447 Number of Boomerangs

2018年4月11日 16:32

Given *n* points in the plane that are all pairwise distinct, a "boomerang" is a tuple of points (i, j, k) such that the distance between i and j equals the distance between i and k (**the order of the tuple matters**).
Find the number of boomerangs. You may assume that *n* will be at most **500** and coordinates of points are all in the range **[-10000, 10000]** (inclusive).
Example:
Input:
[[0,0],[1,0],[2,0]]
Output:
2
Explanation:
The two boomerangs are **[[1,0],[0,0],[2,0]]** and **[[1,0],[2,0],[0,0]]**

来自 <https://leetcode.com/problems/number-of-boomerangs/description/>
给定平面上 *n* 对不同的点，“回旋镖” 是由点表示的元组 (i, j, k)，其中 i 和 j 之间的距离和 i 和 k 之间的距离相等（**需要考虑元组的顺序**）。
找到所有回旋镖的数量。你可以假设 *n* 最大为 **500**，所有点的坐标在闭区间 **[-10000, 10000]** 中。

Solution for Python3:

```python
class Solution1:
    def numberOfBoomerangs(self, points):
        """
        :type points: List[List[int]]
        :rtype: int
        """
        from collections import Counter
        dis = []
        for i in range(len(points)):
           dis.append([self.distance(points[i], x) for x in points])
        res = 0
        print(dis)
       for list in dis:
           for i in Counter(list).values():
               res += i * (i - 1)
        return res


    def distance(self, p1, p2):
       return ((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)**0.5

class Solution2:
    def numberOfBoomerangs(self, points):
        """
        :type points: List[List[int]]
        :rtype: int
        """
        cnt = 0
        for p in points:
           dic = {}
           for q in points:
               dis = (p[0] - q[0])**2 + (p[1] - q[1])**2
               cnt += 2 * dic.setdefault(dis, 0)
               dic[dis] += 1
        return cnt
```

Solution for C++:

```cpp
class Solution1 {
public:
    int numberOfBoomerangs(vector<pair<int, int>>& points) {
        int booms = 0;
        for (auto &p : points) {
            unordered_map<double, int> ctr(points.size());
```

```
            for (auto &q : points)
                booms += 2 * ctr[hypot(p.first - q.first, p.second - q.second)]++;
        }
        return booms;
    }
};

class Solution2 {
public:
    int numberOfBoomerangs(vector<pair<int, int>>& points) {
        int booms = 0;
        for (auto &p : points) {
            unordered_map<double, int> ctr(points.size());
            for (auto &q : points)
                booms += 2 * ctr[pow((p.first - q.first), 2) + pow((p.second - q.second), 2)]++;
        }
        return booms;
    }
};
```