3.2.19

AI25BTECH11003 - Bhavesh Gaikwad

September 2,2025

Question

Two sides of a triangle are of lengths 5cm and 1.5cm. The length of the third side of the triangle cannot be
a) 3.6 cm
b) 4.1 cm
c) 3.8 cm
d) 3.4 cm

Theoretical Solution

Let a=5 cm, b=1.5 cm, and c be the third side. For each option we test:

$$\begin{aligned} &1. \quad a+b>c, && (1)\\ &2. \quad a+c>b, && (2)\\ &3. \quad b+c>a. && (3)\end{aligned}$$

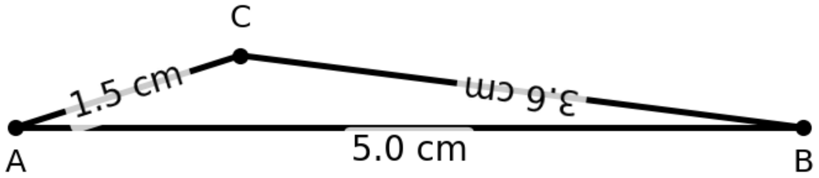
If all three hold, the triangle exists; otherwise it does not.

Option (A): c = 3.6 cm

$$\begin{aligned} 5+1.5>3.6 &\quad\Rightarrow\quad 6.5>3.6 \quad \checkmark, && (4)\\ 5+3.6>1.5 &\quad\Rightarrow\quad 8.6>1.5 \quad \checkmark, && (5)\\ 1.5+3.6>5 &\quad\Rightarrow\quad 5.1>5 \quad \checkmark. && (6)\end{aligned}$$

All conditions satisfied $\Rightarrow$ triangle exists.

Theoretical Solution



Theoretical Solution

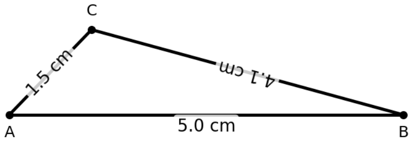
Option (B): $c = 4.1$ cm

$$5 + 1.5 > 4.1 \quad \Rightarrow \quad 6.5 > 4.1 \quad \checkmark, \tag{7}$$

$$5 + 4.1 > 1.5 \quad \Rightarrow \quad 9.1 > 1.5 \quad \checkmark, \tag{8}$$

$$1.5 + 4.1 > 5 \quad \Rightarrow \quad 5.6 > 5 \quad \checkmark. \tag{9}$$

All conditions satisfied $\Rightarrow$ triangle exists.



Theoretical Solution

Option (C): c = 3.8 cm

$$5 + 1.5 > 3.8 \quad \Rightarrow \quad 6.5 > 3.8 \quad \checkmark, \tag{10}$$
$$5 + 3.8 > 1.5 \quad \Rightarrow \quad 8.8 > 1.5 \quad \checkmark, \tag{11}$$
$$1.5 + 3.8 > 5 \quad \Rightarrow \quad 5.3 > 5 \quad \checkmark. \tag{12}$$

All conditions satisfied $\Rightarrow$ triangle exists.

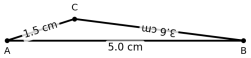


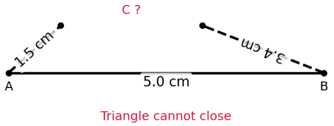
Figure: Triangle

Theoretical Solution

Option (D): c = 3.4 cm

$$5 + 1.5 > 3.4 \quad \Rightarrow \quad 6.5 > 3.4 \quad \checkmark, \tag{13}$$
$$5 + 3.4 > 1.5 \quad \Rightarrow \quad 8.4 > 1.5 \quad \checkmark, \tag{14}$$
$$1.5 + 3.4 > 5 \quad \Rightarrow \quad 4.9 > 5 \quad \times\, . \tag{15}$$

Condition 3 fails $\Rightarrow$ triangle does *not* exist.



Python Code

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

def draw_segment_with_label(P, Q, label, style='k-', text_offset
    =(0.0, 0.0)):
    """Draw segment PQ and place a rotated length label near its
        midpoint."""
    P = np.array(P, dtype=float)
    Q = np.array(Q, dtype=float)
    plt.plot([P[0], Q[0]], [P[1], Q[1]], style, linewidth=2)

    mid = (P + Q) / 2.0
    dx, dy = Q - P
    angle_deg = np.degrees(np.arctan2(dy, dx))
    plt.text(
        mid[0] + text_offset[0],
        mid[1] + text_offset[1],
        label,
```

Python Code

```
         ha='center',
        va='center',
        rotation=angle_deg,
        rotation_mode='anchor',
        fontsize=11,
        bbox=dict(boxstyle='round,pad=0.2', fc='white', ec='none'
            , alpha=0.8)
    )

def plot_triangle_with_lengths(a, b, c, filename, incomplete=
    False):
    """
    Draw triangle with sides: AB=a, AC=b, BC=c.
    If incomplete=True, depict an impossible case by showing
        dashed partial sides from A and B.
    """
```

Python Code

```
# Fix base AB on x-axis
  A = np.array([0.0, 0.0])
  B = np.array([a, 0.0])

  plt.figure(figsize=(4.2, 4.0))

  # Always draw AB
  draw_segment_with_label(A, B, f"{a:.1f} cm", style='k-',
      text_offset=(0, -0.15))

  if not incomplete:
      # Compute C using law of cosines around A
      # cos(theta) = (a^2 + b^2 - c^2) / (2ab)
      cos_theta = (a*a + b*b - c*c) / (2.0*a*b)
      cos_theta = np.clip(cos_theta, -1.0, 1.0)
      theta = np.arccos(cos_theta)
      C = np.array([b*np.cos(theta), b*np.sin(theta)])
```

Python Code

```
# Draw AC and BC
draw_segment_with_label(A, C, f"{b:.1f} cm", style='k-')
draw_segment_with_label(B, C, f"{c:.1f} cm", style='k-')

# Mark points
plt.scatter([A[0], B[0], C[0]], [A[1], B[1], C[1]], c='k'
    , s=18)
plt.text(A[0], A[1]-0.28, 'A', ha='center')
plt.text(B[0], B[1]-0.28, 'B', ha='center')
plt.text(C[0], C[1]+0.18, 'C', ha='center')
```

Python Code

```
    else:
        # Depict an impossible triangle: show partial (dashed)
            edges toward a "would-be" C
        # Choose a visual point above AB just for indicating
            direction
        Cg = np.array([a*0.30, b*0.90]) # purely for illustration

        # Partial dashed from A toward Cg
        A_dash = A + 0.55*(Cg - A)
        draw_segment_with_label(A, A_dash, f"{b:.1f} cm", style='
            k--')

        # Partial dashed from B toward Cg
        B_dash = B + 0.55*(Cg - B)
        draw_segment_with_label(B, B_dash, f"{c:.1f} cm", style='
            k--')
```

Python Code

```
# Mark points
plt.scatter([A[0], B[0], A_dash[0], B_dash[0]], [A[1], B
    [1], A_dash[1], B_dash[1]], c='k', s=18)
plt.text(A[0], A[1]-0.28, 'A', ha='center')
plt.text(B[0], B[1]-0.28, 'B', ha='center')
plt.text((A_dash[0]+B_dash[0])/2, (A_dash[1]+B_dash[1])/2
     + 0.18, 'C ?', ha='center', color='crimson')

# Add a note for impossibility
plt.text(a*0.5, -0.75, 'Triangle cannot close', ha='
    center', color='crimson', fontsize=10)
```

Python Code

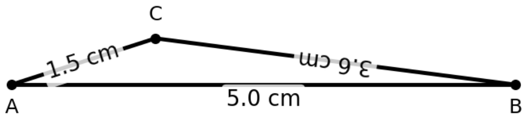
```
  # Final touches
    plt.axis('equal')
    plt.axis('off')
    plt.tight_layout()
    plt.savefig(filename, dpi=180, bbox_inches='tight')
    plt.close()

# Given sides
a = 5.0 # AB
b = 1.5 # AC

# Options for BC
options = [3.6, 4.1, 3.8, 3.4]

# Generate four figures with side-length labels
for i, c in enumerate(options, start=1):
    out = f"fig{i}.png"
    plot_triangle_with_lengths(a, b, c, out, incomplete=(i == 4))
```

Vector Representation


C
1.5 cm
3.6 cm
A
5.0 cm
B