题目地址（447. 回旋镖的数量）

https://leetcode-cn.com/problems/number-of-boomerangs/

入选理由

1. 哈希表的一个重要作用就是空间换时间，当你想出暴力算法，可以考虑是否可用哈希表来优化。哈希表优化时间复杂度算是最最简单的一种优化手段了。相比单调栈，二分等简单很多。

题目描述

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

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。

示例：

输入：
[[0,0],[1,0],[2,0]]

输出：
2

解释：
两个回旋镖为 [[1,0],[0,0],[2,0]] 和 [[1,0],[2,0],[0,0]]

标签

- Math
- 哈希表

难度

- 中等

前置知识

- 哈希表

- 两点间距离计算方法
- 排列组合基础知识

思路

多读两遍题，大概就明白了题意：就是找出所有符合三个点 x,y,z，并且 dis(x,y)=dis(x,z)这种点的个数。首先要明确两点间距离怎么计算：

对于点 $\mathrm{x} = (\mathrm{x1}, \mathrm{x2})$，$\mathrm{y} = (\mathrm{y1}, \mathrm{y2})$，那么 $\mathrm{dis}(\mathrm{x}, \mathrm{y}) = \sqrt{(\mathrm{x1} - \mathrm{y1})^2 + (\mathrm{x2} - \mathrm{y2})^2}$

由于求的是算数平方根，所以我们算距离的时候也没必要开根号了。我们可以很容易地想到暴力解法，也就是来个**三重循环**。

```
public int numberOfBoomerangs(int[][] points) {

    if (points == null || points.length <= 2)
        return 0;

    int res = 0;

    for (int i = 0; i < points.length; i++) {

        for (int j = 0; j < points.length; j++) {

            if (i == j)
                continue;

            for (int k = 0; k < points.length; k++) {

                if (k == i || k == j)
                    continue;

                if (getDistance(points[i], points[j]) == getDistance(points[i], points[k]))
                    res++;
            }
        }
    }

    return res;
}

private int getDistance(int[] x, int[] y) {

    int x1 = y[0] - x[0];
    int y1 = y[1] - x[1];

    return x1 * x1 + y1 * y1;
}
```

这就相当于把题目翻译了一遍，但是提交就会发现 TLE 了，也不难发现时间复杂度是$O(N^3)$，

也就是我们需要优化代码了。。。

首先题目说 n 个点不同且答案考虑元组顺序，那么我们最外层循环是跑不掉了，因为需要固定每一个点。

里面两层循环可不可以优化一下呢?

其实不难想，当我们固定其中一个点 A 的时候，并且想算距离为 3 的点的个数，那么我们就找出所有和点 A 距离为 3 的点，然后来一个简单的排列组合嘛！

比如找到了 n 个距离为 3 的点，那么我们**选择第二个点有 n 种方案，选择第三个点有(n - 1)个方案**。那么固定点 A 且距离为 3 的所有可能就是 $n * (n - 1)$ 种，这只考虑了距离为 3，还有许多其他距离呢，这不就又回到了我们统计元素频率的问题上了嘛，当然哈希表用起来！不明白的看下讲义。

由于题目要求了**需要考虑元组的顺序**，因此枚举的时候不能这样枚举：

```
for i in range(n):
    for j in range(i+1,n):
        pass
```

而需要这样枚举：

```
for i in range(n):
    for j in range(n):
        pass
```

关键点

- 使用哈希表预处理点信息
- n 个数取两个数的全排列种类是 A_n^2，也就是 n*(n-1)

代码

代码支持： Java，Python3

Java Code:

```
public int numberOfBoomerangs(int[][] points) {

    if (points == null || points.length <= 2)
        return 0;
```

```
    int res = 0;
    Map<Integer, Integer> equalCount = new HashMap<>();

    for (int i = 0; i < points.length; ++i) {

        for (int j = 0; j < points.length; ++j) {

            int dinstance = getDistance(points[i], points[j]);
            equalCount.put(dinstance, equalCount.getOrDefault(dinstance, 0) + 1);
        }

        for (int count : equalCount.values())
            res += count * (count - 1);
        equalCount.clear();
    }

    return res;
}

private int getDistance(int[] x, int[] y) {

    int x1 = y[0] - x[0];
    int y1 = y[1] - x[1];

    return x1 * x1 + y1 * y1;
}
```

Python3 Code:

```
class Solution:
    def numberOfBoomerangs(self, points: List[List[int]]) -> int:
        n = len(points)
        ans = 0
        for i in range(n):
            m = collections.defaultdict(int)
            for j in range(n):
                dist = abs(points[i][0] - points[j][0]) ** 2 + abs(points[i][1] - points[j][1]) ** 2
                m[dist] += 1
            for count in m.values():
                ans += count * (count-1)
        return ans
```

复杂度分析

- 时间复杂度: $O(n^2)$
- 空间复杂度：$O(n)$

上一页 下一页