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

УНИВЕРСАЛЬНЫЕ АЛГЕБРЫ И АЛГЕБРА ОТНОШЕНИЙ

ЛАБОРАТОРНАЯ РАБОТА

студента 3 курса 331 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Токарева Никиты Сергеевича

Проверил
аспирант _____________ В. Н. Кутин

Саратов 2022

1 Постановка задачи

Цель работы – изучение основных понятий универсальной алгебры и операций над бинарными отношениями.

Порядок выполнения работы:

1. Рассмотреть понятие алгебраической операции и классификацию свойств операций. Разработать алгоритмы проверки свойств операций: ассоциативность, коммутативность, идемпотентность, обратимость, дистрибутивность.
2. Рассмотреть основные операции над бинарными отношениями. Разработать алгоритмы выполнения операции над бинарными отношениями.
3. Рассмотреть основные операции над матрицами. Разработать алгоритмы выполнения операций над матрицами.

2 Теоретические сведения по рассмотренным темам с их обоснованием

2.1 Понятие алгебраической операции

Отображение $f : A^n \to A$ называется алгебраической n-арной операцией или просто **алгебраической операцией** на множестве A. При этом n называется порядком или арностью алгебраической операции f.

Далее для бинарной операции f по возможности будем использовать мультипликативную запись с помощью символа «$\cdot$», т.е.вместо $f(x, y)$ писать $x \cdot y$. При необходимости для бинарной операции f используется также аддитивная запись с помощью символа «$+$», т.е. вместо $f(x, y)$ записывается $x + y$.

2.2 Классификация свойств операций

Бинарная операция $\cdot$ на множестве A называется:

- идемпотентной, если $\forall x \in A$ выполняется равенство $x \cdot x = x$;
- коммутативной, если $\forall x, y \in A$ выполняется равенство $x \cdot y = y \cdot x$;
- ассоциативной, если $\forall x, y, z \in A$ выполняется равенство $x \cdot (y \cdot z) = (x \cdot y) \cdot z$;
- обратимой, если $\forall x, y \in A$, если уравнения $x \cdot a = y$ и $b \cdot x = y$ имеют решение, причем единственное;
- дистрибутивной относительно операции +, если $\forall x, y, z \in A$ выполняются равенства

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z),$$
$$(y + z) \cdot x = (y \cdot x) + (z \cdot x);$$

2.3 Основные операции над бинарными отношениями

- Над бинарными отношениями можно выполнять любые теоретико-множественные операции, в частности операции объединения $\cup$ и пересечения $\cap$;
- Обратным для бинарного отношения $\rho \subset A \times B$ называется бинарное отношение $\rho^{-1} \subset B \times A$, определяющееся по формуле:

$$\rho^{-1} = \{(b, a) : (a, b) \in \rho\};$$

- Композицией бинарных отношений $\rho \subset A \times B$ и $\sigma \subset B \times C$ называется бинарное отношение $\rho \circ \sigma \subset A \times C$, определяющееся по формуле:

$$\rho \circ \sigma = \{(a, c) : (a, b) \in \rho \text{ и } (b, c) \in \sigma \text{ для некоторого } b \in B\};$$

2.4 Основные операции над матрицами

— Сложение и вычитание матриц.

Суммой $A + B$ матриц $A_{m\times n} = (a_{ij})$ и $B_{m\times n} = (b_{ij})$ называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = a_{ij} + b_{ij}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

Разностью $A - B$ матриц $A_{m\times n} = (a_{ij})$ и $B_{m\times n} = (b_{ij})$ называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = a_{ij} - b_{ij}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

— Умножение матрицы на число.

Произведением матрицы $A_{m\times n} = (a_{ij})$ на число α называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = \alpha a_{ij}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

— Произведение двух матриц.

Произведением матриц $A_{m\times n} = (a_{ij})$ на матрицу $B_{m\times n} = (b_{ij})$ называется матрица $C_{m\times n} = (c_{ij})$, где $c_{ij} = \sum\limits_{k=1}^{n} a_{ik}b_{kj}$ для всех $i = \overline{1, m}$ и $j = \overline{1, n}$.

— Транспонирование матрицы.

Транспонированной по отношению к матрице $A_{m\times n} = (a_{ij})$ называется матрица $A^T_{n\times m} = (a^T_{ij})$ для элементов которой $a^T_{ij} = a_{ji}$.

— Обращение матрицы.

Обращение матрицы $A_{m\times n}$ - получение матрицы A^{-1}, обратной к исходной матрице A. Обратная матрица A^{-1} — такая, при умножении которой на исходную матрицу A получается единичная матрица E. Это такая матрица, которая удовлетворяет равенству

$$AA^{-1} = A^{-1}A = E$$

3 Результаты работы

3.1 Описание алгоритмов проверки свойств операций: ассоциативность, коммутативность, идемпотентность, обратимость, дистрибутивность

Алгоритм 1 – Проверка бинарной операции «$\cdot$» на идемпотентность

Вход: Матрица $A = (a_{ij})$ бинарной операции «$\cdot$» размерности $n \times n$, список элементов l множества X над операцией «$\cdot$».

Выход: «Бинарная операция «$\cdot$» обладает свойством идемпотентности» или «Бинарная операция «$\cdot$» не обладает свойством идемпотентности».

Шаг 1. Инициализировать булеву переменную $is\_idempotent = true$.

Шаг 2. Создать цикл с индексом i $(0 \leq i < n)$ и пройти по всем элементам $l[i]$ множества X. Если хотя бы один элемент $a_{ii} \neq l[i]$, то присвоить $is\_idempotent = false$.

Шаг 3. Если после завершения цикла $is\_idempotent = true$, то вывести в консоль следующее сообщение: «Бинарная операция «$\cdot$» обладает свойством идемпотентности». Если $is\_idempotent = false$ – вывести сообщение: «Бинарная операция «$\cdot$» не обладает свойством идемпотентности».

Оценка сложности данного алгоритма равна $O(n)$.

Алгоритм 2 – Проверка бинарной операции «$\cdot$» на коммутативность

Вход: Матрица $A = (a_{ij})$ бинарной операции «$\cdot$» размерности $n \times n$.

Выход: «Бинарная операция «$\cdot$» обладает свойством коммутативности» или «Бинарная операция «$\cdot$» не обладает свойством коммутативности».

Шаг 1. Инициализировать булеву переменную $is\_commutative = true$.

Шаг 2. Создать цикл с индексом i $(0 \leq i < n)$, в котором создать вложенный цикл с индексом j $(0 \leq j < n)$, чтобы пройти по всем элементам a_{ij} матрицы A. Если хотя бы один элемент $a_{ij} \neq a_{ji}$, то присвоить $is\_commutative = false$.

Шаг 3. Если после завершения цикла $is\_commutative = true$, то вывести в консоль следующее сообщение: «Бинарная операция «$\cdot$» обладает свойством коммутативности». Если $is\_commutative = false$ – вывести сообщение: «Бинарная операция «$\cdot$» не обладает свойством коммутативности».

Оценка сложности данного алгоритма равна $O(n^2)$.

Алгоритм 3 – Проверка бинарной операции «·» на ассоциативность

Вход: Матрица $A = (a_{ij})$ бинарной операции «·» размерности $n \times n$, список элементов l множества X над операцией «·».

Выход: «Бинарная операция «·» обладает свойством ассоциативности» или «Бинарная операция «·» не обладает свойством ассоциативности».

Шаг 1. Инициализировать булеву переменную $is\_associative = true$.

Шаг 2. Создать цикл с индексом i ($0 \leq i < n$), в котором создать вложенный цикл с индексом j ($0 \leq j < n$). Создав еще один вложенный цикл с индексом k ($0 \leq k < n$), чтобы проверить следующее условие. Если хотя бы один элемент $a_{ip} \neq a_{rk}$, где $p = a_{jk}$ и $r = a_{ij}$ – индексы элементов $l[p]$ и $l[r]$ соответственно, то присвоить $is\_associative = false$.

Шаг 3. Если после завершения цикла $is\_associative = true$, то вывести в консоль следующее сообщение: «Бинарная операция «·» обладает свойством ассоциативности». Если $is\_associative = false$ – вывести сообщение: «Бинарная операция «·» не обладает свойством ассоциативности».

Оценка сложности данного алгоритма равна $O(n^3)$.

Алгоритм 4 – Проверка бинарной операции «·» на обратимость *Вход*: Матрица $A = (a_{ij})$ бинарной операции «·» размерности $n \times n$, список элементов l множества X над операцией «·».

Выход: «Бинарная операция «·» обладает свойством обратимости» или «Бинарная операция «·» не обладает свойством обратимости».

Шаг 1. Инициализировать булеву переменную $is\_invertible = true$.

Шаг 2. Создать цикл с индексом i ($0 \leq i < n$), в котором создать вложенный цикл с индексом j ($0 \leq j < n$), чтобы пройти по всем элементам a_{ij}. Если условие $a_{ij} = a_{ji} = 1$ не выполняется, то присвоить $is\_invertible = false$.

Шаг 3. Если после завершения цикла $is\_invertible = true$, то вывести в консоль следующее сообщение: «Бинарная операция «·» обладает свойством обратимости». Если $is\_invertible = false$ – вывести сообщение: «Бинарная операция «·» не обладает свойством обратимости».

Оценка сложности данного алгоритма равна $O(n^2)$.

Алгоритм 5 – Проверка бинарной операции «$\cdot$» на дистрибутивность *Вход*: Матрица $A = (a_{ij})$ бинарной операции «$\cdot$» размерности $n \times n$, матрица $A = (a_{ij})$ бинарной операции «$+$» размерности $n \times n$, список элементов l множества X над операцией «$\cdot$» и «$+$».

Выход: «Бинарная операция «$\cdot$» обладает свойством дистрибутивности» или «Бинарная операция «$\cdot$» не обладает свойством дистрибутивности».

Шаг 1. Инициализировать булеву переменную $is\_distributive = true$.

Шаг 2. Создать цикл с индексом i ($0 \leq i < n$), в котором создать вложенный цикл с индексом j ($0 \leq j < n$). Создав еще один вложенный цикл с индексом k ($0 \leq k < n$), чтобы проверить следующее условие. Если хотя бы один элемент $a_{ip} \neq b_{rq}$, где $p = b_{jk}$, $r = a_{ij}$ и $q = a_{ik}$ – индексы элементов $l[p]$, $l[r]$, $l[q]$ соответственно или $a_{si} \neq b_{uv}$, где $s = b_{jk}$, $u = a_{ji}$ и $v = a_{ki}$ – индексы элементов $l[s]$, $l[u]$, $l[v]$ соответственно, то присвоить $is\_distributive = false$.

Шаг 3. Если после завершения цикла $is\_distributive = true$, то вывести в консоль следующее сообщение: «Бинарная операция «$\cdot$» обладает свойством дистрибутивности». Если $is\_distributive = false$ – вывести сообщение: «Бинарная операция «$\cdot$» не обладает свойством дистрибутивности».

Оценка сложности данного алгоритма равна $O(n^3)$.

3.2 Описание алгоритмов выполнения операций над бинарными отношениями

Алгоритм 6 – Построение выполнения операции объединения бинарных отноше

Вход: Бинарное отношение ρ и бинарное отношение σ.

Выход: Бинарное отношение $\rho \cup \sigma$

Шаг 1. Инициализировать список $union\_list$ = [].

Шаг 2. Создать цикл, в котором необходимо пройти по элементам $p \in \rho$. Если текущий элемент $p \notin \sigma$, то добавить элемент p в список $union\_list$.

Шаг 3. Вывести в консоль список $union\_list$.

Оценка сложности данного алгоритма равна $O(n^2)$.

Алгоритм 7 – Построение выполнения операции пересечения бинарных отношен

Вход: Бинарное отношение ρ и бинарное отношение σ.

Выход: Бинарное отношение $\rho \cap \sigma$

Шаг 1. Инициализировать список $intersec\_list$ = [].

Шаг 2. Создать цикл, в котором необходимо пройти по элементам $p \in \rho$. Если текущий элемент $p \in \rho \wedge p \in \sigma$, то добавить элемент p в список $intersec\_list$.

Шаг 3. Вывести в консоль список $intersec\_list$.

Оценка сложности данного алгоритма равна $O(n^2)$.

Алгоритм 8 – Построение выполнения обратной операции бинарных отношений

Вход: Бинарное отношение ρ и бинарное отношение σ.

Выход: Бинарное отношение ρ^{-1}

Шаг 1. Инициализировать список rev_list = [].

Шаг 2. Создать цикл, в котором необходимо пройти по элементам $p \in \rho$. Каждый элемент p есть пара чисел (x, y). Добавить элемент $p = (y, x)$ в список rev_list.

Шаг 3. Вывести в консоль список rev_list.

Оценка сложности данного алгоритма равна $O(n)$.

Алгоритм 9 – Построение выполнения операции композиции бинарных отношен

Вход: Бинарное отношение ρ и бинарное отношение σ.

Выход: Бинарное отношение $\rho \circ \sigma$

Шаг 1. Инициализировать список $comp\_list$ = [].

Шаг 2. Создать цикл, в котором необходимо пройти по элементам $p \in \rho$. А также создать вложеный цикл для прохождения по элеменентам $s \in \sigma$. Так как $p = (x_1, y_1)$ и $s = (x_2, y_2)$, то нужно проверить следующее условие. Если элементы $y_1 = x_2$, то добавить в список $comp\_list$ пару чисел (x_1, y_2).

Шаг 3. Вывести в консоль список $comp\_list$.

Оценка сложности данного алгоритма равна $O(n^2)$.

3.3 Описание алгоритмов выполнения операций над матрицами

Алгоритм 10 – Построение операции сложения двух матриц

Вход: Матрица $A = (a_{ij})$ размерности $n \times m$, матрица $B = (b_{ij})$ размерности

$n \times m$.

Выход: Матрица $C = (c_{ij})$ размерности $n \times m$, полученная из суммы $A + B$.

Шаг 1. Элементы c_{ij} матрицы C находятся как $c_{ij} = a_{ij} + b_{ij}$, где $0 \le i < n$, $0 \le j < m$.

Шаг 2. Вернуть матрицу C в качестве выхода функции.

Оценка сложности данного алгоритма равна $O(n \cdot m)$.

Алгоритм 11 – Построение операции вычитания двух матриц

Вход: Матрица $A = (a_{ij})$ размерности $n \times m$, матрица $B = (b_{ij})$ размерности $n \times m$.

Выход: Матрица $C = (c_{ij})$ размерности $n \times m$, полученная из разности $A - B$.

Шаг 1. Элементы c_{ij} матрицы C находятся как $c_{ij} = a_{ij} - b_{ij}$, где $0 \le i < n$, $0 \le j < m$.

Шаг 2. Вернуть матрицу C в качестве выхода функции.

Оценка сложности данного алгоритма равна $O(n \cdot m)$.

Алгоритм 12 – Построение операции умножения матрицы на число

Вход: Матрица $A = (a_{ij})$ размерности $n \times m$, α – некоторое фиксированное число.

Выход: Матрица $\alpha A = (\alpha a_{ij})$ размерности $n \times m$, полученная из умножения матрицы A На число α.

Шаг 1. Необходимо пройти по всем элементам a_{ij} матрицы A, $0 \le i < n$, $0 \le j < m$. Тогда умножив каждый элемент a_{ij} на число α, то получим матрицу αA.

Шаг 2. Вернуть матрицу αA в качестве выхода функции.

Оценка сложности данного алгоритма равна $O(n \cdot m)$.

Алгоритм 13 – Построение операции произведения двух матриц

Вход: Матрица $A = (a_{ij})$ размерности $n \times l$, матрица $B = (b_{ij})$ размерности $l \times m$.

Выход: Матрица $C = (c_{ij})$ размерности $n \times m$, полученная из произведения матриц A и B.

Шаг 1. Элементы c_{ij} матрицы C находятся следующим образом:

$$c_{ij} = \sum_{k=0}^{l} a_{ik} \cdot b_{kj}, \text{ где } 0 \leq i < n, 0 \leq j < m.$$

Шаг 2. Вернуть матрицу C в качестве выхода функции.

Оценка сложности данного алгоритма равна $O(n \cdot m \cdot l)$.

Алгоритм 14 – Построение операции транспонирования матрицы

Вход: Матрица $A = (a_{ij})$ размерности $n \times m$.

Выход: Матрица $A^T = (a_{ij}^T)$ размерности $n \times m$, полученная путем транспонирования матрицы A.

Шаг 1. Элементы a_{ij}^T матрицы A^T находятся так что: $a_{ij}^T = a_{ji}$, где $0 \leq i < n$, $0 \leq j < m$.

Шаг 2. Вернуть матрицу A^T в качестве выхода функции.

Оценка сложности данного алгоритма равна $O(n \cdot m)$.

Алгоритм 15 – Нахождение обратной матрицы

Вход: Матрица $A = (a_{ij})$ размерности $n \times m$.

Выход: Матрица $A^{-1} = (a_{ij}^{-1})$ размерности $n \times m$, полученная путем транспонирования матрицы A.

Шаг 1. Необходимо найти определитель $|A|$.

Шаг 2. Запустив алгоритм 14, найти матрицу A^T.

Шаг 3. Посчитать матрицу алгебраических дополнений $A^{'}$. Пройдя двумя циклами по $0 \leq i < n$ и $0 \leq j < m$, нужно рассмотреть такие a_{ij}^T, которые не стоят на i-й строке и j-м столбце текущей итерации. Далее считается определитель из рассматриваемых элементов. Значение полученного определителя является элементом $a_{ij}^{'}$ матрицы $A^{'}$.

Шаг 4. Используя алгоритм 12, необходимо умножить матрицу $A^{'}$ на число, посчитанное на шаге 1. Таким образом, получим матрицу $A^{-1} = (a_{ij}^{-1})$.

Шаг 5. Вернуть матрицу A^{-1} в качестве выхода функции.

Оценка сложности данного алгоритма равна $O(n \cdot m \cdot k)$, где k – количество итераций при нахождении миноров.

3.4 Коды программ, реализующей рассмотренные алгоритмы

```
import numpy as np


# Построение объединение бинарных отношений
def get_union_bin_rel(br1, br2=[]):
  if br2 == []:
    union = br1.copy()
    for pair in br1:
      if pair not in union:
        union.append(pair)
    print('Answer:')
    print_binary_relation(union, len(union))
  else:
    union = br1.copy()
    for pair in br2:
      if pair not in union:
        union.append(pair)
    print('Answer:')
    print_binary_relation(union, len(union))


# Построение пересечения бинарных отношений
def get_intersection_bin_rel(br1, br2=[]):
  if br2 == []:
    intersec = []
    for x1, y1 in br1:
      for x2, y2 in br1:
        if x1 == x2 and y1 == y2:
          intersec.append((x1, y1))
    print('Answer:')
    print_binary_relation(intersec, len(intersec))
  else:
    intersec = []
    for x1, y1 in br1:
      for x2, y2 in br2:
        if x1 == x2 and y1 == y2:
          intersec.append((x1, y1))
    print('Answer:')
    print_binary_relation(intersec, len(intersec))
```

```python
# Построение обратного бинарного отношения
def get_reverse_bin_rel(br):
  rev_br = []
  for x, y in br:
    rev_br.append((y, x))
  print('Answer:')
  print_binary_relation(rev_br, len(rev_br))


# Построение композиции бинарных отношений
def get_composition_bin_rel(br1, br2=[]):
  if br2 == []:
    comp_br = []
    for x1, y1 in br1:
      for x2, y2 in br1:
        if y1 == x2:
          comp_br.append((x1, y2))
    print('Answer:')
    print_binary_relation(comp_br, len(comp_br))
  else:
    comp_br = []
    for x1, y1 in br1:
      for x2, y2 in br2:
        if y1 == x2:
          comp_br.append((x1, y2))
    print('Answer:')
    print_binary_relation(comp_br, len(comp_br))


# Выполнение операции сложения матриц
def get_addition_operation(a, n):
  print('Enter values of another matrix')
  b = []
  for i in range(n):
    tmp = input().split()
    for j in range(len(tmp)):
      try:
        tmp[j] = float(tmp[j])
      except:
        k = 0
```

```
      else:
        tmp[j] = float(tmp[j])
    b.append(tmp)
  c = []
  for i in range(n):
    tmp = []
    for j in range(len(a[i])):
      tmp.append(a[i][j] + b[i][j])
    c.append(tmp)

  return c


# Выполнение операции вычитания матриц
def get_subtraction_operation(a, n):
  print('Enter values of another matrix')
  b = []
  for i in range(n):
    tmp = input().split()
    for j in range(len(tmp)):
      try:
        tmp[j] = float(tmp[j])
      except:
        k = 0
      else:
        tmp[j] = float(tmp[j])
    b.append(tmp)
  c = []
  for i in range(n):
    tmp = []
    for j in range(len(a[i])):
      tmp.append(a[i][j] - b[i][j])
    c.append(tmp)

  return c


# Выполнение операции умножение матрицы на число
def get_multiplication_matrix_on_number(a):
  print('Enter number:')
  num = float(input())
```

```
  c = []
  for i in range(len(a)):
    tmp = []
    for j in range(len(a[i])):
      tmp.append(a[i][j] * num)
    c.append(tmp)

  return c


# Выполнение операции умножения матриц
def get_multiplication_operation(a):
  print('Enter the number of rows')
  n = int(input())
  print('Enter the number of columns')
  m = int(input())
  print('Enter values of another matrix')
  b = []
  for i in range(n):
    tmp = input().split()
    for j in range(m):
      try:
        tmp[j] = float(tmp[j])
      except:
        k = 0
      else:
        tmp[j] = float(tmp[j])
    b.append(tmp)
  c = []
  for i in range(m):
    tmp = []
    for j in range(m):
      sum = 0
      for k in range(n):
        sum += a[i][k] * b[k][j]
      tmp.append(sum)
    c.append(tmp)

  return c
```

```
# Выполнение операции транспонирования матрицы
def get_transpose_operation(a):
  c = []
  for i in range(len(a)):
    tmp = []
    for j in range(len(a[i])):
      tmp.append(a[j][i])
    c.append(tmp)

  return c


# Нахождение обратной матрицы
def get_inverse_matrix(a):
  a = np.array(a)
  return np.linalg.inv(a)


# Вывод матрицы
def print_matrix(a, n):
    cnt = 0
    print('Your matrix:')
    for i in a:
        if (cnt < n):
            print(f'{i}' + ' ')
        else:
            print(f'{i}' + '\n')
            cnt = 0
        cnt += 1


# Вывод бинарного отношения
def print_binary_relation(br, n):
    print('{ ', end='')
    for i in range(n):
        if i == n - 1:
            print('(' + str(br[i][0]) + ', ' + str(br[i][1]), end=')')
        else:
            print('(' + str(br[i][0]) + ', ' + str(br[i][1]), end='), ')
    print(' }')
```

```python
# Проверка операции на идемпотентность
def check_idempotence(set_list, a):
  is_idempotent = True
  for i in range(len(set_list)):
    if a[i][i] != set_list[i]:
      is_idempotent = False
      break
  if is_idempotent:
    print('Binary operation is idempotent')
  else:
    print('Binary operation is not idempotent')


# Проверка операции на коммутативность
def check_commutative(set_list, a):
  is_commutative = True
  n = len(set_list)
  for i in range(n):
    for j in range(n):
      if a[i][j] != a[j][i]:
        is_commutative = False
        break
  if is_commutative:
    print('Binary operation is commutative')
  else:
    print('Binary operation is not commutative')


# Проверка операции на ассоциативность
def check_associative(set_list, a):
  is_associative = True
  n = len(set_list)
  for i in range(n):
    for j in range(n):
      for k in range(n):
        if a[i][set_list.index(str(a[j][k]))] != \
          a[set_list.index(str(a[i][j]))][k]:
          is_associative = False
          break
  if is_associative:
```

```
    print('Binary operation is associative')
  else:
    print('Binary operation is not associative')


# Проверка операции на обратимость
def check_invertibility(set_list, a):
  is_invertible = True
  n = len(set_list)
  for i in range(n):
    for j in range(n):
      if not (a[i][j] == a[j][i] == '1'):
        is_invertible = False
        break
  if is_invertible:
    print('Binary operation is invertible')
  else:
    print('Binary operation is not invertible')


# Проверка операции на дистрибутивность
def check_distributivity(set_list, a):
  print('Enter matrix values')
  b = []
  for i in range(len(set_list)):
    b.append([j for j in input().split()])

  is_distributive = True
  n = len(set_list)
  for i in range(n):
    for j in range(n):
      for k in range(n):
        if (a[i][set_list.index(b[j][k])] != \
           b[set_list.index(a[i][j])][set_list.index(a[i][k])]) \
           or (a[set_list.index(b[j][k])][i] != \
           b[set_list.index(a[j][i])][set_list.index(a[k][i])]):
          is_invertible = False
          break
  if is_distributive:
    print('Binary operation is distributive')
  else:
```

```
    print('Binary operation is not distributive')


# Проверка свойств (меню)
def check_properties_mode(set_list, a):
    print_matrix(a, len(a))
    print('Press 1 to check idempotence property')
    print('Press 2 to check commutative property')
    print('Press 3 to check associative property')
    print('Press 4 to check invertibility property')
    print('Press 5 to check distributivity property')
    print('Press 6 to exit')
    bl = input()
    if bl == '1':
      check_idempotence(set_list, a)
      check_properties_mode(set_list, a)
    elif bl == '2':
      check_commutative(set_list, a)
      check_properties_mode(set_list, a)
    elif bl == '3':
      check_associative(set_list, a)
      check_properties_mode(set_list, a)
    elif bl == '4':
      check_invertibility(set_list, a)
      check_properties_mode(set_list, a)
    elif bl == '5':
      check_distributivity(set_list, a)
      check_properties_mode(set_list, a)
    else:
      return choose_mode()


# Проверка свойств бинарных отношений
def check_all_bin_rel_properties(br1, br2=[]):
  if br2 == []:
    print('Your binary relation:')
    print_binary_relation(br1, len(br1))
    print('Choose operation:')
    print('Press 1 to get union of binary relation')
    print('Press 2 to get intersection of binary relation')
    print('Press 3 to get reverse binary relation')
```

```
    print('Press 4 to get composition of binary relation')
    bl = input()
    if bl == '1':
      get_union_bin_rel(br1)
      check_all_bin_rel_properties(br1)
    elif bl == '2':
      get_intersection_bin_rel(br1)
      check_all_bin_rel_properties(br1)
    elif bl == '3':
      get_reverse_bin_rel(br1)
      check_all_bin_rel_properties(br1)
    elif bl == '4':
      get_composition_bin_rel(br1)
      check_all_bin_rel_properties(br1)
    else:
      return choose_mode()
  else:
    print('Your binary relations:')
    print('First:', end='')
    print_binary_relation(br1, len(br1))
    print('Second', end='')
    print_binary_relation(br2, len(br2))
    print('Choose operation:')
    print('Press 1 to get union of binary relations')
    print('Press 2 to get intersection of binary relations')
    print('Press 3 to get reverse binary relations')
    print('Press 4 to get composition of binary relations')
    bl = input()
    if bl == '1':
      get_union_bin_rel(br1, br2)
      check_all_bin_rel_properties(br1, br2)
    elif bl == '2':
      get_intersection_bin_rel(br1, br2)
      check_all_bin_rel_properties(br1, br2)
    elif bl == '3':
      print('First:', end='')
      get_reverse_bin_rel(br1)
      print('Second', end='')
      get_reverse_bin_rel(br2)
      check_all_bin_rel_properties(br1, br2)
    elif bl == '4':
```

```python
        get_composition_bin_rel(br1, br2)
        check_all_bin_rel_properties(br1, br2)
    else:
        print('Exit...')
        return choose_mode()


# Построение бинарных отношений
def construction_of_binary_relation():
    print('Enter numbers of binary relation:')
    s = input()
    tmp = [i for i in s.split(' ')]
    if len(tmp) % 2 != 0:
        print('Incorrect input!')
        return choose_mode()
    else:
        br1 = []
        k = 0
        for i in range(1, len(tmp)):
            if i % 2 != 0:
                br1.append((tmp[i - 1], tmp[i]))
                k += 1
        print('Do you want to enter other binary relation? (0 - no, 1 - yes):')
        bl = input()
        print('Enter numbers of binary relation:')
        if bl == '1':
            s = input()
            tmp = [i for i in s.split(' ')]
            if len(tmp) % 2 != 0:
                print('Incorrect input!')
                choose_mode()
            else:
                br2 = []
                k = 0
                for i in range(1, len(tmp)):
                    if i % 2 != 0:
                        br2.append((tmp[i - 1], tmp[i]))
                        k += 1
                check_all_bin_rel_properties(br1, br2)
        elif bl == '0':
            check_all_bin_rel_properties(br1)
```

```python
    return choose_mode()


# Построение матриц
def construction_of_matrix(a):
  print_matrix(a, len(a))
  n = len(a)
  print('Choose operation:')
  print('Press 1 to add another matrix')
  print('Press 2 to subtract another matrix')
  print('Press 3 to multiply this matrix on number')
  print('Press 4 to multiply on another matrix')
  print('Press 5 to transpose this matrix')
  print('Press 6 to find inverse matrix')
  bl = input()
  if bl == '1':
    a = get_addition_operation(a, n)
    construction_of_matrix(a)
  elif bl == '2':
    a = get_subtraction_operation(a, n)
    construction_of_matrix(a)
  elif bl == '3':
    a = get_multiplication_matrix_on_number(a)
    construction_of_matrix(a)
  elif bl == '4':
    a = get_multiplication_operation(a)
    construction_of_matrix(a)
  elif bl == '5':
    a = get_transpose_operation(a)
    construction_of_matrix(a)
  elif bl == '6':
    a = get_inverse_matrix(a)
    construction_of_matrix(a)
  else:
    return choose_mode()


# Главное меню
def choose_mode():
    print('Choose mode:')
```

```
print('Press 1 to check properties')
print('Press 2 to execute operations for binary relations')
print('Press 3 to execute operations for matrix')
print('Press 4 to exit')
bl = input()
if bl == '1':
    print('Enter numbers of set:')
    s = input()
    set_list = [i for i in s.split(' ')]
    print('Enter matrix values')
    a = []
    for i in range(len(set_list)):
      a.append([j for j in input().split()])
    check_properties_mode(set_list, a)
elif bl == '2':
    construction_of_binary_relation()
elif bl == '3':
    a = []
    print('Enter the number of rows')
    n = int(input())
    print('Enter the number of columns')
    m = int(input())
    print('Enter the matrix values')
    for i in range(n):
      tmp = input().split()
      for j in range(m):
          try:
            tmp[j] = float(tmp[j])
          except:
            k = 0
          else:
            tmp[j] = float(tmp[j])
      a.append(tmp)
    construction_of_matrix(a)
elif bl == '4':
    return
else:
    print('Incorrect output')
    return choose_mode()
```

```
choose_mode()
```

3.5 Результаты тестирования программ

На рисунке 1 показано выполнение задания 1.

```
Choose mode:
Press 1 to check properties
Press 2 to execute operations for binary relations
Press 3 to execute operations for matrix
Press 4 to exit
1
Enter numbers of set:
a b c d
Enter matrix values
a b a b
a b a b
a b c d
a b c d
Your matrix:
['a', 'b', 'a', 'b']
['a', 'b', 'a', 'b']
['a', 'b', 'c', 'd']
['a', 'b', 'c', 'd']
Press 1 to check idempotence property
Press 2 to check commutative property
Press 3 to check associative property
Press 4 to check invertibility property
Press 5 to check distributivity property
Press 6 to exit
3
Binary operation is associative
Your matrix:
['a', 'b', 'a', 'b']
['a', 'b', 'a', 'b']
['a', 'b', 'c', 'd']
['a', 'b', 'c', 'd']
Press 1 to check idempotence property
Press 2 to check commutative property
Press 3 to check associative property
Press 4 to check invertibility property
Press 5 to check distributivity property
Press 6 to exit
```

Рисунок 1 – Тест алгоритма проверки свойств операции «·»

На рисунках 2-5 показано выполнение задания 2. На рисунке 2 изображено нахождение матрицы A^2. На рисунке 3 изображено нахождение матрицы $(10 - \lambda/2) \cdot A$. На рисунке 4 показано нахождение матрицы $\lambda/2 \cdot E$, где E – едининая матрица второго порядка, $\lambda = 6$.

```
Choose mode:
Press 1 to check properties
Press 2 to execute operations for binary relations
Press 3 to execute operations for matrix
Press 4 to exit
3
Enter the number of rows
2
Enter the number of columns
2
Enter the matrix values
1 -2
-3 6
Your matrix:
[1.0, -2.0]
[-3.0, 6.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
4
Enter the number of rows
2
Enter the number of columns
2
Enter values of another matrix
1 -2
-3 6
Your matrix:
[7.0, -14.0]
[-21.0, 42.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
```

Рисунок 2 – Нахождение матрицы A^2

```
Press 6 to find inverse matrix
0
Choose mode:
Press 1 to check properties
Press 2 to execute operations for binary relations
Press 3 to execute operations for matrix
Press 4 to exit
3
Enter the number of rows
2
Enter the number of columns
2
Enter the matrix values
1 -2
-3 6
Your matrix:
[1.0, -2.0]
[-3.0, 6.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
3
Enter number:
7
Your matrix:
[7.0, -14.0]
[-21.0, 42.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
```

Рисунок 3 – Нахождение матрицы $(10 - \lambda/2) \cdot A$

```
Press 6 to find inverse matrix
0
Choose mode:
Press 1 to check properties
Press 2 to execute operations for binary relations
Press 3 to execute operations for matrix
Press 4 to exit
3
Enter the number of rows
2
Enter the number of columns
2
Enter the matrix values
1 0
0 1
Your matrix:
[1.0, 0.0]
[0.0, 1.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
3
Enter number:
3
Your matrix:
[3.0, 0.0]
[0.0, 3.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
```

Рисунок 4 – Нахождение матрицы $\lambda/2 \cdot E$

```
Your matrix:
[7.0, -14.0]
[-21.0, 42.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
1
Enter values of another matrix
7 -14
-21 42
Your matrix:
[14.0, -28.0]
[-42.0, 84.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
1
Enter values of another matrix
3 0
0 3
Your matrix:
[17.0, -28.0]
[-42.0, 87.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
```

Рисунок 5 – Результат всего выражения

На рисунке 6 показано выполение задание 3.

```
Choose mode:
Press 1 to check properties
Press 2 to execute operations for binary relations
Press 3 to execute operations for matrix
Press 4 to exit
3
Enter the number of rows
2
Enter the number of columns
3
Enter the matrix values
-1 6 3
2 2 6
Your matrix:
[-1.0, 6.0, 3.0]
[2.0, 2.0, 6.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
4
Enter the number of rows
3
Enter the number of columns
2
Enter values of another matrix
-6 2
1 7
-3 6
Your matrix:
[3.0, 58.0]
[-28.0, 54.0]
Choose operation:
Press 1 to add another matrix
Press 2 to subtract another matrix
Press 3 to multiply this matrix on number
Press 4 to multiply on another matrix
Press 5 to transpose this matrix
Press 6 to find inverse matrix
```

Рисунок 6 – Произведение двух матриц

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были рассмотрены теоретические сведения об алгебраической операции и классификации свойств операций, основные операции над бинарными отношениями, а также основные операции над матрицами. Опираясь на изложенную выше теорию, были разработаны алгоритмы проверки свойств операций: ассоциативность, коммутативность, идемпотентность, обратимость, дистрибутивность, алгоритмы выполнения операции над бинарными отношениями, а также алгоритмы выполнения операций над матрицами. Была произведена оценка сложности каждого из построенных алгоритмов. Была реализована программа, написанная на языке Python с использованием библиотеки Numpy для преобразования матриц.