

CS206A Data Structure

Project1

Student ID: 20160253, 20160338

name: 박예기 (Park ye gi), 신인철

Part I.

ADT Matrix is

objects : a set of elements which have two indexes, *row* and *column*, where *row* and *column* are integers and form a unique combination.

functions :

for all *matrix*, *matrix1*, *matrix2* \in Matrix, *item* \in element, *m*, *n* \in positive integer

Matrix CreateM (*m*, *n*) ::=

create a matrix which has *m* number of rows and *n* number of columns.

Matrix CreateIdentity (*n*) ::=

create an Identity matrix whose size is *n* \times *n*.

Matrix Add (*matrix1*, *matrix2*) ::=

if (the dimension of *matrix1* and *matrix2* is same)

return the *matrix* produced by adding corresponding *items*.

else return error

Matrix Multiply (*matrix1*, *matrix2*) ::=

if (number of columns in *matrix1* equals number of rows in *matrix2*)

return the matrix produced by multiplying *matrix1* by *matrix2* according to the formula $item = \sum(a[i][k] * b[k][j])$ (This item is (*i*, *j*) element of new matrix)

else return error

Matrix Transpose (*matrix*) ::=

return the matrix whose (*i*, *j*) element is (*j*, *i*) element of *matrix*

Boolean IsSymmetric (*matrix*) ::=

if (*matrix1* == *Transpose*(*matrix*)) **return** TRUE

else return FALSE

Boolean IsSame (*matrix1*, *matrix2*) ::=

if (*matrix1* == *matrix2*) **return** TRUE

else return FALSE

Part II.

Problem(1)

The file name is Matrix.py in Project Folder.

We used construct function(`__init__`) for function CreateM of ADT. This function indicates self.matrix which is two dimensional array and includes the elements of matrix. And also this function indicates self.rows and self.cols which are the number of rows and columns each.

The function Identity indicates self.identity which is two dimensional array and represents an identity matrix of which size is self.cols.

The function Add gets two Matrix objects as parameters. This function determines the two matrices have the same size or not. If they don't have the same size, it prints that they cannot be added and returns 0. If they have the same size, it makes new 2D list, named self.addMatrix and changes the (i, j) element of self.addMatrix with the result of the sum of (i, j) elements.

The function Multiply gets two Matrix objects as parameters. This function determines the column number of first matrix is same as the row number of second matrix. If they are not same, prints that they cannot be multiplied and returns 0. If they are same, it makes new 2D list, named self.mulitMatrix and changes its (i, j) element with the result of the sum of multiplication of ith row and jth column.

The function Transpose makes a new matrix, named self.transMatrix of which size is $j \times i$. (The number of rows is the number of columns of the original matrix and the number of columns is the number of rows of the original matrix.) Then it changes the (i, j) element of self.transMatrix with the (j, i) element of the original matrix.

The function IsSymmetric determines the matrix is same as the self.transMatrix or not. If same return True, and if not same return False.

The function IsSame gets two Matrix objects as parameters and firstly checks they have the same size. And then it checks their elements and the positions of elements are same or not. If not same, return False, and if same, return True.

Problem(2)

Matrix is in form of $m \times n$. And we have to find out time complexity in term of n .

construct function(`__init__`) : Function reads $m \times n$ elements in file and counts the length of `self.matrix` and `self.matrix[0]`. Time complexity of reading elements from file is $O(n)$ and time complexity of computing length of `self.matrix` is $O(1)$ and `self.matrix[0]` is $O(n)$. So, time complexity is $O(n)$.

Function identity : This function has to compute for each (i, j) element such that i is same as j or not. Size of matrix is $n \times n$. So, we have to compute n^2 times. Time complexity is $O(n^2)$.

Function Add : This function have double-for loop to add each element in two matrix. We have to add elements $m \times n$ times. So, Time complexity is $O(n)$.

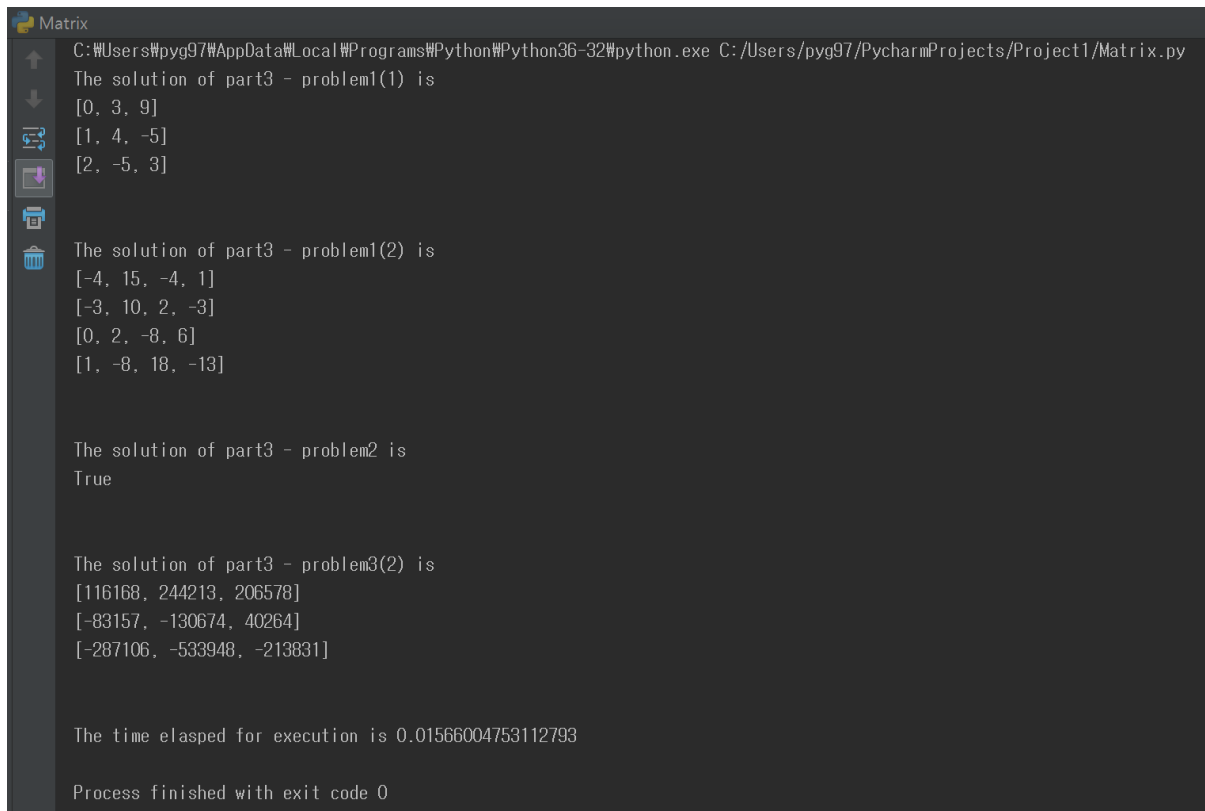
Function Multiply : This function multiply $m \times n$ matrix with $n \times l$ matrix and makes $m \times l$ matrix. When we want to compute each element in $m \times l$ matrix, we have to multiple n elements and add them. Time complexity of multiplying them is $O(n)$ and adding is also $O(n)$. So, time complexity of computing (i, j) element of $m \times l$ matrix is $O(n)$.

Function Transpose : Function compute $m \times n$ times to know each element of matrix `self.transMatrix`. So, time complexity is $O(n)$.

Function IsSymmetric : Function compute $m \times n$ times to know each (i, j) element and (j, i) element is same or not. So, time complexity is $O(n)$.

Function IsSame : Function compute $m \times n$ times to know each element of two matrix is same or not. So, time complexity is $O(n)$.

Part III.



```
Matrix
C:\Users\pyg97\AppData\Local\Programs\Python\Python36-32\python.exe C:/Users/pyg97/PycharmProjects/Project1/Matrix.py
The solution of part3 - problem1(1) is
[0, 3, 9]
[1, 4, -5]
[2, -5, 3]

The solution of part3 - problem1(2) is
[-4, 15, -4, 1]
[-3, 10, 2, -3]
[0, 2, -8, 6]
[1, -8, 18, -13]

The solution of part3 - problem2 is
True

The solution of part3 - problem3(2) is
[116168, 244213, 206578]
[-83157, -130674, 40264]
[-287106, -533948, -213831]

The time elapsed for execution is 0.01566004753112793

Process finished with exit code 0
```

This picture is the results of part3- problem1(1), (2), problem2, problem3(2).

Problem1(1).

The code is included in Matrix.py in Project Folder. The result is on the above picture.

Problem1(2).

The code is included in Matrix.py in Project Folder. The result is on the above picture.

Problem2.

The code is included in Matrix.py in Project Folder. The result is on the above picture.

Because the multiplication of two matrices is same as identity matrix, the two matrices are in inverse relation.

Problem3(1).

We included the code for power function in ADT implementation of part II.

Problem3(2).

The code is included in Matrix.py in Project Folder. The result is on the above picture.