Interior Grid Points - Solution

We use a formula known as Pick's formula and that describes the relation between the area of a polygon A, the number of grid points on the boundary B and the number of grid points inside the polygon I:

$$A = I + B/2 - 1. \tag{1}$$

We are interested in the number of grid points inside the polygon I, and therefore we need to determine **(1)** the number of points on the border of the polygon, and **(2)** the area of the polygon A.

(1)
The area of a polygon can be determined by the Shoelace formula, where (x_i, y_i) are the coordinates of the ith vertex:

$$2A = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + \cdots + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix} \tag{2}$$

```python
def calculate_area(vertices) -> float:
    """
    Calculates the area of the polygon.

    Returns:
        float: The area of the polygon.

    Raises:
        ValueError: If the polygon has less than 3 vertices.
    """
    if len(vertices) < 3:
        raise ValueError("A polygon must have at least 3 vertices.")

    area = 0.0

    for i in range(len(vertices)):
        x1, y1 = vertices[i]
        x2, y2 = vertices[(i + 1) % len(vertices)]
        area += (x1 * y2) - (x2 * y1)

    area = abs(area) / 2.0

    return area
```

(2)
To determine the number of vertices on the polygon's boundary, we can evaluate each edge of the polygon to check for the presence of grid points.

```python
def get_boundary_points(vertices: List[Tuple[int, int]]) -> List[Tuple[int, int]]:
    """
    Given a list of vertices that define the edges of a polygon, this function calculates and returns all the integer boundary points on the polygon's edges.

    Parameters:
    vertices (List[Tuple[int, int]]): A list of tuples representing the (x, y) coordinates
                                      of the vertices of the polygon.

```

```
    Returns:
    List[Tuple[int, int]]: A list of tuples representing the integer coordinates of
                           all the boundary points along the edges of the polygon.
    """
    boundary_points = []

    for i in range(-1, len(vertices) - 1):
        x1, y1 = vertices[i]
        x2, y2 = vertices[i+1]

        boundary_points.append(vertices[i])

        if x1 == x2:  # Vertical line
            y_values = range(min(y1, y2)+1, max(y1, y2))
            x_values = [x1] * len(y_values)
            boundary_points.extend(zip(x_values, y_values))

        else:  # Non-vertical line
            x_domain = range(min(x1, x2)+1, max(x1, x2))
            y_values = [y1 + (y2 - y1) * (x - x1) / (x2 - x1) for x in x_domain]
            y_values_int = [(x_domain[i], int(y_values[i])) for i in range(len(
                y_values)) if y_values[i].is_integer()]
            boundary_points.extend(y_values_int)

    return boundary_points
```

Combining with Pick's formula, we can determine the number of grid points inside the polygon as:

```
def get_points_inside(area: float, boundary_points: List[Tuple[int, int]]) ->
    float:
    """
    Calculates the number of interior lattice points inside a polygon using Pick's Theorem.

    Pick's Theorem states that for a simple polygon whose vertices are points on a lattice
    (i.e., points with integer coordinates), the number of interior lattice points (I) is
    related to the polygon's area (A) and the number of lattice points on its boundary (B) by
    the formula:

        I = A - B/2 + 1

    Args:
        area (float): The area of the polygon.
        boundary_points (List[Tuple[int, int]]): A list of tuples representing the integer coordinates
                                                 of the lattice points on the polygon's boundary.

    Returns:
        float: The number of interior lattice points inside the polygon.
    """
    return area - len(boundary_points) / 2 + 1
```