Home work 1

Policy on plagiarism

These are individual homework. While you may discuss the ideas and algorithms or share the test cases with others, at no time may you read, possess, or submit the solution code of anyone else (including people outside this course), submit anyone else's solution code to your account on the gradebot, or allow anyone else to read or possess your source code. We will detect plagiarism using automated tools and will prosecute all violations to the fullest extent of the university regulations, including failing this course, academic probation, and expulsion from the university.

Goal

- Matrices calcluation
- Built-in data structures
- Control flow statements
- I/O
- Modules

Description

Matrices are widely used both in mathematics and computer sciences, such as game thoery and economics. In homework one, you will implement a module for matrix calculation.

Background

- Matrix Wiki
- Sparse Matrix

Syntax of Matrices

- `Matrix` is defined in Backus-Naur Form, see BNF Wiki.

\`\`\`

~~~
Matrix::= StandardMatrix | SparseMatrix
StandardMatrix::= "[" Rows "]" | "[" "]"
Rows::= Row | Row ";" Rows
Row::= element | element "," Row
elment::= int | float | complex
SparseMatrix::= m"-"n "{" Entities "}" | m"-"n "{" "}"
Entities::= Entity | Entity "," Entities
Entity::= "(" m "," n "," element ")"
m,n::= int
~~~

\`\`\`

Remark:

- Now `StandardMatrix` can be an empty matrix `[]`. Thanks to 王一帆 who points out this flaw. Nevertheless, programs that do not consider this special case can still pass testing.

Example:

The following is a `3-by-4` integer matrix:

$$\begin{bmatrix} 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- In **StandardMatrix** syntax is (**no black space**): [0,0,3,0;0,5,0,0;1,0,0,0]
- In **SparseMatrix** syntax is (**no black space**): 3-4{(1,3,3),(2,2,5),(3,1,1)}

Requirements

- Design two ways to store matrices, one for standard matrix and the another for sparse matrix
- Implement the following functions in the module `SISTMatrix.py`
- There is no type transformation between `int`, `float` and `complex`, e.g., For `5.0`, outputs `5.0`,

for `1+1j` outputs `1+1j` instead of `(1+1j)`, for `0+1j`, outputs `1j`, for `0+0j`, outputs `0j`

- In the following, a matrix means the matrix in the representation that you design to store matrices. You can choose any representations.
- a string representing a matrix means that this is a string following the above syntax

``` def IsStandard(A):

```
"""
Input: a matrix `A`
Output: `True` if `A` is stored as a standard matrix, otherwise `False`
"""
```

def Str2Mat(s):

```
"""
Input: a string `s` representing a matrix either in `StandardMatrix` syntax or
`SparseMatrix` syntax
Output: a matrix (i.e., a object in Python) representing the matrix in `s`
"""
```

def Mat2StrStandard(A):

```
"""
Input: a matrix `A`
Output: a string representing the matrix `A` in `StandardMatrix` syntax
"""
```

def Mat2StrSparse(A):

```
"""
```
```

```
Input: a matrix `A`
Output: a string representing the matrix `A` in `SparseMatrix` syntax. The entities
are sorted by first row indices and then column indices. E.g.,
[0,0,3,1;0,5,0,0;1,0,0,0], is printed as string  `3-
4{(1,3,3),(1,4,1),(2,2,5),(3,1,1)}`
"""
```

def Standard2Sparse(A):

```
"""
Input: a matrix `A` storing a standard matrix
Output: an equivalent sparse matrix of `A`
"""
```

def Sparse2Standard(A):

```
"""
Input: a sparse matrix `A`  storing a sparse matrix
Output: an equivalent standard matrix of `A`
"""
```

def MatAdd(A, B):

```
"""
Input: two `m-by-n` matrices `A` and `B`.
Output: a `m-by-n` matrix representing `A + B` (addition).
"""
```

def MatSub(A, B):

```
"""
Input: two `m-by-n` matrices `A` and `B`.
Output: a `m-by-n` matrix representing `A - B` (subtraction).
"""
```

def MatScalarMul(A, c):

```
"""
```

```
Input: a `m-by-n` matrices `A` and a number `c` that is `int`, or `float` or
`complex`.
Output: a `m-by-n` matrix representing `cA` (scalar multiplication).
"""
```

def MatTransposition(A):

```
"""
Input: a `m-by-n` matrix `A`.
Output: a `n-by-m` matrix representing the transpose of `A`.
"""
```

def MatEq(A, B):

```
"""
Input: two `m-by-n` matrices `A` and `B`.
Output: `True` if `A==B`, otherwise `False`.
"""
```

\`\`\`

Check in

- check in `SISTMatrix.py` file into gradebot