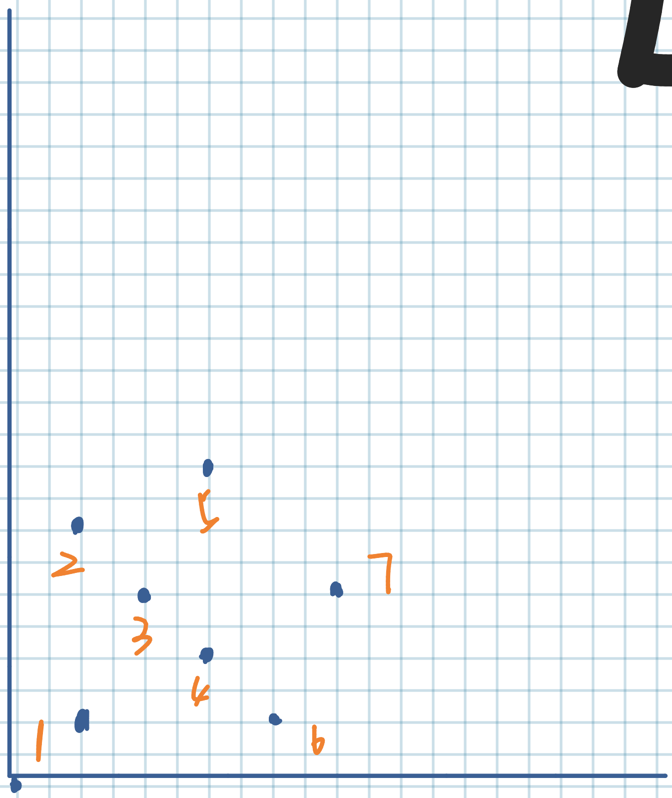
# LC 149 Hard




[1,1] [3,2] [5,3] [4,1] [2,3], [1,4] [3,5]

⇓

| [1,1] | [1,4] | [2,3] | [3,2] | [3,5] | [4,1] | [5,3] |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

精度如何照顾？容易overflow

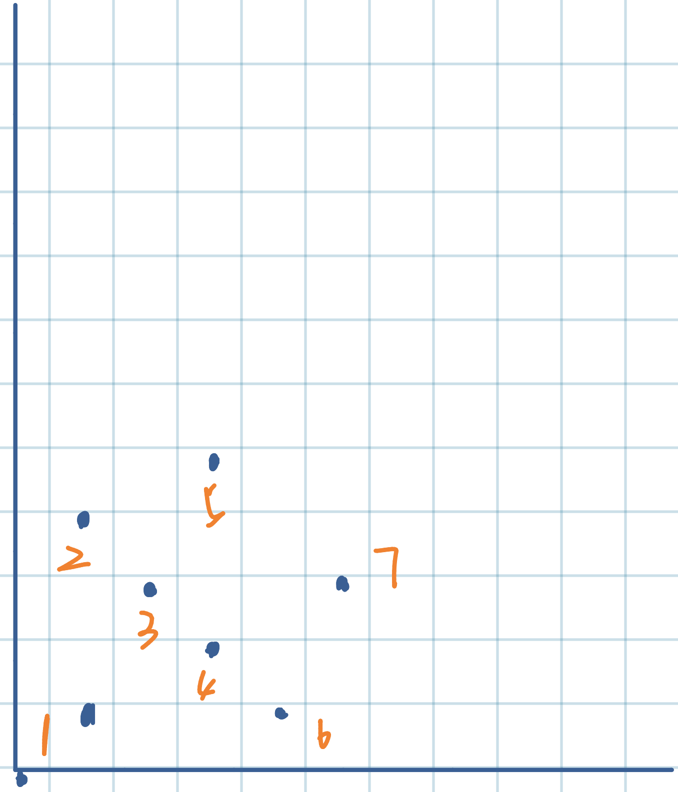
我的想法一：

先排序，然后存斜率和点.

看完大佬解法：

思路：假如某个点在那条拥有最多点的点上，则在以此点为中心的连线过程中，可以把该连的点都连上。

如起始点为右图中点①：




依次连 1-2，1-3，1-4 1-7，1-6

则在共 n-1 轮（有n个点，n-1条线段）中，

此为第1轮。将第一轮的最大点数保存在

全局变量 result 中，并清空记录点数与斜率的 dictionary

result = 2+1 = 3 （别忘记，2条线段对应3个点）

第二轮中，以点②为中心，

依次连 2-3 2-4 2-6，2-5，2-7

result = max(result, 3+1) = max(3,4) = 4

清空dictionary

依此类推，最后 return result 值

## 问题1：如何存斜率？

由直线方程可知 $k = \frac{y_1 - y_0}{x_1 - x_0}$

其中 $(x_0, y_0)$ 为中心点

（这种以点为中心的计算方式带来的优势是，只要确定斜率相同，一定是在同一直线上，而不是在某条平行线上）

由于这里有除法，$x_1 - x_0$ 作为除数，可能为0

此外，除法得出的数可能有精度问题

例如 $\frac{2^{32}-7}{2^{32}}$ 与 $\frac{2^{32}-9}{2^{32}}$ 可能会得到同一个值.

解决：用斜率的分子与分母来表示一条直线

如 $k = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y}{\Delta x}$，其中，

$\Delta y$ 与 $\Delta x$ 均为整数，且互质！

如 $\Delta y = 4$，$\Delta x = 2$，则除以它们的最大公约数

下一页总结了最大公约数的求法

求最大公约数

- 辗转相除法
- 更相减损法 → 如果 x 与 y 不为正数，就求不出来！
- 质因数分解法
- 短除法

如 $x=15$，$y=-3$，会无限循环

while $x \neq y$:
$$\begin{cases} a = \max(x, y) \\ b = \min(x, y) \\ x = a - b \\ y = b \end{cases}$$

若 $x=-15$，$y=-3$，无限循环

若 $x=1$，$y=0$，无限循环

可以直接返回非零值

e.g. 27 与 39 的最大公约数

| x | y | temp |
|---|---|---|
| -27 | -3 | 0 |

code:

```
def gcd(x, y):
    temp = x % y
    while temp != 0
        x = y
        y = temp
        temp = x % y

    return y
```

$-15-(-3)=-12$

$-3-(-12)=9$

$9-(-12)=21$

$21-(-12)$.

由上页，我们确定使用 辗转相除法

来计算最大公约数.

需要注意的问题是，dx, dy 不一定为正，我们会得到什么？

例子：　A ( 1, 1 )

B ( 0, 0 )　　以A为中心计算：

C ( 2, 2 )

AB：$\frac{0-1}{0-1} = \frac{-1}{-1}$　　$dy=-1$, $dx=-1$

AC：$\frac{2-1}{2-1} = \frac{1}{1}$　　$dy=1$, $dx=1$

D ( 0, 2 )

E ( 2, 0 )

$AD = \frac{2-1}{0-1} = \frac{1}{-1}$　　$dy=1$, $dx=-1$

$AE = \frac{0-1}{2-1} = \frac{-1}{1}$　　$dy=-1$, $dx=1$

辗转相除法代码：

```
def gcd(x, y):
    temp = x % y
    while temp != 0:
        x = y
        y = temp
        temp = x % y
    return y
```

return 作为除数的 y

对 $x=-4$, $y=2$ :

| x | y | temp |
| --- | --- | --- |
| -4 | 2 | -4 % 2 = 0 |

return y : return 2

对 $x=4$, $y=-2$

| x | y | temp |
| --- | --- | --- |
| 4 | -2 | 0 |

return y : return -2

$y = y \div 2 = 1$

$x = x \div 2 = -2$

$x = x \div (-2) = -2$

$y = y \div (-2) = 1$

得到同质的结果

问题 2：有重合的点怎么办？

有重合的点时，必有 $dx = dy = 0$

此时记录 overlap 的值，并处理下一个点

在判定和更新 result 时，别忘了加上 overlap.

问题 3：有斜率为 0 或无穷大的线怎么办？

$dx = 0$？ dy 设为 1

$dy = 0$？ dx 设为 1

这样，例如 $dy = 0$ 时，不论 $dx = -1$，$dx = 5$，都设为 1 即可.

那么，平行于 x 轴的线标记为 (1, 0)

平行于 y 轴的线标记为 (0, 1)

问题 4：各种 edge cases：

① points 数组里点少于 2 个：

返回 length of points

② points 里只有重合点：

返回 overlap + 1，如记了有一个点有该点 overlap，那返回值应当是 2.

③ 到最后一轮时，for j in range (i+1, len(points))

这个循环不执行，则 dictionary 为空：

A. 记得判断 dictionary 是否为空，

是空时，不能用 max(dictionary, key=dictionary.get) 方法

求该 dictionary 中最大的 value，

会报错说 max() arg is empty.

B: for i in range(len(points)-1)

而不是 for i in range(len(points))

C: 当 points 中点全 overlap 时，slope 也会为空字典结果哦

code:

```
def maxPoint(self, points):
    if len(points) < 3: return len(points)
    result = 0
    for i in range(len(points)-1):
        slope, overlap = {}, 0
        for j in range(i+1, len(points)):
            dx = points[j][0] - points[i][0]
            dy = points[j][1] - points[i][1]
            if dx == 0 and dy == 0:
                overlap += 1
                continue
            elif dx == 0: dy = 1
            elif dy == 0: dx = 1
            else:
                gcd = self.gcd(dx, dy)
                dx /= gcd
                dy /= gcd
            if (dx, dy) in slope.keys():
                slope[(dx, dy)] += 1
            else: slope[(dx, dy)] = 2
        if slope:
            result = max(res, slope[max(slope, key=slope.get)] + overlap)
        else: result = max(overlap+1, result)
    return result
```

时间复杂度：

由于 dictionary 查找默认为 $O(1)$

所以本解法时间复杂度 $O(n^2)$

暴力破解将是 $O(n^3)$：

连两个点成线，有 $O(n^2)$ 种连法

然后判断剩下 $n-1$ 个点哪些点在线上

$\therefore O(n^2 * (n-1)) = O(n^3)$

扒下大佬的 Java 解法：

```java
class Solution {
    public int maxPoints(int[][] points) {
        if (points == null) return 0;
        if (points.length < 3) return points.length;
        Map<Integer, Map<Integer, Integer>> map = new HashMap<Integer, Map<Integer, Integer>>();
        int result = 0;
        for (int i = 0; i < points.length; i++){
            int overlap = 0, dx = 0, dy = 0, max = 0;
            for (int j = i+1; j < points.length; j++){
                dx = points[j][0] - points[i][0];
                dy = points[j][1] - points[i][1];
                if (dx == 0 && dy == 0) {
                    overlap++;
                    continue;
                }
                else if (dx == 0) dy = 1;
                else if (dy == 0) dx = 1;
                else {
                    int g = gcd(dx, dy);
                    dx /= g;
                    dy /= g;
                }
                    if (map.containsKey(dx)){
                        if (map.get(dx).containsKey(dy)) map.get(dx).put(dy, map.get(dx).get(dy) + 1);
                        else map.get(dx).put(dy, 1);
                    }
                    else{
                        Map<Integer, Integer> new_line = new HashMap<Integer, Integer>();
                        new_line.put(dy, 1);
                        map.put(dx, new_line);
                    }
                max = Math.max(max, map.get(dx).get(dy));
            }
            result = Math.max(result, max + overlap + 1);
            map.clear();
        }
        return result;
    }

    public int gcd(int x, int y){
        int temp = x % y;
        while (temp != 0){
            x = y;
            y = temp;
            temp = x % y;
        }
        return y;
    }
}
```

时间复杂度：$O(n^2)$

这份代码可以优化的地方：

可以将 dx, dy 映射成 hash 值

这样就不需要使用

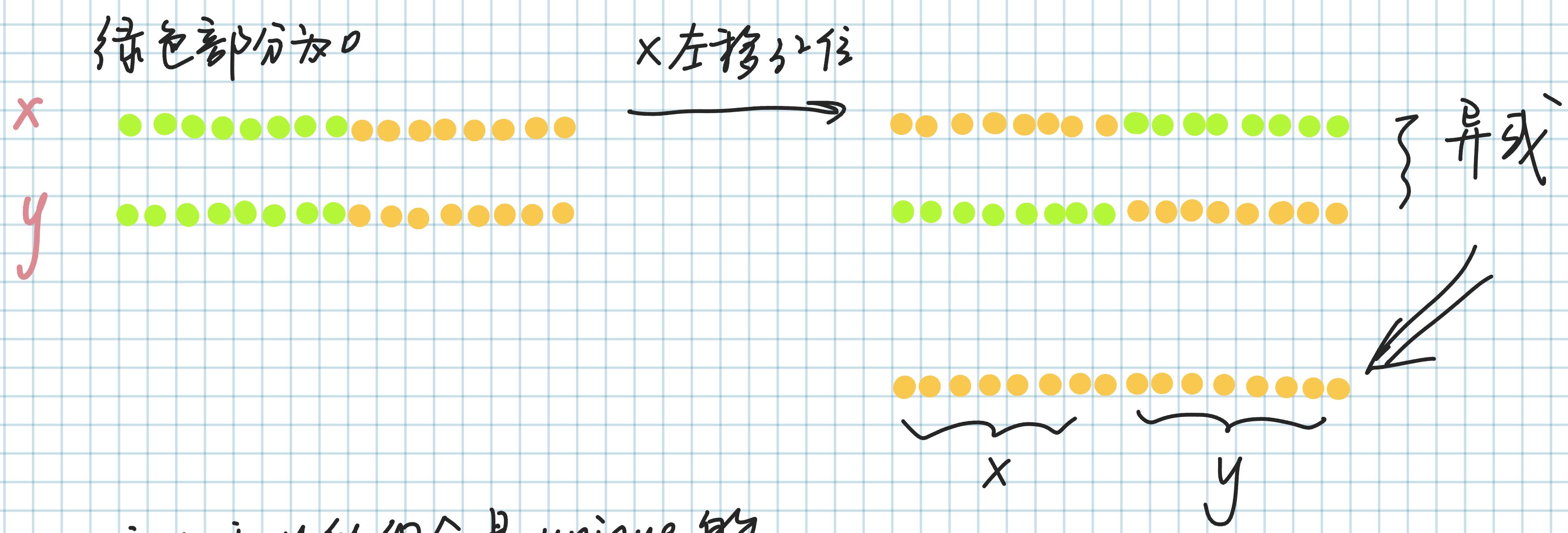
Map<int, Map<int, int>> 这样变态的数据结构了.

如何映射呢？

考虑到 dx, dy 均为 int，默认为 32位 int.

则一个类型为 long，即 64位的 hash 码，可以避免“hash 碰撞”

```
private int getSlopeKey(long x, long y)
{
    return (x << 32) ^ y;
}
```




则只要 x 和 y 的组合是 unique 的

$(x<<32) \wedge y$ 的值也是 unique 的.

```java
public int maxPoints(int[][] points){
    if (points == null) return 0;
    if (points.length < 3) return points.length;
    int result = 0;
    Map<Long, Integer> slope = new HashMap<Long, Integer> ();
    for (int i = 0; i < points.length; i++){
        int overlap = 0, max = 0;
        slope.clear();
        for (int j = i + 1; j < points.length; j++){
            int dx = points[i][0] - points[j][0];
            int dy = points[i][1] - points[j][1];
            if (dx == 0 && dy == 0){
                overlap ++;
                continue;
            }
            else if (dx == 0) dy = 1;
            else if (dy == 0) dx = 1;
            else{
                int g = gcd(dx, dy);
                dx /= g;
                dy /= g;
            }
            long hashkey = getSlopeKey(dx, dy);
            if (slope.containsKey(hashkey)) slope.put(hashkey, slope.get(hashkey) + 1);
            else slope.put(hashkey, 1);
            max  = Math.max(max, slope.get(hashkey));
        }
        result = Math.max(result, max + overlap + 1);
    }
    return result;
}

private long getSlopeKey(long x, long y){
    return (x << 32) ^ y;
}
```