

---

**bentley\_ottmann**

*Release 7.3.0*

**Azat Ibrakov**

**Jun 14, 2022**



# CONTENTS

<b>Python Module Index</b>	<b>3</b>
<b>Index</b>	<b>5</b>



---

**Note:** If object is not listed in documentation it should be considered as implementation detail that can change and should not be relied upon.

---

`bentley_ottmann.planar.contour_self_intersects` (*contour*: `ground.hints.Contour`, \*, *context*: `Optional[ground.base.Context]` = `None`) → `bool`

Checks if contour has self-intersection.

Based on Bentley-Ottmann algorithm.

**Time complexity:**  $O(\text{len}(\text{contour.vertices}) * \log \text{len}(\text{contour.vertices}))$

**Memory complexity:**  $O(\text{len}(\text{contour.vertices}))$

**Reference:** [https://en.wikipedia.org/wiki/Sweep\\_line\\_algorithm](https://en.wikipedia.org/wiki/Sweep_line_algorithm)

#### Parameters

- **contour** – contour to check.
- **context** – geometrical context.

**Returns** true if contour is self-intersecting, false otherwise.

---

**Note:** Consecutive equal vertices like `Point(2, 0)` in

```
Contour([Point(0, 0), Point(2, 0), Point(2, 0), Point(2, 2)])
```

will be considered as self-intersection, if you don't want them to be treated as such – filter out before passing as argument.

---

```
>>> from ground.base import get_context
>>> context = get_context()
>>> Contour, Point = context.contour_cls, context.point_cls
>>> contour_self_intersects(Contour([Point(0, 0), Point(2, 0),
...                               Point(2, 2)]))
False
>>> contour_self_intersects(Contour([Point(0, 0), Point(2, 0),
...                               Point(1, 0)]))
True
```

`bentley_ottmann.planar.segments_intersect` (*segments*: `Sequence[ground.hints.Segment]`, \*, *context*: `Optional[ground.base.Context]` = `None`) → `bool`

Checks if segments have at least one intersection.

Based on Shamos-Hoey algorithm.

**Time complexity:**  $O(\text{len}(\text{segments}) * \log \text{len}(\text{segments}))$

**Memory complexity:**  $O(\text{len}(\text{segments}))$

**Reference:** [https://en.wikipedia.org/wiki/Sweep\\_line\\_algorithm](https://en.wikipedia.org/wiki/Sweep_line_algorithm)

#### Parameters

- **segments** – sequence of segments.
- **context** – geometrical context.

**Returns** true if segments intersection found, false otherwise.

```
>>> from ground.base import get_context
>>> context = get_context()
>>> Point, Segment = context.point_cls, context.segment_cls
>>> segments_intersect([])
False
>>> segments_intersect([Segment(Point(0, 0), Point(2, 2))])
False
>>> segments_intersect([Segment(Point(0, 0), Point(2, 0)),
...                      Segment(Point(0, 2), Point(2, 2))])
False
>>> segments_intersect([Segment(Point(0, 0), Point(2, 2)),
...                      Segment(Point(0, 0), Point(2, 2))])
True
>>> segments_intersect([Segment(Point(0, 0), Point(2, 2)),
...                      Segment(Point(2, 0), Point(0, 2))])
True
```

bentley\_ottmann.planar.**segments\_cross\_or\_overlap**(*segments*:  
*sequence*[*ground.hints.Segment*],  
 \*, *context*:  
*optional*[*ground.base.Context*] =  
 None) → bool

Checks if at least one pair of segments crosses or overlaps.

Based on Bentley-Ottmann algorithm.

**Time complexity:**  $O(\text{len}(\text{segments}) * \log \text{len}(\text{segments}))$

**Memory complexity:**  $O(\text{len}(\text{segments}))$

**Reference:** [https://en.wikipedia.org/wiki/Bentley%E2%80%93Ottmann\\_algorithm](https://en.wikipedia.org/wiki/Bentley%E2%80%93Ottmann_algorithm)

#### Parameters

- **segments** – sequence of segments.
- **context** – geometrical context.

**Returns** true if segments overlap or cross found, false otherwise.

```
>>> from ground.base import get_context
>>> context = get_context()
>>> Point, Segment = context.point_cls, context.segment_cls
>>> segments_cross_or_overlap([])
False
>>> segments_cross_or_overlap([Segment(Point(0, 0), Point(2, 2))])
False
>>> segments_cross_or_overlap([Segment(Point(0, 0), Point(2, 0)),
...                      Segment(Point(0, 2), Point(2, 2))])
False
>>> segments_cross_or_overlap([Segment(Point(0, 0), Point(2, 2)),
...                      Segment(Point(0, 0), Point(2, 2))])
True
>>> segments_cross_or_overlap([Segment(Point(0, 0), Point(2, 2)),
...                      Segment(Point(0, 2), Point(2, 0))])
True
```

## PYTHON MODULE INDEX

### **b**

`bentley_ottmann.planar`, 1





## INDEX

### B

bentley\_ottmann.planar  
module, 1

### C

contour\_self\_intersects() (*in module bentley\_ottmann.planar*), 1

### M

module  
bentley\_ottmann.planar, 1

### S

segments\_cross\_or\_overlap() (*in module bentley\_ottmann.planar*), 2

segments\_intersect() (*in module bentley\_ottmann.planar*), 1