593. Valid Square

Description HintsSubmissionsSolutions

- Total Accepted: **3135**
- Total Submissions: **8229**
- Difficulty: **Medium**
- Contributors:fallcreek

Given the coordinates of four points in 2D space, return whether the four points could construct a square.

The coordinate (x,y) of a point is represented by an integer array with two integers.

Example:

Input: `p1 = [0,0], p2 = [1,1], p3 = [1,0], p4 = [0,1]` **Output:** `True`

Note:

1. All the input integers are in the range [-10000, 10000].
2. A valid square has four equal sides with positive length and four equal angles (90-degree angles).
3. Input points have no order.

Subscribe to see which companies asked this question.

C++ 3 lines (unordered_set)
If we calculate all distances between 4 points, 4 smaller distances should be equal (sides), and 2 larger distances should be equal too (diagonals). As an optimization, we can compare squares of the distances, so we do not have to deal with the square root and precision loss.

Therefore, our set will only contain 2 unique distances in case of square (beware of the zero distance though).

```
int d(vector<int>& p1, vector<int>& p2) {
    return (p1[0] - p2[0]) * (p1[0] - p2[0]) + (p1[1] - p2[1]) * (p1[1] - p2[1]);
}
bool validSquare(vector<int>& p1, vector<int>& p2, vector<int>& p3, vector<int>& p4) {
    unordered_set<int> s({ d(p1, p2), d(p1, p3), d(p1, p4), d(p2, p3), d(p2, p4), d(p3, p4) });
    return !s.count(0) && s.size() == 2;
}
```