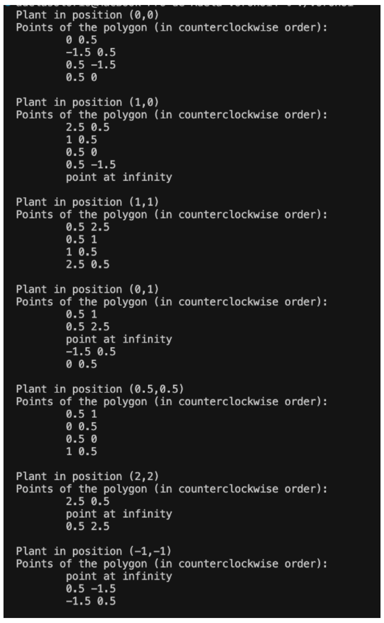
Voronoi test cases explanation

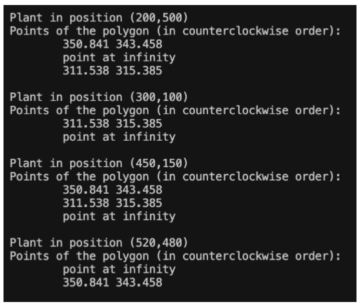
Test case 1:



```
Plant in position (0,0)
Points of the polygon (in counterclockwise order):
        0 0.5
        -1.5 0.5
        0.5 -1.5
        0.5 0

Plant in position (1,0)
Points of the polygon (in counterclockwise order):
        2.5 0.5
        1 0.5
        0.5 0
        0.5 -1.5
        point at infinity

Plant in position (1,1)
Points of the polygon (in counterclockwise order):
        0.5 2.5
        0.5 1
        1 0.5
        2.5 0.5

Plant in position (0,1)
Points of the polygon (in counterclockwise order):
        0.5 1
        0.5 2.5
        point at infinity
        -1.5 0.5
        0 0.5

Plant in position (0.5,0.5)
Points of the polygon (in counterclockwise order):
        0.5 1
        0 0.5
        0.5 0
        1 0.5

Plant in position (2,2)
Points of the polygon (in counterclockwise order):
        2.5 0.5
        point at infinity
        0.5 2.5

Plant in position (-1,-1)
Points of the polygon (in counterclockwise order):
        point at infinity
        0.5 -1.5
        -1.5 0.5
```

This case tests points forming a square with a central point inside and two far-off points. The algorithm correctly divides the space, creating a central region around (0.5, 0.5) and larger regions for the far-away points. It demonstrates the algorithm's ability to handle both close and distant points.

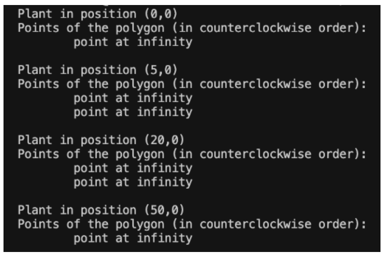
Test case 2:



```
Plant in position (200,500)
Points of the polygon (in counterclockwise order):
        350.841 343.458
        point at infinity
        311.538 315.385

Plant in position (300,100)
Points of the polygon (in counterclockwise order):
        311.538 315.385
        point at infinity

Plant in position (450,150)
Points of the polygon (in counterclockwise order):
        350.841 343.458
        311.538 315.385
        point at infinity

Plant in position (520,480)
Points of the polygon (in counterclockwise order):
        point at infinity
        350.841 343.458
```

This case test a set of scattered points that are not forming any regular shape. The algorithm handles widely separated points and correctly calculates the regions based on proximity. It demonstrates the ability to handle points that are spread across a larger area, with no immediate structure.

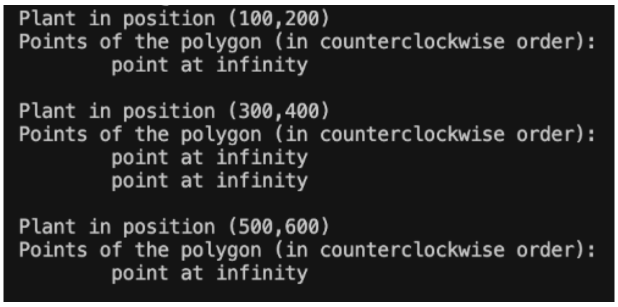
Test case 3:



```
Plant in position (0,0)
Points of the polygon (in counterclockwise order):
        point at infinity

Plant in position (5,0)
Points of the polygon (in counterclockwise order):
        point at infinity
        point at infinity

Plant in position (20,0)
Points of the polygon (in counterclockwise order):
        point at infinity
        point at infinity

Plant in position (50,0)
Points of the polygon (in counterclockwise order):
        point at infinity
```

This case tests collinear points on the x-axis. It shows that the algorithm cannot handle points that are perfectly aligned since they are partitioned in a 1D scenario (along a line).

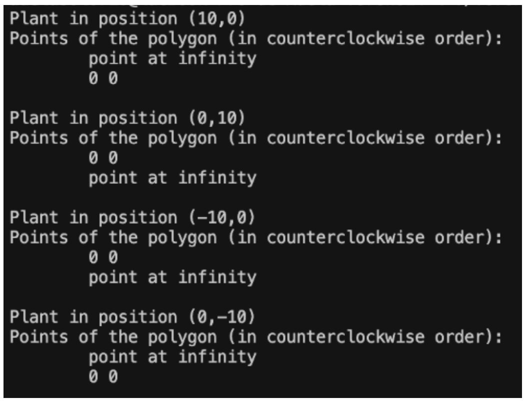
Test case 4:



```
Plant in position (100,200)
Points of the polygon (in counterclockwise order):
        point at infinity

Plant in position (300,400)
Points of the polygon (in counterclockwise order):
        point at infinity
        point at infinity

Plant in position (500,600)
Points of the polygon (in counterclockwise order):
        point at infinity
```

Similarly to the last one, this case tests a diagonal line, these type of cases generate cells that stretch infinitely in some directions, causing the result to be point at infinity.

Test case 5:



```
Plant in position (10,0)
Points of the polygon (in counterclockwise order):
        point at infinity
        0 0

Plant in position (0,10)
Points of the polygon (in counterclockwise order):
        0 0
        point at infinity

Plant in position (-10,0)
Points of the polygon (in counterclockwise order):
        0 0
        point at infinity

Plant in position (0,-10)
Points of the polygon (in counterclockwise order):
        point at infinity
        0 0
```

This case tests the behavior with points that are evenly distributed in a circular symmetry around the origin. It demonstrates the algorithm's ability to handle a scenario where the points are spaced evenly and symmetrically, ensuring it identifies the regions between these equidistant points.