Question: https://leetcode.com/problems/valid-square/

So the logic is very simple, in square we will have 4 equal sides, and 2 equal diagonals.

So if we store all of that in a set, we will get 2 unique values only.

So thats what we did, we stored square of each side in a set and at the end if we have 2 unique values and none of the values are 0 then we have a valid square.

Code:  
class Solution {

public boolean validSquare(int[] p1, int[] p2, int[] p3, int[] p4) {

Set<Integer> uniqDist = new HashSet<>(Arrays.asList(dis(p1, p2), dis(p1, p3), dis(p1, p4), dis(p2, p3), dis(p2, p4), dis(p3, p4)));

return !uniqDist.contains(0) && uniqDist.size()==2;

}

public int dis(int[] a, int[] b){

return (a[0]-b[0])\*(a[0]-b[0]) + (a[1]-b[1])\*(a[1]-b[1]);

}

}

Github Link :<https://lnkd.in/ecwtJeaz>