МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

КЛАССИФИКАЦИЯ БИНАРНЫХ ОТНОШЕНИЙ И СИСТЕМЫ ЗАМЫКАНИЙ

ЛАБОРАТОРНАЯ РАБОТА

студента 3 курса 331 группы
специальности 100501 — Компьютерная безопасность
факультета КНиИТ
Окунькова Сергея Викторовича

Проверил
аспирант _____ В. Н. Кутин

Саратов 2022

СОДЕРЖАНИЕ

1 Постановка задачи

Цель работы:

Изучение основных свойств бинарных отношений и операций замыкания бинарных отношений.

Порядок выполнения работы:

1. Разобрать основные определения видов бинарных отношений и разработать алгоритмы классификации бинарных отношений.
2. Изучить свойства бинарных отношений и рассмотреть основные системы замыкания на множестве бинарных отношений.
3. Разработать алгоритмы построения основных замыканий бинарных отношений.

2 Теоретические сведения по рассмотренным темам с их обоснованием

2.1 Определение бинарного отношения:

Подмножества декартова произведения $A \cdot B$ множеств A и B называются **бинарными отношениями** между элементами множеств A, B и обозначаются строчными греческими буквами: $\rho, \sigma, \rho_1, \rho_2, ...$

Для бинарного отношения $\rho \subset A \cdot B$ область определения D_ρ и множество значений E_ρ определяется как подмножества соответствующих множеств A и B по следующим формулам:

$$D_\rho = \{a : (a, b) \in \rho \text{ для некоторого } b \in B\}$$

$$E_\rho = \{b : (a, b) \in \rho \text{ для некоторого } a \in A\}$$

2.2 Свойства бинарных отношений:

Бинарное отношени является:

1. *рефлексивным*, если $(a, a) \in \rho$ для любого $a \in A$;
2. *антирефлексивным*, если $(a, a) \notin \rho$ для любого $a \in A$;
3. *симметричным*, если $(a, b) \in \rho \Rightarrow (b, a) \in \rho$;
4. *антисимметричным*, если $(a, b) \in \rho$ и $(b, a) \in \rho \Rightarrow a = b$;
5. *транзитивным*, если $(a, b) \in \rho$ и $(b, c) \in \rho \Rightarrow (a, c) \in \rho$
6. *антитранзитивным*, если $(a, b) \in \rho$ и $(b, c) \in \rho \Rightarrow (a, c) \notin \rho$

2.3 Классификация бинарных отношений:

Классификация бинарных отношений напрямую определяются их свойствами.

1. Рефлексивное транзитивное отношение называется отношением квазипорядка.
2. Рефлексивное симметричное транзитивное отношение называется отношением эквивалентности.
3. Рефлексивное антисимметричное транзитивное отношение называется отношением (частичного) порядка.
4. Антирефлексивное, антисимметричное и транзитивное отношение называется отношением строгого порядка.

2.4 Замыкание отношения:

Замыканием отношения R относительно свойства P называется такое множество R^*, что:

1. $R \subset R^*$.
2. R^* обладает свойством P.
3. R^* является подмножеством любого другого отношения, содержащего R и обладающего свойством P.

3 Результаты работы

3.1 Описание алгоритма классификации бинарных отношений

1. Чтобы проверить бинарное отношение на рефлексивность достаточно посмотреть на главную диагональ матрицы бинарного отношения. Если все ее элементы являются 1, то бинарное отношение является рефлексивным, если же все ее элементы 0, та бинарное отношение является антирефлексивным, иначе бинарное отношение является не рефлексивным.
2. Чтобы проверить бинарное отношение на симетричность нужно транспонировать его матрицу и сравнить полученную с исходной, если они равны, то отношение симметрично, иначе нужно проверить на антисиметричность: если при умножении матрицу бинарного отношения на себя транспонированную на главной диагонали стоят все 0, то бинарное отношение, явлчется антисиметричным, иначе не симетричным.
3. Чтобы проверить бинарное отношение на транзитивность нужно возвести его матрицу в квадрат и сравнить полученную матрицу с исходной, если исходная матрица $\leq$ полученной матрицы, то бинарное отношение является транзитивным, иначе следует поэлементно проверить все элементы бинарного отношения на выполнение свойства антитранзитивности, если все элементы удовлетворяют свойству антитранзитивности, то отношение антитранзитивное, иначе отношение не является транзитивным.

3.2 Описание алгоритмов построения основных замыканий бинарных отношений

1. Чтобы построить замыкание на рефлексивности нужно дополнить главную диагональ матрицы бинарного отношения 1.
2. Чтобы построить замыкание на рефлексивности нужно доставить на симетричных позициях матрицы ($a[i][j]$ и $a[j][i]$) 1, если $a[i][j] = 1$.
3. Чтобы построить замыкание транзитивности нужно дополнить матрицу до отношениия транзитивности, в моей реализации это достигается за счет прохождения матрицы 3 циклами и установки значения $a[i][j] = 1$, если $a[i][k] = 1 \wedge a[k][j] = 1$.

3.3 Коды программ, реализующей рассмотренные алгоритмы

```
import numpy as np
import json
```

```
def isTran(a, n):
    b = np.matmul(a, a)
    for i in range(n):
        for j in range(n):
            if b[i][j]:
                b[i][j] = b[i][j] / b[i][j]
    f = (a >= b).all()
    if f:
        print("Set is transitive")
        return 't'
    else:
        f1 = True
        for i in range(n):
            for j in range(n):
                for k in range(n):
                    if a[i][k] and a[k][j] and a[i][j]:
                        f1 = False
        if f1:
            print("Set is anti-transitive")
            return 'at'
        else:
            print("Set is not transitive")
            return 'nt'


def isSymm(a):
    b = a.transpose()
    if np.array_equal(a, b):
        print("Set is symmetry")
        return 's'
    else:
        f = True
        b = np.matmul(a, b)
        for i in range(len(b)):
            for j in range(len(b[i])):
                if b[i][j] == 1 and i != j:
                    f = False
                    break
        if f:
```

```
            print("Set is anti-symmetry")
            return 'as'
        else:
            print("Set is not symmetry")
            return 'ns'


def isRefl(a, n):
    sum = 0
    for i in range(n):
        sum += a[i][i]
    if sum == n:
        print("Set is reflexive")
        return 'r'
    elif sum == 0:
        print("Set is anti-reflexive")
        return 'ar'
    else:
        print("Set is not reflexive")
        return 'nr'


def makeRefl(a, n):
    for i in range(n):
        a[i][i] = 1


def makeSymm(a, n):
    for i in range(n):
        for j in range(n):
            if a[i][j]:
                a[j][i] = 1


def makeTran(a, n):
    for k in range(n):
        for i in range(n):
            for j in range(n):
                if a[i][k] and a[k][j]:
                    a[i][j] = 1
```

```
def get_set(a):
    s = []
    for i in range(len(a)):
        for j in range(len(a[i])):
            if a[i][j] == 1:
                s.append((i + 1, j + 1))
    return s


if __name__ == "__main__":
    print("How you want enter your relation (set or matrix)?")
    enter = input()
    if enter == 'set':
        print('Enter the number of elements in relation')
        n = int(input())
        print("Enter your set")
        # st = [(1, 3), (3, 4), (1, 4), (2, 5), (5, 3)]
        s = list(map(str, input().split(' ')))
        st = []
        for c in s:
            a = ''
            i = 1
            while c[i] != ',':
                a += c[i]
                i += 1
            i += 1
            b = ''
            while c[i] != ')':
                b += c[i]
                i += 1
            st.append((int(a), int(b)))
        a = np.zeros((n, n), int)
        for s in st:
            a[s[0] - 1][s[1] - 1] = 1
        print('----------------------------------------')
        print("Relation's matrix")
        print(a)
    else:
        print('Enter the number of elements in relation')
        n = int(input())
        print("Enter your matrix")
```

```
    a = [list(map(int, input().split())) for i in range(n)]
    a = np.array(a).reshape(n, n)
    print('----------------------------------------')
    print("Relation's set")
    print(*get_set(a))
print('----------------------------------------')
properties = {"reflexive": isRefl(a, n), "symmetry": isSymm(a), "transitive": isT
print('----------------------------------------')
if properties['reflexive'] == 'r' and properties['symmetry'] == 's' and properties
    print('Relation is equivalent')
    print('----------------------------------------')
if properties['reflexive'] == 'r' and properties['transitive'] == 't':
    print('Relation is the relation of the quasi-order')
    print('----------------------------------------')
if properties['reflexive'] == 'r' and properties['symmetry'] == 'as' and properti
    print('Relation is the relation of the partial order')
    print('----------------------------------------')
if properties['reflexive'] == 'ar' and properties['symmetry'] == 'as' and properti
    print('Relation is the relation of the strict order')
    print('----------------------------------------')
if properties['reflexive'] != 'r':
    makeRefl(a, n)
if properties['symmetry'] != 's':
    makeSymm(a, n)
if properties['transitive'] != 't':
    makeTran(a, n)
print('Closure matrix: ')
print(a)
print('----------------------------------------')
print('Closure set: ')
print(*get_set(a))
print('----------------------------------------')
```

3.4 Результаты тестирования программ

```
How you want enter your relation (set or matrix)?
set
Enter the number of elements in relation
3
Enter your set
(1,2) (1,3)
----------------------------------------
Relation's matrix
[[0 1 1]
 [0 0 0]
 [0 0 0]]
----------------------------------------
Set is anti-reflexive
Set is anti-symmetry
Set is transitive
----------------------------------------
Relation is the relation of the strict order
----------------------------------------
Closure matrix:
[[1 1 1]
 [1 1 0]
 [1 0 1]]
----------------------------------------
Closure set:
(1, 1) (1, 2) (1, 3) (2, 1) (2, 2) (3, 1) (3, 3)
----------------------------------------
```

Рисунок 1 – Тест 1

```
How you want enter your relation (set or matrix)?
set
Enter the number of elements in relation
5
Enter your set
(3,2) (2,5) (4,2)
----------------------------------------
Relation's matrix
[[0 0 0 0 0]
 [0 0 0 0 1]
 [0 1 0 0 0]
 [0 1 0 0 0]
 [0 0 0 0 0]]
----------------------------------------
Set is anti-reflexive
Set is not symmetry
Set is anti-transitive
----------------------------------------
Closure matrix:
[[1 0 0 0 0]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]]
----------------------------------------
Closure set:
(1, 1) (2, 2) (2, 3) (2, 4) (2, 5) (3, 2) (3, 3) (3, 4) (3, 5) (4, 2) (4, 3) (4, 4) (4, 5) (5, 2) (5, 3) (5, 4) (5, 5)
----------------------------------------
```

Рисунок 2 – Тест 2

```
How you want enter your relation (set or matrix)?
matrix
Enter the number of elements in relation
5
Enter your matrix
0 0 0 0 0
0 0 0 0 1
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
----------------------------------------
Relation's set
(2, 5) (3, 2) (4, 2)
----------------------------------------
Set is anti-reflexive
Set is not symmetry
Set is anti-transitive
----------------------------------------
Closure matrix:
[[1 0 0 0 0]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]
 [0 1 1 1 1]]
----------------------------------------
Closure set:
(1, 1) (2, 2) (2, 3) (2, 4) (2, 5) (3, 2) (3, 3) (3, 4) (3, 5) (4, 2) (4, 3) (4, 4) (4, 5) (5, 2) (5, 3) (5, 4) (5, 5)
----------------------------------------
```

Рисунок 3 – Тест 3

```
How you want enter your relation (set or matrix)?
matrix
Enter the number of elements in relation
5
Enter your matrix
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
----------------------------------------
Relation's set
(1, 2) (2, 3) (4, 3)
----------------------------------------
Set is anti-reflexive
Set is not symmetry
Set is anti-transitive
----------------------------------------
Closure matrix:
[[1 1 1 1 0]
 [1 1 1 1 0]
 [1 1 1 1 0]
 [1 1 1 1 0]
 [0 0 0 0 1]]
----------------------------------------
Closure set:
(1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) (2, 3) (2, 4) (3, 1) (3, 2) (3, 3) (3, 4) (4, 1) (4, 2) (4, 3) (4, 4) (5, 5)
----------------------------------------
```

Рисунок 4 – Тест 4

3.5 Оценки сложности рассмотренных алгоритмов

3.5.1 Алгоритм определения рефлексивности

Сложность выполнения проверки на рефлексивность или антирефлексивность определяется как $O(n)$.

3.5.2 Алгоритм определения симметричности

Сложность транспонирования в numpy определяется как $O(n^{3/2}log\ n)$, сложность умножения матриц определяется как $O(n^3)$, сложность сравнение двух матриц поэлементно определяется как $O(n^2)$. Отсюда можно сделать вывод, что в случае, если наше отношение будет симметричным, что будет являтся лучшим случаем раоты алгоритма,то общая сложность алгоритма будет определятся как $O(n^{3/2}log\ n + n^2) = O(n^{3/2}log\ n)$, иначе, в худшем случае, сложность будет определятся как $O(n^{3/2}log\ n + n^2 + n^2 + n^3) = O(n^3)$

3.5.3 Алгоритм определения транзитивности

Из всего выше сказанного очевидно, что сложность проверки на транзитивность или антитранзитивность составляет $O(n^3)$, так как в нем используется

умножение, сравнение матриц и тройной цикл для проверки рефлексивности в худшем случае матриц.

3.5.4 Алгоритм классификации

Сложность выполнения самого алгоритма классификации бинарных отношений реализованно через питоновский словарь и оператор if, поэтому является константной ($O(1)$), если не учитывать сложность выполнения проверки свойств отношения.

3.5.5 Построение замыкания рефлексивности

Так как весь алгоритм строится на заполнении главной диагонали матрицы 1, то его сложность состовляет $O(n)$.

3.5.6 Построение замыкания симметричности

Для посторения замыкания симметричности используются вложенный цикла, поэтому сложность алгоритма определяется как $O(n^2)$.

3.5.7 Построение замыкания транзитивности

Для посторения замыкания транзитивности используются два вложенный цикла, поэтому сложность алгоритма определяется как $O(n^3)$.

ЗАКЛЮЧЕНИЕ

В рамках данной лабораторной работы были рассмотренны теоритические основы свойств бинарных отношений, их видов и методов их замыкания по каждому из свойств. На основе этой теоретической части была смоделирована программа, которая способна определить свойства заданного множества, его вид и построить систему замыкания по каждому из основных свойств бинарного отношения.