highest possible value of b will be the second-to-last element. Now we repeatedly compare $a + b$ against c, and perform the following:

- If $a + b < c$, move the index of a up in the list, to make our squared total higher.
- If $a + b > c$, move the index of b down in the list, to make our squared total lower.
- If $a + b = c$, return True, as we have found a solution.

Once a and b cross paths, we know there cannot be any more solutions with our current value of c, so we decrement c and try again. If we check all values of c and fail to find a solution, there cannot be a triplet.

```
def triplet(array):
    array = sorted([x ** 2 for x in array])

    for c in range(len(array) - 1, 1, -1):
        a, b = 0, c - 1

        while a < b:
            if array[a] + array[b] == array[c]:
                return True
            elif array[a] + array[b] < array[c]:
                a += 1
            else:
                b -= 1

    return False
```

With this algorithm, we only need to iterate through the array once for each value of c, so

the runtime will be $O(N^2)$.