Line Reflection

Given n points on a 2D plane, find if there is such a line parallel to y-axis that reflect the given set of points.

Example 1:

Given *points* = `[[1,1],[-1,1]]`, return `true`.

Example 2:

Given *points* = `[[1,1],[-1,-1]]`, return `false`.

Credits:
Special thanks to @memoryless for adding this problem and creating all test cases.

Solution 1

Idea: Reflect the points by replacing every x with minX+maxX-x and then check whether you get the same points. Why minX+maxX-x? I actually thought of it as minX+(maxX-x), i.e., first the subtraction (maxX-x). That's how far x is away from the max, so instead go that distance from the min.

Ruby

```ruby
def is_reflected(points)
  points.sort! == points.map { |x, y| [points[0][0] + points[-1][0] - x, y] }.sor
t
end
```

Python

```python
def isReflected(self, points):
    points.sort()
    return points == sorted([points[0][0] + points[-1][0] - x, y]
                            for x, y in points)
```

A linear time one:

```python
def isReflected(self, points):
    if not points: return True
    X = min(points)[0] + max(points)[0]
    return {(x, y) for x, y in points} == {(X - x, y) for x, y in points}
```

Shorter, but I think less nice:

```python
    return set(map(tuple, points)) == {(X - x, y) for x, y in points}
```

written by StefanPochmann original link here

Solution 2

The idea is quite simple. If there exists a line reflecting the points, then each pair of symmetric points will have their x coordinates adding up to the same value, including the pair with the maximum and minimum x coordinates. So, in the first pass, I iterate through the array, adding each point to the hash set, and keeping record of the minimum and maximum x coordinates. Then, in the second pass, I check for every point to the left of the reflecting line, if its symmetric point is in the point set or not. If all points pass the test, then there exists a reflecting line. Otherwise, not.

By the way, here, to hash the content of an array, rather than the reference value, I use **Arrays.hashCode(int[])** first, and then re-hash this hash code. You can also use **Arrays.toString(int[])** to first convey the 2d array to a string, and then hash the string. But the second method is slower.

```
public class Solution {
    public boolean isReflected(int[][] points) {
        HashSet<Integer> pointSet = new HashSet<>();
        int sum;
        int maxX, minX;

        minX = Integer.MAX_VALUE;
        maxX = Integer.MIN_VALUE;
        for(int[] point:points) {
            maxX = Math.max(maxX, point[ 0 ]);
            minX = Math.min(minX, point[ 0 ]);
            pointSet.add(Arrays.hashCode(point));
        }

        sum = maxX+minX;
        for(int[] point:points) {
            if(!pointSet.contains(Arrays.hashCode(new int[]{sum-point[ 0 ], point
[ 1 ]}))) {
                return false;
            }
        }
        return true;
    }
}
```

written by Fanchao original link here

Solution 3

```
   public boolean isReflected(int[][] points) {
    int max = Integer.MIN_VALUE;
    int min = Integer.MAX_VALUE;
    HashSet<String> set = new HashSet<>();
    for(int[] p:points){
        max = Math.max(max,p[0]);
        min = Math.min(min,p[0]);
        String str = p[0] + "a" + p[1];
        set.add(str);
    }
    int sum = max+min;
    for(int[] p:points){
        //int[] arr = {sum-p[0],p[1]};
        String str = (sum-p[0]) + "a" + p[1];
        if( !set.contains(str))
            return false;

    }
    return true;
}
```

written by juanren original link here