Convex Polygon

Given a list of points that form a polygon when joined sequentially, find if this polygon is convex (Convex polygon definition).

Note:

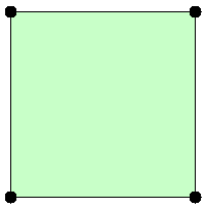
1. There are at least 3 and at most 10,000 points.
2. Coordinates are in the range -10,000 to 10,000.
3. You may assume the polygon formed by given points is always a simple polygon (Simple polygon definition). In other words, we ensure that exactly two edges intersect at each vertex, and that edges otherwise **don't intersect each other**.

Example 1:

```
[[0,0],[0,1],[1,1],[1,0]]

Answer: True

Explanation:
```

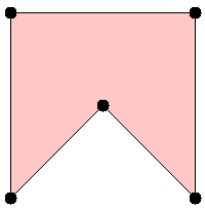


Example 2:

```
[[0,0],[0,10],[10,10],[10,0],[5,5]]

Answer: False

Explanation:
```



Solution 1

get the **cross product** of the sequential input edge a, b as **tmp**, then:

if **tmp** == 0, a -> b 180° on the same line;
elif **tmp** > 0, a -> b clockwise;
else **tmp** < 0, a -> anticlockwise;

tmp = (p1[0]-p0[0])*(p2[1]-p0[1])*-(p2[0]-p0[0])*(p1[1]-p0[1])

```
class Solution(object):
    def isConvex(self, points):
        last = 0
        for i in xrange(2, len(points) + 2):
            p0, p1, p2 = points[(i - 2) % len(points)], points[(i - 1) % len(poin
ts)], points[i % len(points)]
            tmp = (p1[0]-p0[0])*(p2[1]-p0[1])-(p2[0]-p0[0])*(p1[1]-p0[1])
            if last * tmp < 0:
                return False
            last = tmp
        return last * tmp >= 0
```

Update instead of just maintaining the sequential cross product result, any of the two cross product shouldn't multiplies to minus:

```
class Solution(object):
    def isConvex(self, points):
        last, tmp = 0, 0
        for i in xrange(2, len(points) + 3):
            p0, p1, p2 = points[(i - 2) % len(points)], points[(i - 1) % len(poin
ts)], points[i % len(points)]
            tmp = (p1[0]-p0[0])*(p2[1]-p0[1])-(p2[0]-p0[0])*(p1[1]-p0[1])
            if tmp:
                if last * tmp < 0:
                    return False
                last = tmp
        return True
```

written by Ipeq1 original link here

Solution 2

Well, I have to say that this problem is beyond my knowledge.@Ipeq1 and @zzg_zzm have explained how to solve this problem in their posts
https://discuss.leetcode.com/topic/70643/i-believe-this-time-it-s-far-beyond-my-ability-to-get-a-good-grade-of-the-contest
https://discuss.leetcode.com/topic/70664/c-7-line-o-n-solution-to-check-convexity-with-cross-product-of-adajcent-vectors-detailed-explanation
The algorithm itself is not hard but I have no idea there exists such a way to determine if a polygon is convex or not. Laugh at me for my ignorance... I believe 90% of programmers can solve this problem if they were given the formula.
Anyway, following is the Java solution with in-line explanation. Accepted, 32ms.
Reference: http://csharphelper.com/blog/2014/07/determine-whether-a-polygon-is-convex-in-c/

```java
public class Solution {
    public boolean isConvex(List<List<Integer>> points) {
        // For each set of three adjacent points A, B, C, find the cross product
AB · BC. If the sign of
        // all the cross products is the same, the angles are all positive or nega
tive (depending on the
        // order in which we visit them) so the polygon is convex.
        boolean gotNegative = false;
        boolean gotPositive = false;
        int numPoints = points.size();
        int B, C;
        for (int A = 0; A < numPoints; A++) {
            // Trick to calc the last 3 points: n - 1, 0 and 1.
            B = (A + 1) % numPoints;
            C = (B + 1) % numPoints;

            int crossProduct =
                crossProductLength(
                    points.get(A).get(0), points.get(A).get(1),
                    points.get(B).get(0), points.get(B).get(1),
                    points.get(C).get(0), points.get(C).get(1));
            if (crossProduct < 0) {
                gotNegative = true;
            }
            else if (crossProduct > 0) {
                gotPositive = true;
            }
            if (gotNegative && gotPositive) return false;
        }

        // If we got this far, the polygon is convex.
        return true;
    }

    // Return the cross product AB x BC.
    // The cross product is a vector perpendicular to AB and BC having length |AB
| * |BC| * Sin(theta) and
    // with direction given by the right-hand rule. For two vectors in the X-Y pla
ne, the result is a
    // vector with X and Y components 0 so the Z component gives the vector's len
gth and direction.
    private int crossProductLength(int Ax, int Ay, int Bx, int By, int Cx, int Cy)
    {
        // Get the vectors' coordinates.
        int BAx = Ax - Bx;
        int BAy = Ay - By;
        int BCx = Cx - Bx;
        int BCy = Cy - By;

        // Calculate the Z coordinate of the cross product.
        return (BAx * BCy - BAy * BCx);
    }
}
```

written by shawngao original link here

Solution 3

Re: I believe this time it's far beyond my ability to get a good grade of the contest!

Great solution inspired by @Ipeq1! Here is a C++ version with extracted determinant calculation.

The key observation for convexity is that **vector p_{i+1}-p_i always turns to the same direction to p_{i+2}-p_i formed by any 3 sequentially adjacent vertices, i.e., cross product (p_{i+1}-p_i) x (p_{i+2}-p_i) does not change sign when traversing sequentially along polygon vertices**.

Note that for any 2D vectors $\mathbf{v_1}$, $\mathbf{v_2}$,

- $\mathbf{v_1} \times \mathbf{v_2} = \det([\mathbf{v_1}, \mathbf{v_2}])$

which is the determinant of 2x2 matrix $[\mathbf{v_1}, \mathbf{v_2}]$. And the sign of $\det([\mathbf{v_1}, \mathbf{v_2}])$ represents the positive z-direction of right-hand system from $\mathbf{v_1}$ to $\mathbf{v_2}$. So $\det([\mathbf{v_1}, \mathbf{v_2}]) \geq 0$ if and only if $\mathbf{v_1}$ turns at most 180 degrees **counterclockwise** to $\mathbf{v_2}$.

```
    bool isConvex(vector<vector<int>>& p) {
      long n = p.size(), prev = 0, cur;
      for (int i = 0; i < n; ++i) {
        vector<vector<int>> A; // = {p[(i+1)%n]-p[i], p[(i+2)%n]-p[i]}
        for (int j = 1; j < 3; ++j) A.push_back({p[(i+j)%n][0]-p[i][0], p[(i+j)%n
][1]-p[i][1]});
        if (cur = det2(A)) if (cur*prev < 0) return false; else prev = cur;
      }
      return true;
    }
    // calculate determinant of 2*2 matrix A
    long det2(vector<vector<int>>& A) { return A[0][0]*A[1][1] - A[0][1]*A[1][0];
}
```

written by zzg_zzm original link here