1.5.28

Naman kumar-EE25BTECH11041

August 26,2025

Question

P$(5, -3)$ and **Q** $(3,\ y)$ are the points of trisection of the line segment joining **A**$(7, -2)$ and **B**$(1, -5)$. Then y equals.

Equation Used

$$\mathbf{Q} = \frac{1}{1+k}\left(\mathbf{A} + k\mathbf{B}\right) \tag{1}$$

Theoratical Solution

$$\mathbf{Q} = \frac{1}{1+2}\left(\begin{pmatrix}7\\-2\end{pmatrix} + 2\begin{pmatrix}1\\-5\end{pmatrix}\right) \tag{2}$$

$$\mathbf{Q} = \frac{1}{1+2}\left(\begin{pmatrix}9\\-12\end{pmatrix}\right) \tag{3}$$

$$\mathbf{Q} = \begin{pmatrix}3\\-4\end{pmatrix} \tag{4}$$

$$\mathbf{Q} = \begin{pmatrix}3\\y\end{pmatrix} = \begin{pmatrix}3\\-4\end{pmatrix} \tag{5}$$

Therefore, value of y $= 4$

C Code - Section formula function

```c
#include <stdio.h>

void trisec(double k, double x1, double y1, double x2, double y2,
     double* a, double* b){
    *a= (x1+k*x2)/(1+k);
    *b= (y1+k*y2)/(1+k);
}
```

Python Code through shared output

```
import ctypes
import numpy as np
import matplotlib.pyplot as plt

# --- Ctypes Setup ---

# Load the shared library.
# Make sure 'main.so' is in the same directory as this Python
    script,
# or provide the full path to it.
try:
    c_lib = ctypes.CDLL('./main.so')
except OSError as e:
    print(f"Error loading shared library: {e}")
    print("Please ensure 'main.so' is in the same directory as
        this script.")
    exit()

# Define the argument types for the C function.
```

Python Code through shared output

```
# The C function signature is:
# void trisec(double x1, double y1, double x2, double y2, double*
    a, double* b, double* c, double* d)
c_lib.trisec.argtypes = [
    ctypes.c_double,
    ctypes.c_double,
    ctypes.c_double,
    ctypes.c_double,
    ctypes.c_double,
    ctypes.POINTER(ctypes.c_double),
    ctypes.POINTER(ctypes.c_double)
]

# Define the return type of the function.
c_lib.trisec.restype = None

# --- Calculation ---
```

Python Code through shared output

```
# Define the input coordinates for the two endpoints of the line
    segment

k = 2
x1, y1 = 7.0, -2.0
x2, y2 = 1.0, -5.0

# Prepare ctypes variables to hold the results.
# These will act as the pointers that the C function will write
    to.
ta = ctypes.c_double()
tb = ctypes.c_double()

# Call the C function from Python to calculate the trisection
    point
c_lib.trisec(k, x1, y1, x2, y2, ctypes.byref(ta), ctypes.byref(tb
    ))
```

Python Code through shared output

```
# Extract the float values from the ctypes variables
ta_val, tb_val = ta.value, tb.value
print(f"Line segment from ({x1}, {y1}) to ({x2}, {y2})")
print(f"Trisection point 1 calculated by C code: ({ta_val:.2f}, {
    tb_val:.2f})")

# --- Plotting ---

# Create the plot
plt.figure(figsize=(8, 6))

# Plot the full line segment
plt.plot([x1, x2], [y1, y2], 'g--', label="Line Segment")

# Plot the endpoints of the line
plt.scatter([x1, x2], [y1, y2], color="red", s=100, zorder=5,
    label="Endpoints")
```

Python Code through shared output

```
plt.text(x1, y1 - 0.5, f"A ({x1:.1f}, {y1:.1f})", color="red",
    fontsize=10)
plt.text(x2, y2 - 0.5, f"B ({x2:.1f}, {y2:.1f})", color="red",
    fontsize=10)

# Plot the calculated trisection point
plt.scatter(ta_val, tb_val, color="blue", marker="X", s=150,
    zorder=5, label="Trisection Point")
plt.text(ta_val, tb_val + 0.3, f"Trisection Pt 1\n({ta_val:.2f},
    {tb_val:.2f})", color="blue", fontsize=10)

# Configure plot appearance
plt.title("Line Segment and its Trisection Point")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend(loc="upper left")
plt.grid(True)
plt.axis("equal") # Ensures the scaling is the same on both axes
plt.show()
```

Python Code: Direct

```
import matplotlib.pyplot as plt
import numpy as np

# --- Function Definition (similar to the C code) ---
# This function calculates the coordinates (a, b) of a point that
    divides
# the line segment from (x1, y1) to (x2, y2) in the ratio k:1.
# This is derived from the section formula: (n*x1 + m*x2)/(m+n)
# by setting the ratio as k = m/n.
def trisec(k, x1, y1, x2, y2):
    """
    Calculates the coordinates of a point dividing a line segment
        .

    Args:
        k (float): The ratio m/n for the section formula.
        x1, y1 (float): Coordinates of the first point (A).
        x2, y2 (float): Coordinates of the second point (B).
```

Python Code: Direct

```
    Returns:
        tuple: A tuple containing the (x, y) coordinates of the
            dividing point.
    """
    a = (x1 + k * x2) / (1 + k)
    b = (y1 + k * y2) / (1 + k)
    return (a, b)

# --- Problem Setup ---
# Given points for the line segment
A = (7, -2)
B = (1, -5)

# --- Calculations ---
# The points of trisection divide the segment in ratios 1:2 and
    2:1.

# 1. Calculate Point P (divides AB in ratio 1:2)
```

Python Code: Direct

```
# Here, m=1, n=2, so k = m/n = 1/2 = 0.5
k_p = 0.5
P_calculated = trisec(k_p, A[0], A[1], B[0], B[1])

# 2. Calculate Point Q (divides AB in ratio 2:1)
# Here, m=2, n=1, so k = m/n = 2/1 = 2.0
k_q = 2.0
Q_calculated = trisec(k_q, A[0], A[1], B[0], B[1])

y_solution = Q_calculated[1]

# --- Output the Results to Console ---
print("--- Trisection Calculation ---")
print(f"Point A: {A}")
print(f"Point B: {B}\n")

print(f"Calculated coordinates for P (ratio 1:2, k={k_p}): {
    P_calculated}")
```

Python Code: Direct

```
print(f"Calculated coordinates for Q (ratio 2:1, k={k_q}): {
    Q_calculated}\n")

print("--- Solution ---")
print(f"The problem states Q is (3, y). Our calculation gives Q
    as {Q_calculated}.")
print(f"Therefore, the value of y is {int(y_solution)}.")
plt.figure(figsize=(10, 8))
ax = plt.gca()

plt.plot([A[0], B[0]], [A[1], B[1]], 'b-', label='Line Segment AB
    ', zorder=1)
points = {'A': A, 'B': B, 'P': P_calculated, 'Q': Q_calculated}
```

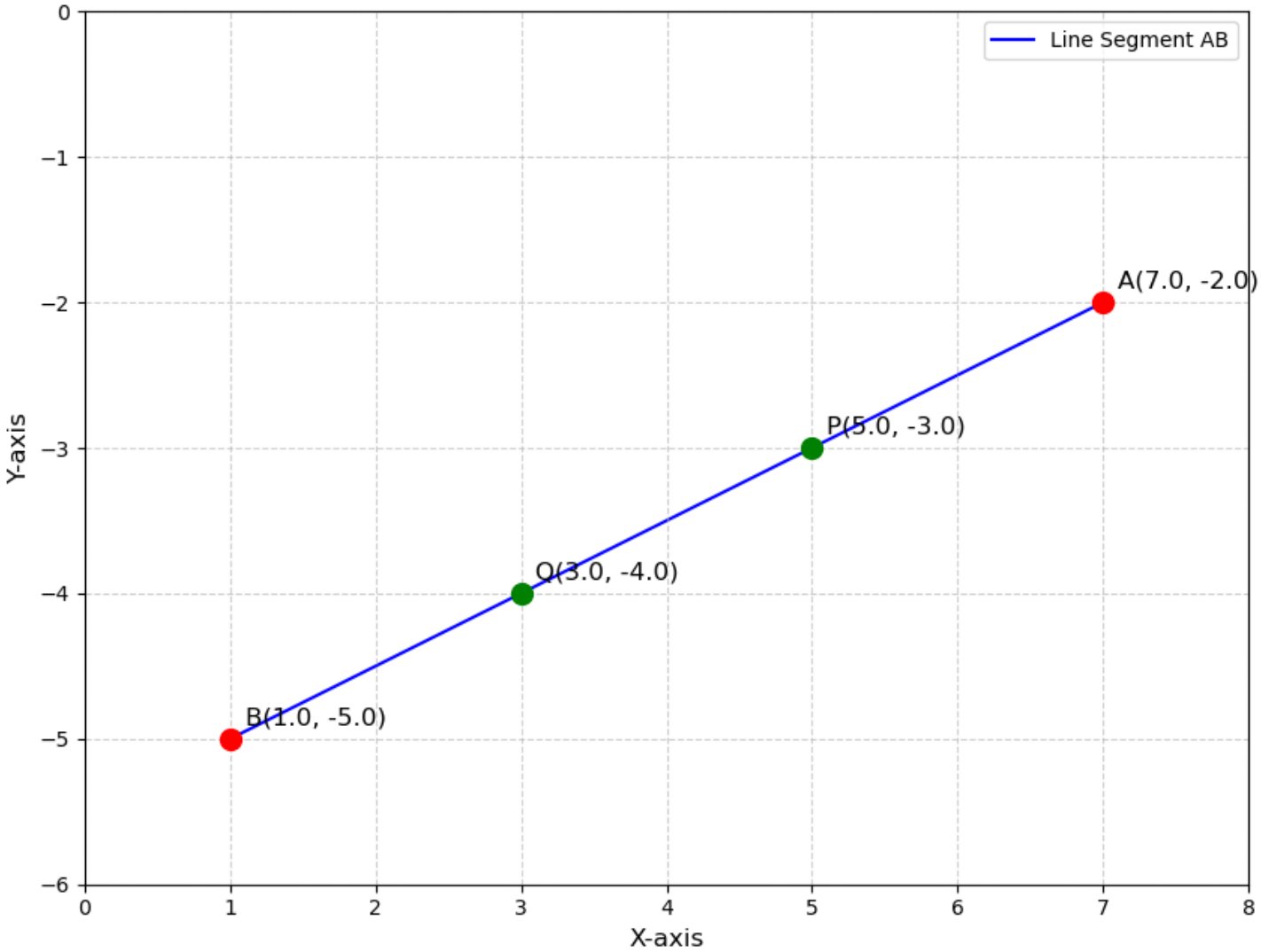
Python Code: Direct

```
colors = {'A': 'red', 'B': 'red', 'P': 'green', 'Q': 'green'}
for name, (px, py) in points.items():
    plt.scatter(px, py, color=colors[name], s=100, zorder=2)
    plt.text(px + 0.1, py + 0.1, f'{name}({px:.1f}, {py:.1f})',
        fontsize=12)

plt.xlabel('X-axis', fontsize=12)
plt.ylabel('Y-axis', fontsize=12)
ax.set_aspect('equal', adjustable='box')
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.xlim(0, 8)
plt.ylim(-6, 0)

plt.savefig('trisection_diagram.png')
print("\nDiagram saved as 'trisection_diagram.png'")
plt.show()
```

Graph


Line Segment AB
A(7.0, -2.0)
P(5.0, -3.0)
Q(3.0, -4.0)
B(1.0, -5.0)
Y-axis
X-axis
0
−1
−2
−3
−4
−5
−6
0
1
2
3
4
5
6
7
8