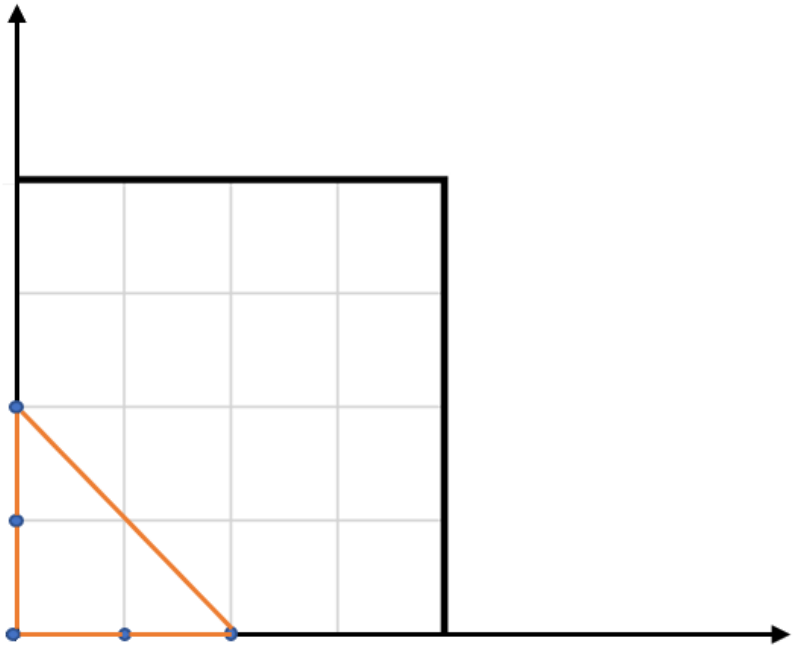
Largest Triangle Area

You have a list of points in the plane. Return the area of the largest triangle that can be formed by any 3 of the points.

Example:
Input: `points = [[0,0],[0,1],[1,0],[0,2],[2,0]]`
Output: `2`
Explanation:
`The five points are show in the figure below. The red triangle is the largest.`



Notes:

- `3 <= points.length <= 50`.
- No points will be duplicated.
- `-50 <= points[i][j] <= 50`.
- Answers within `10^-6` of the true value will be accepted as correct.

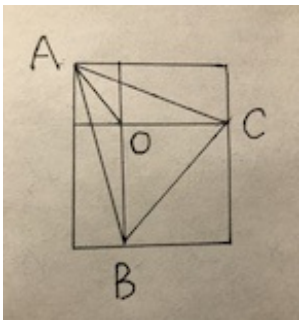
Solution 1

Explanaiton
Burete force loop on all combinations of three points and calculate the area of these three points.
If you google “three pointes triangle area formula”, you can find the answer with the first result in second.

Time complexity
`O(N^3)` solution, but `N <= 50`, so it’s fast enough.
You may find convex hull first as @weidairpi replies. It help improve to O(M^3 + NlogN) in the best case where M is the number of points on the hull.
But it make this easy problem complex and it stays same complexity in the worst case.

Prove 1
Well, someone complains the situation without any formula.
In fact the formula is not that difficult to find out.

For this case:



We can calculate the area as follow:

$$S_\Delta ABC = S_\Delta AOB + S_\Delta BOC + S_\Delta COA$$

$$S_\Delta ABC = \frac{1}{2}(x_b - x_a)(y_c - y_b) + \frac{1}{2}(x_c - x_b)(y_a - y_c) + \frac{1}{2}(x_c - x_b)(y_c - y_b)$$

$$S_\Delta ABC = \frac{1}{2}(x_a y_b + x_b y_c + x_c y_a - x_a y_c - x_c y_b - x_b y_c)$$

In the result A,B,C are symmetrical, so it won’t matter what order we name it.
In this case, we calculate the total area by addition three triangle.
In the other cases, you may need to use substraction and it’s quite the same process.

Prove 2
If you are familar with vector product. The result is quite obvious.

$$Area = \frac{1}{2}|AB \times AC|$$

$$Area = \frac{1}{2}|(x_b - x_a, y_b - y_a)) \times (x_c - x_a, y_c - y_a))|$$

$$Area = \frac{1}{2}|(x_b - x_a)(y_c - y_a) - (x_c - x_a, y_b - y_a))|$$

$$Area = \frac{1}{2}|x_a y_b + x_b y_c + x_c y_a - x_a y_c - x_c y_b - x_b y_c|$$

C++:

```
    double largestTriangleArea(vector<vector<int>>& p) {
        double res = 0;
        for (auto &i : p)
            for (auto &j : p)
                for (auto &k : p)
            res = max(res, 0.5 * abs(i[0] * j[1] + j[0] * k[1] + k[0] * i[1]- j[0]
* i[1] - k[0] * j[1] - i[0] * k[1]));
        return res;
    }
```

Java:

```
    public double largestTriangleArea(int[][] p) {
        double res = 0;
        for (int[] i: p)
            for (int[] j: p)
                for (int[] k: p)
            res = Math.max(res, 0.5 * Math.abs(i[0] * j[1] + j[0] * k[1] + k[0] * i
[1]- j[0] * i[1] - k[0] * j[1] - i[0] * k[1]));
        return res;
    }
```

Python

```
    def largestTriangleArea(self, p):
        def f(p1, p2, p3):
            (x1, y1), (x2, y2), (x3, y3) = p1,p2,p3
            return 0.5 * abs(x2 * y3 + x1 * y2 + x3 * y1 - x3 * y2 - x2 * y1 - x1
* y3)
        return max(f(a, b, c) for a, b, c in itertools.combinations(p, 3))
```

1-line Python

```
def largestTriangleArea(self, p):
        return max(0.5 * abs(i[0] * j[1] + j[0] * k[1] + k[0] * i[1]- j[0] * i[1]
- k[0] * j[1] - i[0] * k[1])
            for i, j, k in itertools.combinations(p, 3))
```

written by lee215 original link here

Solution 2

So, during the contest I wanted to use the brute-force approach, which is
Iterate over the with three loops, calculate each area for the three points and keep track of the max area

This gives complexity of O(N^3) . It looks like this approach is generally accepted but I got stumped when I could not even recall the formula for calculating the area when given three points.

Assuming this is an interview, you do not have the option of googling the formula and I honestly don't think most people walking around kow that formula off the top of their heads... I could not come up with an alternate solution either.

So, did anyone else have a solution which does not involve the Heron's formula or the popular predefined one like this

```
Math.abs(0.5*(points[0][0]*(points[1][1]-points[2][1])+points[1][0]*(points[2][1]-p
oints[0][1])+points[2][0]*(points[0][1]-points[1][1]))
```

Thanks.

written by vanderpuye original link here

Solution 3

for triangle with `(x1,y1) (x2,y2) (x3,y3)` points,
the area is calculated as `0.5 * abs(x1y2+x2y3+x3y1-y1x2-y2x3-y3x1)`

```
class Solution {

    let x = 0
    let y = 1

    func largestTriangleArea(_ points: [[Int]]) -> Double {
        var ans = 0.0
        for x in stride(from: 0, through: points.count - 3, by: 1) {
            for y in stride(from: x + 1, through: points.count - 2, by: 1) {
                for z in stride(from: y + 1, through: points.count - 1, by: 1) {
                    let array = [points[x], points[y], points[z]]
                    ans = max(ans, area(array))
                }
            }
        }
        return ans
    }

    private func area(_ points: [[Int]]) -> Double {
        var ans = points[0][x] * points[1][y]
        ans += points[1][x] * points[2][y]
        ans += points[2][x] * points[0][y]

        ans -= points[0][y] * points[1][x]
        ans -= points[1][y] * points[2][x]
        ans -= points[2][y] * points[0][x]

        return abs(0.5 * Double(ans))
    }
}
```

written by ejames original link here