Codility_

CodeCheck Report: trainingSHQWMP-TF7

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
PolygonConcavityIndex Python	5 min	58%

Total score

Tasks Details

1. PolygonConcavityIndex

Hard

Check whether a given polygon in a 2D plane is convex; if not, return the index of a vertex that doesn't belong to the convex hull.

Task Score	Correctness	Performance
58%	66%	33%

Task description

An array A of points in a 2D plane is given. These points represent a polygon: every two consecutive points describe an edge of the polygon, and there is an edge connecting the last point and the first point in the array.

A set of points in a 2D plane, whose boundary is a straight line, is called a *semiplane*. More precisely, any set of the form {(x, y) : ax + by ≥ c} is a semiplane. The semiplane contains its boundary.

A polygon is *convex* if and only if, no line segment between two points on the boundary ever goes outside the polygon.

For example, the polygon consisting of vertices whose Cartesian coordinates are consecutively:

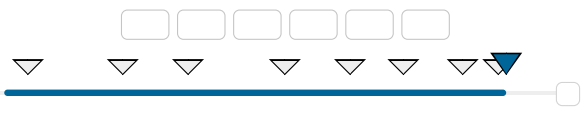
```
(-1, 3)   (3, 1)   (0, -1)   (-2, 1)
```

is convex.

Solution

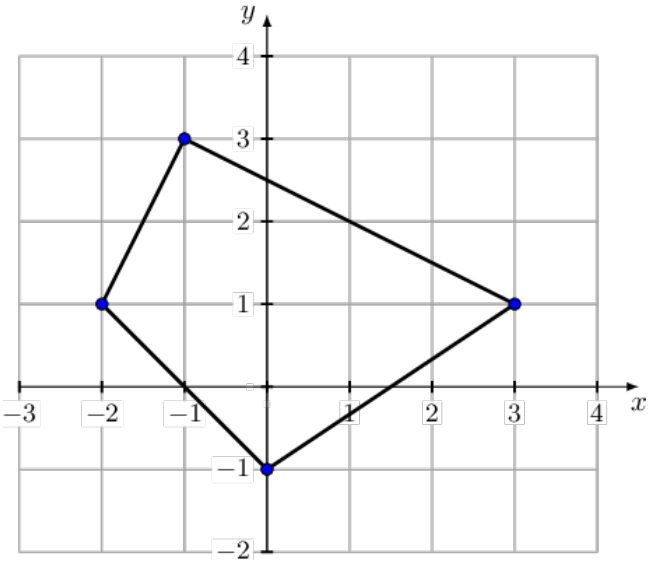
Programming language used:	Python
Total time used:	5 minutes
Effective time used:	5 minutes
Notes:	*not defined yet*

Task timeline



09:51:34 09:55:50

Code: 09:55:49 UTC, py, show code in pop-up

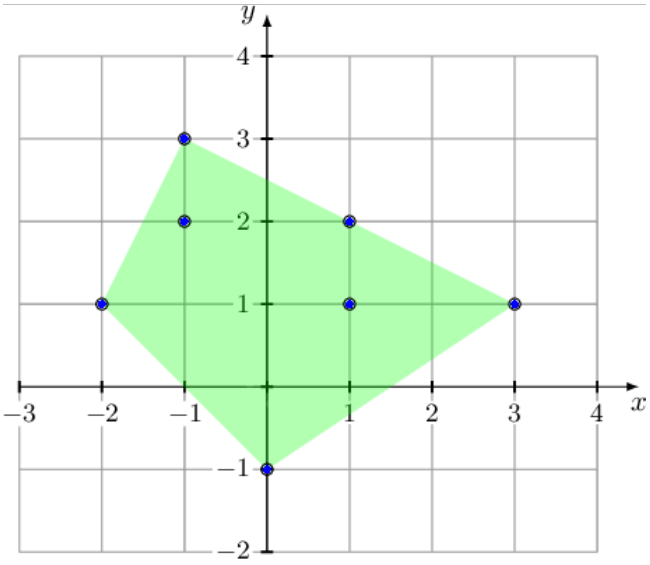


The *convex hull* of a finite set of points in a 2D plane is the smallest convex polygon that contains all points in this set. For example, the convex hull of a set consisting of seven points whose Cartesian coordinates are:

```
(−1, 3)   (1, 2)   (3, 1)   (1, 1)   (0, −1)
(−2, 1)   (−1, 2)
```

is a polygon that has five vertices. When traversed clockwise, its vertices are:

```
(−1, 3)   (1, 2)   (3, 1)   (0, −1)   (−2, 1)
```



If a polygon is concave (that is, it is not convex), it has a vertex which does not lie on its convex hull border. Your assignment is to find such a vertex.

Assume that the following declarations are given:

```
from dataclasses import dataclass, field

@dataclass
class Point2D:
    x: int
    y: int
```

Write a function:

```
def solution(A)
```

that, given a non-empty array A consisting of N elements describing a polygon, returns −1 if the polygon is convex.

final, score: **58**

```
1  from dataclasses import dataclass, field
2  import math
3
4  @dataclass
5  class Point2D:
6      x: int
7      y: int
8
9  def polar_angle(point, ref_point):
10     return math.atan2(point.y - ref_point.y,
11
12 def left_turn(p1, p2, p3):
13     return (p2.x - p1.x) * (p3.y - p1.y) - (p
14
15 def graham_scan(points):
16     n = len(points)
17     if n <= 3:
18         return points
19
20     # Find the reference point (lowest y-coor
21     ref_point = min(points, key=lambda p: (p.
22
23     # Sort the points based on their polar an
24     sorted_points = sorted(points, key=lambda
25
26     # Build the convex hull using a stack
27     stack = [sorted_points[0], sorted_points[
28     for i in range(2, n):
29         while len(stack) >= 2 and not left_tu
30             stack.pop()
31         stack.append(sorted_points[i])
32
33     return stack
34
35 def solution(A):
36     convex_hull = graham_scan(A)
37
38     for i, point in enumerate(A):
39         if point not in convex_hull:
40             return i
41
42     # If all points in the original array are
43     return -1
```

Analysis summary

The following issues have been detected: wrong answers, timeout errors.

Analysis

expand all	**Example tests**	
▶	example1 first example test	✔ **OK**
▶	example2 second example test	✔ **OK**
expand all	**Correctness tests**	
▶	simple0 boomerang	✔ **OK**
▶	simple1 star	✔ **OK**
▶	simple2	✔ **OK**
▶	simple3 the polygon has exactly one angle equals to (90 + epsilon) degrees	✘ **WRONG ANSWER** Given polygon is convex, but got 3

Otherwise, the function should return the index of any point that doesn't belong to the convex hull border. Note that consecutive edges of the polygon may be collinear (that is, the polygon might have 180−degrees angles).

To access the coordinates of the K-th point (where 0 ≤ K < N), use the following syntax:

- `A[K].x` to access the x-coordinate,
- `A[K].y` to access the y-coordinate.

For example, given array A such that:

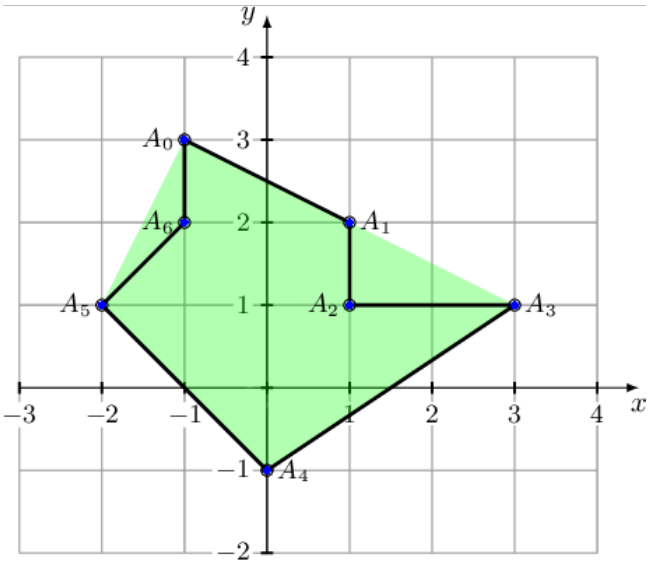
```
A[0].x = −1  A[0].y =  3
A[1].x =  1  A[1].y =  2
A[2].x =  3  A[2].y =  1
A[3].x =  0  A[3].y = −1
A[4].x = −2  A[4].y =  1
```

the function should return −1, as explained in the example above.

However, given array A such that:

```
A[0].x = −1  A[0].y =  3
A[1].x =  1  A[1].y =  2
A[2].x =  1  A[2].y =  1
A[3].x =  3  A[3].y =  1
A[4].x =  0  A[4].y = −1
A[5].x = −2  A[5].y =  1
A[6].x = −1  A[6].y =  2
```

the function should return either 2 or 6. These are the indices of the polygon lying strictly in its convex hull (that is, not on the convex hull border).



Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [3..10,000];
- the coordinates of each point in array A are integers within the range [−1,000,000,000..1,000,000,000];
- no two edges of the polygon A intersect, other than meeting at their endpoints;
- array A does not contain duplicate points.

▶ corner_cases corner cases	✔ OK
▶ cyclic all possible representations of a simple case	✘ WRONG ANSWER Got 0, expected 3
▶ collinear_vertices tests with many collinear triples of vertices	✘ WRONG ANSWER Given polygon is convex, but got 1
▶ medium1	✔ OK
▶ medium2	✔ OK
expand all	**Performance tests**
▶ big1 almost diamond	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ big2	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ big3	✔ OK