12.546

BEERAM MADHURI - EE25BTECH11012

October 2025

Question

Consider the following two statements

P: $\begin{pmatrix} 0 & 5 \\ 0 & 7 \end{pmatrix}$ has infinitely many LU factorizations, where $\mathbf{L}$ is lower triangular with each diagonal entry 1 and $\mathbf{U}$ is upper triangular.

Q: $\begin{pmatrix} 0 & 0 \\ 2 & 5 \end{pmatrix}$ has no LU factorization, where $\mathbf{L}$ is lower triangular with each diagonal entry 1 and $\mathbf{U}$ is upper triangular.

Then which one of the following options is correct? (MA 2018)

a) P is TRUE and Q is FALSE
b) Both P and Q are TRUE
c) P is FALSE and Q is TRUE
d) Both P and Q are FALSE

given data

statement	given matrix
P	$\begin{pmatrix} 0 & 5 \\ 0 & 7 \end{pmatrix}$
Q	$\begin{pmatrix} 0 & 0 \\ 2 & 5 \end{pmatrix}$

Table: Variables used

solution

Let

$$L = \begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix} \tag{1}$$

$$U = \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} \tag{2}$$

$$LU = \begin{pmatrix} u_{11} & u_{12} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} \end{pmatrix} \tag{3}$$

Statement P:

$$LU = \begin{pmatrix} 0 & 5 \\ 0 & 7 \end{pmatrix} \tag{4}$$

$$u_{11} = 0, u_{12} = 5 \tag{5}$$

$$l_{21} u_{12} + u_{22} = 7 \tag{6}$$

has infinite solutions.
$\therefore$ there are infinitely many pairs (l_{21}, u_{22})
Statement P is true

Statement Q:

$$LU = \begin{pmatrix} 0 & 0 \\ 2 & 5 \end{pmatrix} \tag{7}$$

$$u_{11} = 0 \tag{8}$$

$$l_21u_11 = 0 \tag{9}$$

$$\text{but} l_21u_11 = 2 \tag{10}$$

$\therefore$ no pairs(L,U) exists.
$\therefore$ Statement Q is true.
Option b is correct.

Python Code

```python
def print_matrix(name: str, matrix: list[list[float]]):
    """Prints a 2x2 matrix with a given name in a formatted way."""
    print(f"Matrix {name}:")
    for row in matrix:
        print(" [ ", end="")
        for val in row:
            # Format to 2 decimal places with a width of 5 characters
            print(f"{val:5.2f} ", end="")
        print("]")
    print()
```

```python
def analyze_lu_factorization(matrix_name: str, A: list[list[float
    ]]):
    """
    Analyzes the LU factorization for a 2x2 matrix A.

    It checks the conditions derived from A = LU, where L is
        lower
    triangular with 1s on the diagonal and U is upper triangular.
```

Python Code

```
"""
print(f"--- Analyzing Matrix {matrix_name} ---")
print_matrix(matrix_name, A)

# For a matrix A = [[a, b], [c, d]], we want to find L and U
    such that A = LU:
# L = [[1, 0], [l21, 1]] and U = [[u11, u12], [0, u22]]
#
# Multiplying L and U and equating to A gives the system of
    equations:
```

Python Code

```
# 1) u11 = A[0][0]
# 2) u12 = A[0][1]
# 3) l21 * u11 = A[1][0] <-- This is the critical equation.
# 4) l21 * u12 + u22 = A[1][1]
u11 = A[0][0]
a10 = A[1][0] # The element at row 1, column 0
print("From the definition, the critical equation is: l21 *
    u11 = A[1][0]")
```

Python Code

```python
    print(f"Substituting known values from matrix {matrix_name}:")
    print(f" -> l21 * {u11:.2f} = {a10:.2f}\n")

    # Check the condition of the critical equation (3)
    if u11 == 0:
        if a10 == 0:
            # This is the case for Matrix P: l21 * 0 = 0
            print(f"Result for {matrix_name}:")
            print("The equation becomes 0 = 0, which is always true.")
```

Python Code

```
            print("This means 'l21' can be any real number,
                leading to infinitely many solutions.")
            print(f"Conclusion: Statement {matrix_name} is TRUE.")
        else:
            # This is the case for Matrix Q: l21 * 0 = 2
            print(f"Result for {matrix_name}:")
            print(f"The equation becomes 0 = {a10:.2f}, which is a
                 contradiction.")
            print("No value of 'l21' can satisfy this, so no LU
                factorization exists.")
            print(f"Conclusion: Statement {matrix_name} is TRUE.")
```

Python Code

```
    else:
        # This is the standard case where a unique solution would
            exist
        print(f"Result for {matrix_name}:")
        print(f"Since u11 ({u11:.2f}) is non-zero, a unique
            solution for l21 could be found.")
        print(f"This case does not apply to matrices P or Q.")
# This block ensures the code runs only when the script is
    executed directly
```

Python Code

```python
if _name_ == "_main_":
    # Matrix from Statement P
    P = [
        [0.0, 5.0],
        [0.0, 7.0]
    ]

    # Matrix from Statement Q
    Q = [
        [0.0, 0.0],
        [2.0, 5.0]
    ]
```

Python Code

```
print("This program analyzes the LU factorization for the two
    given matrices.\n")
# Analyze Matrix P
analyze_lu_factorization("P", P)
# Analyze Matrix Q
analyze_lu_factorization("Q", Q)
print("Final Conclusion:")
print("Both statements P and Q are TRUE. The correct option
    is (b).")
```

C Code

```c
#include <stdio.h>

/**
 * @file compile_time_lu.c
 * @brief Analyzes LU factorization at compile time using the
     preprocessor.
 *
 * The logic to determine the truthiness of statements P and Q is
      resolved
 * before the program is compiled. The runtime executable only
     contains the
 * final, pre-determined answer.
 */
```

C Code

```c
// --- Matrix P Definition ---
// A = [[P_00, P_01], [P_10, P_11]]
#define P_00 0
#define P_01 5
#define P_10 0
#define P_11 7
// --- Matrix Q Definition ---
// A = [[Q_00, Q_01], [Q_10, Q_11]]
#define Q_00 0
#define Q_01 0
#define Q_10 2
#define Q_11 5
```

C Code

```c
// --- COMPILE-TIME ANALYSIS ---
// The preprocessor will evaluate these #if statements.
// Analyze Statement P: "P has infinitely many LU factorizations"
// This is TRUE if A[0][0] is 0 AND A[1][0] is 0, leading to 0 =
    0.
#if P_00 == 0 && P_10 == 0
  #define P_IS_TRUE 1
#else
  #define P_IS_TRUE 0
#endif
```

C Code

```
// Analyze Statement Q: "Q has no LU factorization"
// This is TRUE if A[0][0] is 0 AND A[1][0] is not 0, leading to
    0 = non-zero.
#if Q_00 == 0 && Q_10 != 0
  #define Q_IS_TRUE 1
#else
  #define Q_IS_TRUE 0
#endif
```

C Code

```c
int main() {
    printf("This conclusion was determined entirely at COMPILE
        TIME.\n");
    printf("The running program is just printing the pre-
        calculated result.\n\n");

    // The preprocessor uses the P_IS_TRUE and Q_IS_TRUE macros
    // to select ONLY ONE of the following printf statements to
    // include in the final compiled program.
```

C Code

```c
#if P_IS_TRUE && Q_IS_TRUE
  printf("Conclusion: Both P and Q are TRUE. The correct
      option is (b).\n");
#elif P_IS_TRUE && !Q_IS_TRUE
  printf("Conclusion: P is TRUE and Q is FALSE. The correct
      option is (a).\n");
```

C Code

```
    #elif !P_IS_TRUE && Q_IS_TRUE
      printf("Conclusion: P is FALSE and Q is TRUE. The correct
           option is (c).\n");
    #else
      printf("Conclusion: Both P and Q are FALSE. The correct
           option is (d).\n");
    #endif
    return 0;
}
```

```
from ctypes import c_int
"""
Simulating C preprocessor-based compile-time LU factorization
    analysis
in Python using ctypes and top-level evaluation.
"""
# --- Matrix P Definition ---
P_00 = c_int(0)
P_01 = c_int(5)
P_10 = c_int(0)
P_11 = c_int(7)
```

Python and C Code

```python
# --- Matrix Q Definition ---
Q_00 = c_int(0)
Q_01 = c_int(0)
Q_10 = c_int(2)
Q_11 = c_int(5)
# --- "Compile-time" evaluation simulated at import time ---
# Analyze Statement P: "P has infinitely many LU factorizations"
```

Python and C Code

```
# TRUE if A[0][0] == 0 and A[1][0] == 0
if P_00.value == 0 and P_10.value == 0:
    P_IS_TRUE = True
else:
    P_IS_TRUE = False

# Analyze Statement Q: "Q has no LU factorization"
# TRUE if A[0][0] == 0 and A[1][0] != 0
if Q_00.value == 0 and Q_10.value != 0:
    Q_IS_TRUE = True
```

Python and C Code

```python
else:
    Q_IS_TRUE = False


def main():
    print("This conclusion was determined entirely at 'import
        time' (simulating compile time).")
    print("The running program is just printing the pre-
        determined result.\n")
```

Python and C Code

```
if P_IS_TRUE and Q_IS_TRUE:
    print("Conclusion: Both P and Q are TRUE. The correct
        option is (b).")
elif P_IS_TRUE and not Q_IS_TRUE:
    print("Conclusion: P is TRUE and Q is FALSE. The correct
        option is (a).")
elif not P_IS_TRUE and Q_IS_TRUE:
    print("Conclusion: P is FALSE and Q is TRUE. The correct
        option is (c).")
```

Python and C Code

```python
    else:
        print("Conclusion: Both P and Q are FALSE. The correct
            option is (d).")


if __name__ == "__main__":
    main()
```