📚 Module 12: Proportional Reasoning - Complete Notes

🎯 What You'll Learn

In this module, you'll master **direct and inverse proportion** — understanding how quantities scale together or against each other.

📖 Concept Explained (Like a YouTube Video)

The Basics

Direct Proportion

When one increases, the other increases at the same rate.

```
More workers → More work done
Double input → Double output

y = k × x (k is the constant)
```

Inverse Proportion

When one increases, the other decreases.

```
More workers → Less time needed
Double speed → Half the time

y = k / x (k is the constant)
```

Visual

```
DIRECT: x increases, y increases
        ↗
       ↗
      ↗
     ↗
    x

INVERSE: x increases, y decreases
     \
      \
       \
        \
         x
```

💻 Programming Connection

Code Examples

```python
# Example 1: Direct Proportion

def calculate_direct(known_x, known_y, new_x):
    """If y is directly proportional to x, find new y"""
    k = known_y / known_x  # constant of proportionality
    return k * new_x

# 5 items cost $20, how much for 8 items?
print(calculate_direct(5, 20, 8))   # 32.0

# 3 workers produce 150 units, how many for 7 workers?
print(calculate_direct(3, 150, 7))  # 350.0
```

```python
# Example 2: Inverse Proportion

def calculate_inverse(known_x, known_y, new_x):
    """If y is inversely proportional to x, find new y"""
    k = known_x * known_y  # constant
    return k / new_x

# 4 workers take 6 days, how long for 8 workers?
print(calculate_inverse(4, 6, 8))   # 3.0 days

# 3 pipes fill tank in 12 hours, how long for 6 pipes?
print(calculate_inverse(3, 12, 6))  # 6.0 hours
```

```python
# Example 3: Scale Factor

def scale_dimensions(width, height, factor):
    """Scale dimensions by factor"""
    return (width * factor, height * factor)

def scale_to_width(orig_w, orig_h, new_w):
    """Scale proportionally to target width"""
    factor = new_w / orig_w
    return (new_w, orig_h * factor)

# Scale 800x600 to width 400 (maintain aspect ratio)
print(scale_to_width(800, 600, 400))  # (400, 300.0)
```

```python
# Example 4: Estimate Using Proportion

def estimate_time(known_items, known_time, target_items):
    """Estimate time for different item count (direct proportion)"""
    return (known_time / known_items) * target_items

# 50 tests take 10 minutes, estimate for 200 tests
print(estimate_time(50, 10, 200))  # 40.0 minutes
```

🧪 SDET/Testing Application

```python
# SDET Scenario: Estimate Test Execution Time

def estimate_test_duration(benchmark_tests, benchmark_time, actual_tests):
    """Estimate total duration based on benchmark"""
    # Direct proportion: more tests = more time
    estimated = calculate_direct(benchmark_tests, benchmark_time, actual_tests)

    return {
        "benchmark": f"{benchmark_tests} tests in {benchmark_time} min",
        "actual_tests": actual_tests,
        "estimated_time": f"{estimated:.1f} min",
        "per_test": f"{benchmark_time/benchmark_tests:.2f} min"
    }

result = estimate_test_duration(50, 10, 175)
print(f"Estimated: {result['estimated_time']}")  # 35.0 min
```

```python
# SDET Scenario: Parallel Execution Time

def parallel_execution_time(tests, single_runner_time, num_runners):
    """Calculate time with parallel runners (inverse proportion)"""
    # Inverse: more runners = less time
    parallel_time = single_runner_time / num_runners

    return {
        "total_tests": tests,
        "single_runner": f"{single_runner_time} min",
        "num_runners": num_runners,
        "parallel_time": f"{parallel_time:.1f} min",
        "speedup": f"{num_runners}x faster"
    }

result = parallel_execution_time(100, 60, 4)
print(f"With 4 runners: {result['parallel_time']}")  # 15.0 min
```

🎓 Practice Problems

Problem 1: Easy 🟢

Challenge: A car travels 150km in 2 hours. How far in 5 hours?

Problem 2: Medium 🟡

Challenge: 6 painters finish a job in 4 days. With 8 painters, how many days?

Problem 3: Application 🔴

Scenario: Your single test runner takes 120 minutes. Budget is 30 minutes.

Challenge: How many parallel runners do you need?

🔑 Key Takeaways

✅ **Direct**: More input → More output (multiply)

✅ **Inverse**: More input → Less of other (divide)

✅ **Scale factor**: Maintain proportions while resizing

✅ **Cross-multiply**: a/b = c/d → a×d = b×c

💾 *Save as: `Module_12_Proportional_Reasoning.md`*