# 4.13.24

EE25BTECH11019 – Darji Vivek M.

# Question

**Question:**
The number of points, having both coordinates as integers, that lie in the interior of the triangle with vertices

$$(0,0), \quad (0,41), \quad (41,0)$$

is?

1. 820
2. 780
3. 901
4. 861

# Solution:Represent vertices and Area using determinant

$$\mathbf{A} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} 0 \\ 41 \end{pmatrix}, \qquad \mathbf{C} = \begin{pmatrix} 41 \\ 0 \end{pmatrix} \tag{1}$$

$$A = \frac{1}{2} \left\| \begin{vmatrix} 0 & 41 \\ 41 & 0 \end{vmatrix} \right\| \tag{2}$$

$$A = \frac{1}{2} \left| 0 - (41)(41) \right| \tag{3}$$

$$A = \frac{1681}{2} \tag{4}$$

# Solution:Boundary lattice points formula

For a line joining integer points $(x_1, y_1)$ and $(x_2, y_2)$, the number of lattice points on it is

$$N = \gcd(|x_2 - x_1|, |y_2 - y_1|) + 1$$

**Compute for each side:**

$$B_1 = \gcd(|0 - 0|, |41 - 0|) + 1 = 42 \tag{5}$$
$$B_2 = \gcd(|41 - 0|, |0 - 0|) + 1 = 42 \tag{6}$$
$$B_3 = \gcd(|41 - 0|, |0 - 41|) + 1 = 42 \tag{7}$$

Each vertex is counted twice, so

$$B = 42 + 42 + 42 - 3 = 123$$

# Solution:Apply Pick's Theorem

Pick's theorem:

$$A = I + \frac{B}{2} - 1$$

Rearranging for interior points $I$:

$$I = A - \frac{B}{2} + 1$$

# Solution:Substitute values and Answer

$$I = \frac{1681}{2} - \frac{123}{2} + 1 \tag{8}$$

$$I = \frac{1558}{2} + 1 \tag{9}$$

$$I = 780 \tag{10}$$

The number of integer lattice points lying strictly inside the triangle is

$$\boxed{780}$$

# C Code

```
#include <stdio.h>
#include <stdlib.h>

// Function to find GCD (used for boundary lattice
    points)
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

// Function to find number of boundary points on one
    edge
int boundary_points(int x1, int y1, int x2, int y2) {
    return gcd(abs(x2 - x1), abs(y2 - y1)) + 1;
}

// Function to find total interior lattice points
    using Pick's theorem
```

# C Code

```
int interior_points(int x1, int y1, int x2, int y2,
    int x3, int y3) {
    // Area = 1/2 * |x1(y2 - y3) + x2(y3 - y1) + x3(y1
        - y2)|
    float area = 0.5 * (float)abs(x1*(y2 - y3) + x2*(
        y3 - y1) + x3*(y1 - y2));
    // Boundary points on all three sides
    int b1 = boundary_points(x1, y1, x2, y2);
    int b2 = boundary_points(x2, y2, x3, y3);
    int b3 = boundary_points(x3, y3, x1, y1);
    // Subtract 3 since each vertex is counted twice
    int B = b1 + b2 + b3 - 3;
    // Pick's theorem: A = I + B/2 - 1  =>  I = A - B
        /2 + 1
    int I = (int)(area - (float)B/2 + 1);

    return I;
}
```

# Python (Call)

```
import ctypes
import numpy as np
import matplotlib.pyplot as plt
# Load the compiled C library
lib = ctypes.CDLL('./9.so')
# Define argument and return types
lib.interior_points.argtypes = [ctypes.c_int, ctypes.
    c_int, ctypes.c_int, ctypes.c_int, ctypes.c_int,
    ctypes.c_int]
lib.interior_points.restype = ctypes.c_int
# Triangle vertices
x1, y1 = 0, 0
x2, y2 = 0, 41
x3, y3 = 41, 0
# Call C function to get interior lattice points
I = lib.interior_points(x1, y1, x2, y2, x3, y3)
print("Number of interior lattice points:", I)
```

# Python (Call)

```
# ---- Generate interior points ----
points = []
for i in range(42):
    for j in range(42):
        if i > 0 and j > 0 and i + j < 41:
            points.append((i, j))
pts = np.array(points)
x, y = pts[:, 0], pts[:, 1]

# ---- Plotting ----
plt.figure(figsize=(6,6))
plt.plot([x1, x2, x3, x1], [y1, y2, y3, y1], 'k-',
    label="Triangle")
plt.scatter(x, y, s=10, color='red', label="Interior
    Lattice Points")
```

# Python (Call)

```
# Label the vertices with coordinates
plt.text(x1, y1, f"({x1},{y1})", fontsize=10,
    verticalalignment='bottom', horizontalalignment='
    right')
plt.text(x2, y2, f"({x2},{y2})", fontsize=10,
    verticalalignment='bottom', horizontalalignment='
    left')
plt.text(x3, y3, f"({x3},{y3})", fontsize=10,
    verticalalignment='top', horizontalalignment='right
    ')

plt.title("Integer Lattice Points Inside Triangle")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.axis('equal')
plt.grid(True)
plt.show()
```

# Python Output and Plot