MSSV: 19127102

Họ tên: Võ Hoàng Gia Bảo

Project 2: Viết hàm inverse(A)

Tìm ma trận nghịch đảo theo thuật toán tạo ma trận bổ sung

Bước 1: Kiểm tra định thức của ma trận

```python
def determinant(A):
    # 1x1
    if len(A) == len(A[0]) == 1:
        return A[0][0]

    # 2x2
    if len(A) == len(A[0]) == 2:
        return A[0][0] * A[1][1] - A[1][0] * A[0][1]

    # else
    total = 0
    for removed_col in range(len(A[0])):
        # Lay dong 1 roi lap tat ca cac cot
        sub_A = sub_matrix(A, 0, removed_col)

        sign = (-1) ** (1 + removed_col + 1)

        # Tim dinh thuc con bang de quy
        sub_det = determinant(sub_A)

        # Cong cac dinh thuc con
        total += A[0][removed_col] * sign * sub_det

    return total
```

Nếu ma trận là 1x1 thì trả về phần tử đầu tiên của ma trận
Nếu ma trận là 2x2 thì trả về kết quả tính được từ 2x2
Đối với ma trận 3x3, ở đây sử dụng thuật toán theo dòng đầu tiên của ma trận. Nghĩa là ta tính định thức bằng cách khai triển theo dòng thứ 1, và lặp với tất cả các cột trên dòng 1, chưa xét đến performance của thuật toán. Sau đó tìm dấu và đệ quy để tìm định thức của ma trận con với phần bù đại số và cộng các định thức ma trận phần bù đại số lại

Để tìm định thức của ma trận, ta sẽ xây dựng hàm sub_matrix để xây dựng một ma trận có được với phần bù đại số. Sau đó sẽ sử dụng hàm determinant để tìm định thức.

```
def sub_matrix(A, removed_row, removed_col):
    sub_A = copy.deepcopy(A)

    # Bo dong
    sub_A = np.delete(sub_A, removed_row, 0)

    # Bo cot
    sub_A = np.delete(sub_A, removed_col, 1)

    return sub_A
```

Ở đây sử dụng hàm numpy.delete để loại bỏ dòng cột để thuận tiện nếu người dùng nhập vào là 1 tuple thay vì 1 list

```
def inverse(A):
    det = determinant(A)
    if det == 0:
        print(" Ma tran khong kha nghich ")
        return None

    # 1x1
    if len(A) == len(A[0]) == 1:
        return 1/det
```

Sau khi kiểm tra định thức của ma trận xong thì xét nếu det(A) = 0 thì in ra màn hình "Ma trận không khả nghịch" và trả về None
Nếu ma trận là 1x1 thì trả về 1/det(A)

Bước 2: Chuyển vị ma trận gốc

```
# Tao ma tran bo sung
row = len(A)
col = len(A[0])
B = np.zeros((row, col))

# Tao ma tran chuyen vi tu ma tran goc
A_trans = np.transpose(A)
```

Dùng hàm numpy.transpose để chuyển vị ma trận và tạo ma trận bổ sung với các phần tử 0

Bước 3: Tìm định thức của từng ma trận con với các phần bù đại số liên kết với ma trận chuyển vị mới từ ma trận gốc và tạo ma trận các phần phụ đại số

```
for i in range(row):
    for j in range(col):
        # Tao cac ma tran phan bu dai so
        sub_A = sub_matrix(A_trans, i, j)

        # Tim dau
        sign = (-1) ** (i + j)
        B[i][j] = sign * determinant(sub_A)
```

Bước 5: Thực hiện phép chia của toàn bộ các phần tử của ma trận bổ sung với det(A)

```
# Chia ma tran bo sung voi dinh thuc cua ma tran
B = multiply_number_to_matrix(1/det, B)

# In ra gia tri lam tron 5 chu so thap phan, tra ve ma tran nghich dao
print(np.round(B, 5))
return B
```

Hàm multiply_number_to_matrix dùng để nhân 1 số với ma trận đó

```python
def multiply_number_to_matrix(num, A):
    return [[num * a for a in row] for row in A]
```