A. Полиномы. 2021

Ограничение времени	1.5 секунд
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Реализуйте модуль `polynomial.py`, содержащий класс `Polynomial`, который описывает полином. Конструктор класса (метод `__init__`) принимает неограниченное число аргументов, каждый из которых является коэффициентом полинома. Номер аргумента соответствует степени монома, к которому относится данный коэффициент (нумерация с нуля).
Например, `Polynomial(2, 3, 1)` задает полином $2+3x+x22+3x+x2$.
Объект класса моделирует конкретный полином. При вызове объекта класса от вещественной переменной `x`, возвращается значение полинома в точке `x`.

Например:

```
polynom = Polynomial(2, 3, 1)

print(polynom(4))

>>> 30
```

Замечание: Допускается написать реализацию класса `Polynomial` без использования классов, только с помощью функций.

B. Разреженные матрицы. 2021

Ограничение времени	6 секунд
Ограничение памяти	16.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Реализуйте модуль `sparse_matrix.py`, содержащий класс `CooSparseMatrix`, который реализует разреженную матрицу с координатным форматом хранения. В памяти должны хранится только ненулевые элементы матрицы, то есть затраты по памяти должны составлять $O(n)O(n)$, где nn — число ненулевых элементов матрицы.

Условия:

1. Конструктор класса (метод `__init__`) принимает два аргумента:
   - `ijx_list` — список кортежей `(i, j, x)`, где `i`, `j` — координаты элемента в матрице, `x` — значение элемента;
   - `shape` — кортеж из двух элементов, размер матрицы.

   Необходимо выбрасывать исключение `TypeError` в случаях, когда в метод `__init__` приходят любые некорректные данные (продумайте такие сценарии самостоятельно).

```
matrix = CooSparseMatrix(
    ijx_list=[
        (0, 0, 1),
        (1, 0, 2)
    ], shape=(2, 2)
)
```

2. Необходимо реализовать простую индексацию матриц. При вызове `matrix[i]` необходимо вернуть объект `CooSparseMatrix`, соответствующий строке с номером `i` (нумерация с нуля) в исходной матрице размера 11 на количество столбцов в матрице. При вызове `matrix[i, j]` необходимо вернуть элемент матрицы с координатами `i`, `j`. Также реализуйте возможность присвоить элементу матрицы с координатами `i`, `j` вещественное число.

   Необходимо выбрасывать исключение `TypeError` в случаях, когда в методы индексации приходят любые некорректные данные (продумайте такие сценарии самостоятельно).

```
matrix = CooSparseMatrix(
    ijx_list=[
        (0, 0, 1),
```

```
            (1, 0, 2)
        ], shape=(2, 2)
    )

    to_array(matrix[0])
    >>> array([[1., 0.]])

    matrix[0, 0]
    >>> 1

    matrix[1, 1] = 3
    matrix[1, 1]
    >>> 3
```

Замечание: Также рекомендуется реализовать для себя функцию или метод `to_array`, который преобразует разреженную матрицу в `numpy` матрицу. Так как на проверяющем сервере нет библиотеки `numpy`, при засылке файла с импортом `numpy` будет возникать ошибка.

```
matrix = CooSparseMatrix(
    ijx_list=[
        (0, 0, 1),
        (1, 0, 2)
    ], shape=(2, 2)
)

to_array(matrix)
>>> array([[1., 0.],
           [2., 0.]])
```

C. Сложение матриц. 2021

Ограничение времени	7 секунд
Ограничение памяти	5.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Добавьте в класс `CooSparseMatrix` возможность сложения и вычитания матриц, а также умножения матриц на число. При сложнении и вычитании матриц разного размера должно выбрасываться исключение `TypeError`.

```
A = CooSparseMatrix(
    ijx_list=[
        (0, 0, 1),
        (1, 0, 2)
    ], shape=(2, 2)
)
B = CooSparseMatrix(
    ijx_list=[
        (0, 1, 2),
        (1, 0, 1)
    ], shape=(2, 2)
)
C = A + B

to_array(C)
>>> array([[1., 2.],
           [3., 0.]])

to_array(A * 5)
>>> array([[ 5.,  0.],
           [10.,  0.]])
```

D. Изменение формы матрицы. 2021

Ограничение времени	0.1 секунд
Ограничение памяти	5.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

1. Добавьте в класс `CooSparseMatrix` следующие атрибуты:
Атрибут `shape` с помощью которого можно изменять размер массива.
Значение атрибута — кортеж из двух целых положительных чисел. При попытке присвоить атрибуту что-либо другое должно выбрасываться исключение `TypeError`. При попытке присвоить атрибуту размер, не согласованный с текущим (например, при попытке изменить размер матрицы с $(2,5)$ на $(3,4)$), должно выбрасываться исключение `TypeError`. При корректном присваивании необходимо изменить размер матрицы согласно C-order.

```
matrix = CooSparseMatrix(
    ijx_list=[
        (0, 0, 1),
        (1, 1, 2),
        (2, 3, 5),
        (1, 3, 0)
    ], shape=(3, 5)
)
matrix.shape
>>> (3, 5)


matrix.shape = (5, 3)
to_array(matrix)
>>> array([[1., 0., 0.],
           [0., 0., 0.],
           [2., 0., 0.],
           [0., 0., 0.],
           [0., 5., 0.]])
```

2. Атрибут `T`, возвращающий транспонированную матрицу. При попытке присваивания атрибуту должно выбрасываться исключение `AttributeError`. Обращение к атрибуту не должно влиять на исходную матрицу.

```
matrix = CooSparseMatrix(
    ijx_list=[
        (0, 0, 1),
        (1, 1, 2),
        (2, 3, 5),
        (1, 3, 0)
    ], shape=(3, 5)
)
to_array(matrix.T)
>>> array([[1., 0., 0.],
           [0., 2., 0.],
           [0., 0., 0.],
           [0., 0., 5.],
           [0., 0., 0.]])

matrix.shape
          >>> (3, 5)
```