Звіт по лабораторному завданню №1.

Автори:
Гусєв Радомир
Бронецький Володимир

```python
def read_file(name):
    """
    (str) -> list
    Gets a text matrix and returns it as a list
    """
    txt_matrix = [i.replace('\n', '') for i in open(name, "r").readlines()]
    matrix = []
    for i in range(len(txt_matrix)):
        line = str(txt_matrix[i])
        txt_matrix[i] = [_ for _ in line]
        line_math = []
        for j in range(len(txt_matrix[i])):
            line_math.append(int(txt_matrix[i][j]))
        matrix.append(line_math)
    return matrix
```

1) Зчитує матрицю з текстового файлу і перетворює її на список.

```python
def write_matrix_to_file(matrix, name):
    """
    (list) -> str
    Gets the matrix in the form of a list, and writes it to a file
    """
    txt_matrix = open(name, "w")
    for line in range(len(matrix)):
        txt_matrix.writelines(str(matrix[line]).replace(' ', '').replace('[', '').replace(']', '').replace(',',
```

2) Записує матрицю у текстовий файл.

```python
def find_reflexive_closure(matrix: list) -> list:
    """
    Returns the reflexive closure of a given matrix
    -------
    list
        A list of lists representing the reflexive closure of the given matrix

    Doctests
    --------
    >>> find_reflexive_closure([[0, 1, 0], [0, 0, 1], [1, 0, 0]])
    [[1, 1, 0], [0, 1, 1], [1, 0, 1]]
    >>> find_reflexive_closure([[0, 1, 1], [0, 0, 1], [0, 0, 0]])
    [[1, 1, 1], [0, 1, 1], [0, 0, 1]]
    """
    for i in range(len(matrix)):
        matrix[i][i] = 1
    write_matrix_to_file(matrix, 'Reflexive.txt')
    return matrix
```

3) Повертає список списків, що представляють рефлексивне замикання певної матриці. Перевіряє чи елементи діагоналі = 1.

```python
def find_symmetric_closure(matrix: list) -> list:
    """
    Returns the symmetric closure of a given matrix
    --------
    >>> find_symmetric_closure([[0, 1, 0], [0, 0, 1], [1, 0, 0]])
    [[0, 1, 1], [1, 0, 1], [1, 1, 0]]
    >>> find_symmetric_closure([[1, 1, 1], [0, 0, 1], [0, 0, 0]])
    [[1, 1, 1], [1, 0, 1], [1, 1, 0]]
    """
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if matrix[i][j] == 1:
                matrix[j][i] = 1
    write_matrix_to_file(matrix, 'Symmetric.txt')
    return matrix
```

4) Повертає список списків, що представляють симетричне замикання даної матриці. Замінює кожен [j][i] на [i][j], якщо [i][j] дорівнює 1

```python
def find_transitive_closure(matrix: list, write_to_file: bool = True) -> list:
    """
    Find a transitive closure of a matrix
    --------
    >>> find_transitive_closure([[1, 1, 0], [0, 1, 1], [0, 0, 1]])
    [[1, 1, 1], [0, 1, 1], [0, 0, 1]]
    >>> find_transitive_closure([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
    >>> find_transitive_closure([[0, 1, 0], [1, 1, 0], [0, 0, 1]])
    [[1, 1, 0], [1, 1, 0], [0, 0, 1]]
    """
    length = len(matrix)
    new_matrix = [_[:] for _ in matrix]
    for k in range(length):
        for i in range(length):
            for j in range(length):
                if new_matrix[i][k] and new_matrix[k][j]:
                    new_matrix[i][j] = 1
    if write_to_file:
        write_matrix_to_file(new_matrix, 'Transitive.txt')
    return new_matrix
```

5) Знаходить транзитивне замикання матриці. Спочатку копіює матрицю, потім виконується алгоритм Воршала та записує його у текстовий файл (якщо цього не було зроблено раніше).

```python
def split_equivalent_relation(matrix: list) -> list:
    """
    Split an equivalence relation into equivalence classes.
    --------
    >>> split_equivalent_relation([[1, 1, 0], [1, 1, 0], [0, 0, 1]])
    [{1, 2}, {3}]
    >>> split_equivalent_relation([[0, 1, 1], [1, 1, 1], [0, 1, 1]])
    [{1, 3}, {2}]
    >>> split_equivalent_relation([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    [{1}, {2}, {3}]
    """
    classes: list = []
    for i, row_i in enumerate(matrix):
        for j, row_j in enumerate(matrix):
            if row_i == row_j:
                if all((i + 1 not in c for c in classes)):
                    classes.append(set([i + 1, j + 1]))
                else:
                    for equivalence_class in classes:
                        if i + 1 in equivalence_class:
                            equivalence_class.add(j + 1)
    _write_equivalence_classes(classes)
    return classes
```

6) Розбиває відношення еквівалентності на класи еквівалентності. Перевіряє кожні 2 ряди. Якщо вони еквівалентні, додає їх до одного класу. Якщо новий клас ще не створений, створює його, а якщо створений – додає до існуючого.

```python
def _write_equivalence_classes(classes: list):
    """Write the equivalence classes to a file."""
    with open('Equivalence classes.txt', 'w', encoding='utf-8') as file:
        file.write(str(classes))
```

7) Записує класи еквівалентності у файл.

```python
def is_transitive(matrix: list) -> bool:
    """
    Check if a relation is transitive.
    --------
    >>> is_transitive([[1, 0, 1, 1], [1, 1, 1, 1], [1, 0, 1, 1], [1, 0, 1, 1]])
    True
    >>> is_transitive([[1, 1], [1, 1]])
    True
    >>> is_transitive([[1, 1], [1, 0]])
    False
    """
    if matrix == find_transitive_closure(matrix, False):
        return True
    return False
```

8) Перевіряє відношення на транзитивність.
Перевіряє чи транзитивне відношення таке ж, як оригінал. Якщо так, тоді повертає True, а інакше – False.

```python
def find_number_of_transitive(n: int) -> int:
    """
    Count the number of transitive closures on a set of n elements.
    --------
    # >>> find_number_of_transitive(4)
    # 3994
    >>> find_number_of_transitive(3)
    171
    >>> find_number_of_transitive(2)
    13
    """
    def _get_matrixes(j: int) -> list:...

    matrixes: list = _get_matrixes(n)
    return len(matrixes)
```

9) Підраховує кількість транзитивних замикань на наборі з n елементів.

```python
def _get_matrixes(j: int) -> list:
    """
    Get the matrixes for all closures of ixj
    """
    transitive_matrixes = []
    numbers = []
    for _ in range(2 ** (j * j)):
        numbers.append(f"{str(bin(_))[2:]:0>{j*j}}")
    for _ in range(len(numbers)):
        matrix: list = []
        for k in range(0, j):
            matrix.append([])
            for q in range(0, j):
                matrix[k].append(int(numbers[_][k * j : (k + 1) * j][q]))
        if is_transitive(matrix):
            transitive_matrixes.append(matrix)
    return transitive_matrixes
```

10) Повертає матриці для всіх замикань і на j.

Генерування всіх можливих ліній з 1 і 0 довжини j.

Перетворює кожен рядок на матрицю, розбиває кожні j елементів на рядок, додається порожня матриця. Після - до кожного рядка додається порожній список.

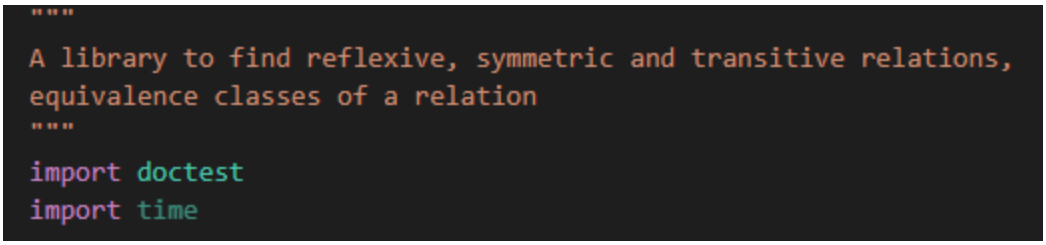
Отримується кожен елемент числа та додається до рядка (елемент потрібно взяти від j*k до j*(k+1)). Таким чином ми розбиваємо число на рядки.

Якщо матриця є транзитивним замиканням, тоді додаємо її до списку.

```python
def _main():
    """Run doctests."""
    print(doctest.testmod())


if __name__ == "__main__":
    _main()
```

Доктести.


"""
A library to find reflexive, symmetric and transitive relations,
equivalence classes of a relation
"""
import doctest
import time

Бібліотеки.