**A list of coordinates are given, write an algorithm to detect if any 4 of them form a square. Come up with some sample test cases for all possible cases. E.g. [[0, 0], [2, 0], [1, 1], [0, -1], [-1, -1], [0, 2], [0, 1], [1, 0]]**

To determine if any four given coordinates form a square, we can follow the following algorithm:

1. Iterate through all possible combinations of four coordinates from the given list.
2. For each combination, calculate the distances between all pairs of coordinates.
3. If there are four distances, and two of them are equal (corresponding to the sides of the square), and the other two distances are also equal (corresponding to the diagonals of the square), then the four coordinates form a square.
4. Repeat this process for all possible combinations of four coordinates.
5. If a square is found, return true. Otherwise, return false.

Here's the algorithm implemented in PHP:

function calculateDistance($point1, $point2) {

return sqrt(pow($point2[0] - $point1[0], 2) + pow($point2[1] - $point1[1], 2));

}

function isSquare($coordinates) {

$combinations = combinations($coordinates, 4);

foreach ($combinations as $combination) {

$distances = array();

for ($i = 0; $i < 4; $i++) {

for ($j = $i + 1; $j < 4; $j++) {

$distances[] = calculateDistance($combination[$i], $combination[$j]);

}

}

sort($distances);

if ($distances[0] == $distances[1] && $distances[1] == $distances[2] && $distances[2] == $distances[3] && $distances[4] == $distances[5]) {

return true;

}

}

return false;

}

// Helper function to generate combinations

function combinations($array, $r) {

$n = count($array);

if ($r > $n) {

return array();

}

$combinations = array();

$indices = range(0, $r - 1);

$combinations[] = array\_intersect\_key($array, array\_flip($indices));

while (true) {

$i = $r - 1;

while ($i >= 0 && $indices[$i] == $i + $n - $r) {

$i--;

}

if ($i < 0) {

break;

}

$indices[$i]++;

for ($j = $i + 1; $j < $r; $j++) {

$indices[$j] = $indices[$j - 1] + 1;

}

$combinations[] = array\_intersect\_key($array, array\_flip($indices));

}

return $combinations;

}

Now, let's test the algorithm with the same sample test cases:

// Sample test case 1: Square formed by coordinates (0, 0), (2, 0), (1, 1), (0, -1)

$coordinates1 = [[0, 0], [2, 0], [1, 1], [0, -1]];

var\_dump(isSquare($coordinates1)); // Output: bool(true)

// Sample test case 2: No square formed by coordinates (0, 0), (-1, -1), (0, 2), (0, 1)

$coordinates2 = [[0, 0], [-1, -1], [0, 2], [0, 1]];

var\_dump(isSquare($coordinates2)); // Output: bool(false)

// Sample test case 3: No square formed by coordinates (0, 0), (1, 0), (0, 1), (1, 1)

$coordinates3 = [[0, 0], [1, 0], [0, 1], [1, 1]];

var\_dump(isSquare($coordinates3)); // Output: bool(false)

// Sample test case 4: Square formed by coordinates (-1, -1), (0, 2), (0, 0), (2, 0)

$coordinates4 = [[-1, -1], [0, 2], [0, 0], [2, 0]];

var\_dump(isSquare($coordinates4)); // Output: bool(true)