mat

Coding the Matrix, 2015

For auto-graded problems, edit the file mat.py to include your solution.

Matrix Class

Problem 1: You will write a module mat implementing a matrix class Mat. The data structure used for instances of Mat resembles that used for instances of Vec. The only difference is that the domain D will now store a pair (i.e., a 2-tuple) of sets instead of a single set. The keys of the dictionary f are pairs of elements of the Cartesian product of the two sets in D.

The operations defined for Mat include entry setters and getters, an equality test, addition and subtraction and negative, multiplication by a scalar, transpose, vector-matrix and matrix-vector multiplication, and matrix-matrix multiplication. Like Vec, the class Mat is defined to enable use of operators such as + and *. The syntax for using instances of Mat is as follows, where A and B are matrices, v is a vector, alpha is a scalar, r is a row label, and c is a column label:

operation	syntax
Matrix addition and subtraction	`A+B` and `A-B`
Matrix negative	`-A`
Scalar-matrix multiplication	`alpha*A`
Matrix equality test	`A == B`
Matrix transpose	`A.transpose()`
Getting and setting a matrix entry	`A[r,c]` and `A[r,c] = alpha`
Matrix-vector and vector-matrix multiplication	`v*A` and `A*v`
Matrix-matrix multiplication	`A*B`

You are required to write the procedures equal, getitem, setitem, mat_add, mat_scalar_mul, transpose, vector_matrix_mul, matrix_vector_mul, and matrix_matrix_mul. You should start by getting equal working since == is used in the doctests for other procedures.

Note: You are encouraged to use operator syntax (e.g. $M[r,c]$) in your procedures. (Of course, you can't, for example, use the syntax $M[r,c]$ in your getitem procedure.)

Put the file mat.py in your working directory, and, for each procedure, replace the pass statement with a working version. Test your implementation using doctest as you did with vec.py. Make sure your implementation works with matrices whose row-label sets differ from their column-label sets.

Note: Use the sparse matrix-vector multiplication algorithm described in lecture (the one based on the "ordinary" definition") for matrix-vector multiplication. Use the analogous algorithm for vector-matrix multiplication. Do not use transpose in your multiplication algorithms. Do not use any external procedures or modules other than vec. In particular, do not use procedures from matutil. If you do, your Mat implementation is likely not to be efficient enough for use with large sparse matrices.

There is a doctest file mat_sparsity.py that you can try out with your Mat implementation:

```
python3 -m doctest mat_sparsity.py
```

or

```
py -3 -m doctest mat_sparsity.py
```

If your implementation exploits sparsity in transpose, and in vector-matrix, matrix-vector, and matrix-matrix multiplication, these tests should take very little time. (For matrix-matrix multiplication, you need only exploit sparsity in one of the two matrices.)

If your implementation does not exploit sparsity, don't worry. It's not vital for the assignments in this course, but it's needed for more advanced assignments such as Pagerank.