Codility_

CodeCheck Report: training85X4CS-FC8

Test Name:

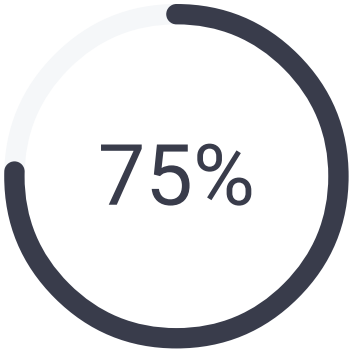
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
PolygonConcavityIndex Java 8 ⚠	1 min	75%

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
75%	77%	66%

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:

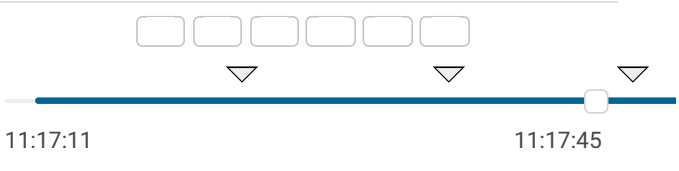
```
(-1, 3)   (3, 1)   (0, -1)   (-2, 1)
```

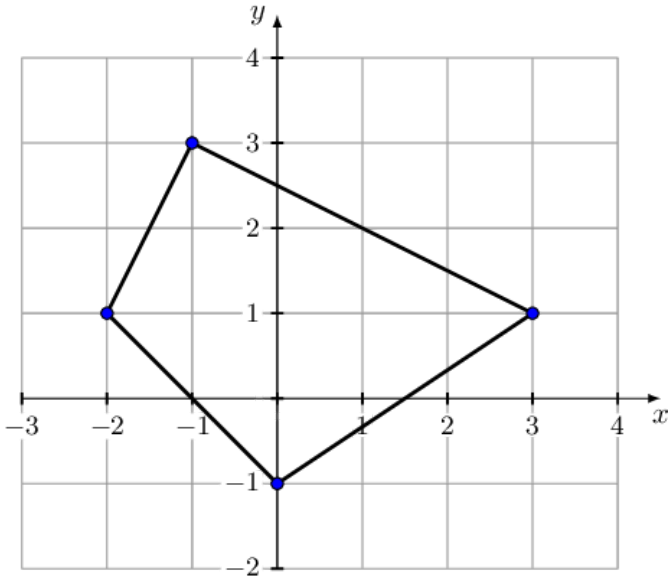
is convex.

Solution

Programming language used:	Java 8
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline



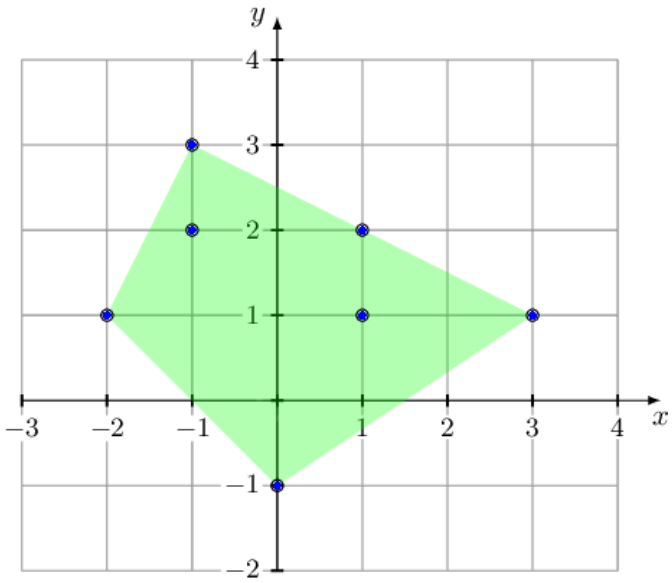


The *convex hull* of a finite set of points in a 2D plane is the smallest convex polygon that contains all points in this set. For example, the convex hull of a set consisting of seven points whose Cartesian coordinates are:

```
  (−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:

```
class Point2D {
  public int x;
  public int y;
}
```

Write a function:

```
class Solution { public int
solution(Point2D[] A); }
```

that, given a non-empty array A consisting of N elements describing a polygon, returns −1 if the polygon is convex. Otherwise, the function should return the index of any point that

Code: 11:17:45 UTC, java, final, score: **75** show code in pop-up

```
1  // you can also use imports, for example:
2  import java.util.*;
3
4  // you can write to stdout for debugging purpo
5  // System.out.println("this is a debug message
6
7  class Solution {
8          private static class Vector2D {
9                  public long x;
10                 public long y;
11
12                 public Vector2D(Point2D p1, Po
13                         x = p2.x - p1.x;
14                         y = p2.y - p1.y;
15                 }
16
17                 public static long multiply(Ve
18                         return v1.x * v2.y - v
19                 }
20
21                 public static int getAngleSign
22                         long product = multipl
23                         return product > 0 ? 1
24                                         produc
25
26                 }
27         }
28
29         public int solution(Point2D[] A) {
30                 int N = A.length;
31
32                 int previousAngleSign = 0;
33                 int firstIndex = -1;
34
35                 Vector2D previousVector = new
36                 Vector2D currentVector;
37
38                 for (int i = 0; i < N; i++, pr
39                         currentVector = new Ve
40                         int angleSign = Vector
41                         if (angleSign != 0) {
42                                 previousAngleS
43                                 firstIndex = i
44                                 break;
45                         }
46                 }
47
48                 Deque<Integer> angleSignChange
49                 if (previousAngleSign == 0) {
50                         throw new IllegalArgum
51                 }
52
53                 // N + 2 tests
54                 for (int i = 0; i <= N + 1; i-
55                         currentVector = new Ve
56                         int angleSign = Vector
57                         if (angleSign == 0) {
58                                 angleSignChang
59                                 angleSignChang
60                                 continue;
61                         }
62
63                         if (angleSign * previo
64                                 angleSignChang
65                         } else if (angleSignCh
66                                 return (angleS
67                         }
68
69                         previousAngleSign = an
70                         angleSignChangeIndicie
71                 }
72
73                 return -1;
74         }
75 }
```

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:

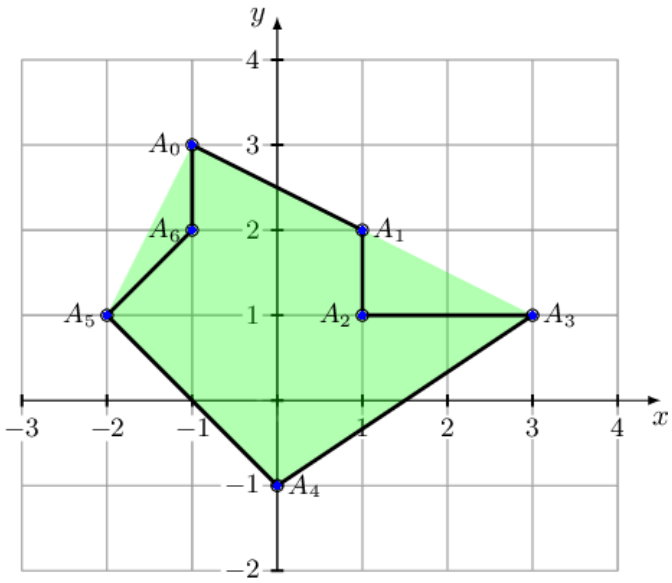
```
A[0].x = −1  A[0].y =  3
A[1].x =  1  A[1].y =  2
A[2].x =  3  A[2].y =  1
A[3].x =  0  A[3].y = −1
A[4].x = −2  A[4].y =  1
```

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.

Analysis summary

The following issues have been detected: wrong answers.

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	✔ OK
▶	corner_cases corner cases	✘ WRONG ANSWER Got 1, but 1 is an index of vertex that belongs to convex hull
▶	cyclic all possible representations of a simple case	✔ OK
▶	collinear_vertices tests with many collinear triples of vertices	✔ OK
▶	medium1	✔ OK
▶	medium2	✘ WRONG ANSWER Got -1, but given polygon isn't convex
expand all	Performance tests	
▶	big1 almost diamond	✘ WRONG ANSWER Got -1, but given polygon isn't convex
▶	big2	✔ OK
▶	big3	✔ OK