FUNCTION inverseMatrix(matrix):

# Kiểm tra ma trận có vuông hay không

IF số hàng matrix ≠ số cột matrix:

THROW lỗi "Matrix must be square."

# Tạo ma trận mở rộng augmented = (matrix | identity)

n = số hàng của matrix

augmented = ma trận kích thước n x (2 \* n) chứa 0

Gán augmented[:, 0:n] = matrix # Sao chép ma trận gốc vào phần bên trái

Gán augmented[:, n:(2 \* n)] = ma trận đơn vị kích thước n x n

# Áp dụng phương pháp Gauss-Jordan

FOR i từ 0 đến n - 1:

# Tìm phần tử không bằng 0 ở đường chéo tại hàng i

IF augmented[i][i] == 0:

swapped = FALSE

FOR j từ i + 1 đến n - 1:

IF augmented[j][i] ≠ 0:

Hoán đổi hàng i và hàng j của augmented

swapped = TRUE

BREAK

IF NOT swapped:

THROW lỗi "Matrix is singular and cannot be inverted."

# Chuẩn hóa hàng i sao cho phần tử đường chéo = 1

pivot = augmented[i][i]

Chia toàn bộ hàng i cho pivot

# Làm các phần tử khác đường chéo ở cột i bằng 0

FOR j từ 0 đến n - 1:

IF j ≠ i:

factor = augmented[j][i]

Trừ factor \* hàng i khỏi hàng j

# Phần bên phải của ma trận augmented là ma trận nghịch đảo

RETURN augmented[:, n:(2 \* n)]

FUNCTION calculateProbability(P, initialState, steps):

# Chéo hóa ma trận P

Sử dụng EigenSolver để tính:

Q = ma trận vector riêng thực của P

Lambda = ma trận đường chéo chứa giá trị riêng thực của P

# Tính ma trận nghịch đảo Q\_inv từ Q

Q\_inv = inverseMatrix(Q)

# Tạo ma trận Lambda^n bằng cách lũy thừa từng giá trị trên đường chéo

Lambda\_n = Lambda

FOR i từ 1 đến steps - 1:

FOR j từ 0 đến kích thước Lambda\_n:

Lambda\_n[j][j] = Lambda\_n[j][j] \* Lambda[j][j]

# Tính ma trận P^n = Q \* Lambda^n \* Q\_inv

P\_n\_temp = Q \* Lambda\_n # Nhân Q với Lambda^n

P\_n = P\_n\_temp \* Q\_inv # Nhân P\_n\_temp với Q\_inv

# In xác suất chuyển đổi trạng thái "Trung bình" sang "Giàu"

FOR i từ 0 đến số hàng của P\_n:

FOR j từ 0 đến số cột của P\_n:

IF i == 1:

IN "P(i|j) = P\_n[i][j]"

# Trả về xác suất từ trạng thái "Trung bình" sang "Giàu"

RETURN P\_n[1][0]

MAIN:

# Định nghĩa ma trận chuyển đổi trạng thái P

P = [

[0.0, 0.75, 0.2, 0.05], # Giàu

[0.05, 0.2, 0.3, 0.45], # Trung bình

[0.1, 0.4, 0.3, 0.2], # Nghèo

[0.0, 0.15, 0.3, 0.55] # Nợ

]

# Định nghĩa trạng thái ban đầu

initialState = [0.0, 1.0, 0.0, 0.0]

# Nhập số bước cần tính

steps = 3

# Tính xác suất chuyển đổi sau từng bước

FOR i từ 1 đến steps:

probability = calculateProbability(P, initialState, i)

IN "Xác suất sau bước", i, "là:", probability