МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

КЛАССИФИКАЦИЯ БИНАРНЫХ ОТНОШЕНИЙ И СИСТЕМЫ ЗАМЫКАНИЙ

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»

студента 3 курса 331 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Бородина Артёма Горовича

Преподаватель аспирант ___________ Р. А. Фарахутдинов

подпись, дата

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

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

Определение. Бинарное отношение $\rho \subset A \times A$ называется:
1. *рефлексивным*, если $(a,\, a) \in \rho$ для всякого $a \in A$;
2. *симметричным*, если $(a,\, b) \in \rho \implies (b,\, a) \in \rho$;
3. *антисимметричным*, если $(a,\, b) \in \rho$ и $(b,\, a) \in \rho \implies a = b$;
4. *транзитивным*, если $(a,\, b) \in \rho$ и $(b,\, c) \in \rho \implies (a,\, c) \in \rho$.

Символом Δ_A обозначается тождественное отношение на множестве A, которое определяется по формуле:

$$\Delta_A = \{\, (a,\, a) \,|\, a \in A \,\}.$$

Тогда бинарное отношение $\rho \subset A \times A$ является:
1. *рефлексивным*, если $\Delta_A \subset \rho$;
2. *симметричным*, если $\rho^{-1} \subset \rho$;
3. *антисимметричным*, если $\rho \cap \rho^{-1} \subset \Delta_A$;
4. *транзитивным*, если $\rho\rho \subset \rho$.

Определение. Бинарное отношение ρ на множестве A называется:
1. *отношением эквивалентности (эквивалентностью)*, если оно рефлексивно, симметрично и транзитивно.
2. *отношением порядка (порядком)*, если оно рефлексивно, антисимметрично и транзитивно.
3. *отношением квазипорядка (квазипорядком)*, если оно рефлексивно и транзитивно.

Определение. Множество Z подмножеств множества A называется *системой замыканий*, если оно замкнуто относительно пересечений, т.е. выполняется:

$$\bigcap B \in Z \quad \forall B \subset Z.$$

В частности, для $\varnothing \subset Z$ выполняется $\cap\varnothing = A \in Z$.

Лемма 1. О системах замыканий бинарных отношений. На множестве $P(A^2)$ всех бинарных отношений между элементами множества A следующие множества являются системами замыканий:

1. Z_r - множество всех рефлексивных бинарных отношений между элементами множества A.

2. Z_s - множество всех симметричных бинарных отношений между элементами множества A.

3. Z_t - множество всех транзитивных бинарных отношений между элементами множества A.

4. $Z_{eq} = Eq(A)$ - множество всех отношений эквивалентности на множестве A.

Множество Z_{as} всех антисимметричных бинарных отношений между элементами множества A не является системой замыканий.

Определение. *Оператором замыкания* на множестве A называется отображение f множества всех подмножеств $P(A)$ в себя, удовлетворяющее условиям:

1) $X \subset Y \implies f(X) \subset f(Y)$;

2) $X \subset f(X)$;

3) $(f \circ f)(X) = f(X)$

для всех $X, Y \in P(A)$. Для подмножества $X \subset A$ значение $f(X)$ называется *замыканием* подмножества X.

Лемма 2. О замыканиях бинарных отношений. На множестве $P(A^2)$ всех бинарных отношений между элементами множества A следующие отображения являются операторами замыканий:

1) $f_r(\rho) = \varrho \cup \Delta_A$ - наименьшее рефлексивное бинарное отношение, содержащее отношение $\rho \subset A^2$,

2) $f_s(\rho) = \varrho \cup \varrho^{-1}$ - наименьшее симметричное бинарное отношение, содержащее отношение $\rho \subset A^2$,

3) $f_t(\rho) = \bigcup\limits_{n=1}^{\infty} \rho^n$ - наименьшее транзитивное бинарное отношение, содержащее отношение $\rho \subset A^2$,

4) $f_{eq}(\rho) = (f_t \circ f_s \circ f_r)(\rho)$ - наименьшее отношение эквивалентности, содержащее отношение $\rho \subset A^2$.

3 Результаты работы

3.1 Алгоритм определения свойства рефлексивности.

Описание алгоритма определения свойства рефлексивности.

Вход: список смежности бинарного отношения ρ.

Выход: строка «Бинарное отношение является/не является рефлексивным.» и bool значение **true** или **false**.

Метод: для каждого элемента a, находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. В этом списке смежности ищется сам элемент a. Алгоритм прекрщает свою работу, если был найден элемент a, список смежности которого не содержит a, или, если для всякого элемента a его список смежности содержит a.

Псевдокод алгоритма определения свойства рефлексивности.

```
isReflexive(binaryRelation)
{
  for element in binaryRelation
  {
    flag = false

    for subelement in element:
      if (subelement == element):
        flag = true

    if (!flag)
      return false
  }

  return true
}
```

Листинг 1: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства рефлексивности.

```
bool isReflexive(map<int, set<int>> binaryRelation)
{
  bool isReflexive = false;
  set<int> ::iterator it;
  int i;

  for (auto element : binaryRelation)
  {
    isReflexive = false;
    it = element.second.begin();

    for (; it != element.second.end(); ++it)
      if (element.first == *it)
      {
        isReflexive = !isReflexive;
        break;
      }

    if (!isReflexive)
      break;
  }

  cout << "\n" <<
    (isReflexive ? "\nBINARY RELATION IS REFLEXIVE.\n" :
                   "\nBINARY RELATION IS NOT REFLEXIVE.\n");
  return isReflexive;
}
```

Листинг 2: Код программы.

Результат тестирования программы определения свойства рефлексивности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ, для одного из которых свойство рефлексивности выполняется, а для другого нет.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(13; 1). (16; 68). (17; 9). (34; 98). (56; 5). (60; 42). (81; 61). (91; 57).
```

Рисунок 1 – Пары элементов, находящихся в нерефлексивном бинарном отношении.

Как видно, данное отношение не является рефлексивным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(13; 1). (16; 68). (17; 9). (34; 98). (56; 5). (60; 42). (81; 61). (91; 57).

BINARY RELATION IS NOT REFLEXIVE.
```

Рисунок 2 – Проверка на рефлексивность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является рефлексивным. Теперь рассмотрим бинарное отношение δ, для которого свойство рефлексивности выполняется. Рассмотрим 8 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 1), (0; 2). (1; 0), (1; 1), (1; 2). (2; 2).
```

Рисунок 3 – Пары элементов, находящихся в рефлексивном бинарном отношении.

Как видно, отношение является рефлексивным. Посмотрим на выход программы:

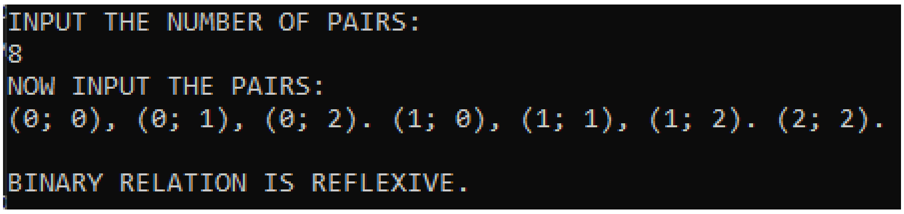


Рисунок 4 – Проверка на рефлексивность пройдена.

Выход программы совпадает с тем фактом, что отношение является рефлексивным.

Оценка временной сложности алгоритма определения свойства рефлексивности.

Будем считать, что алгоритм отрабатывает полностью, а не встречает элемент, который гарантирует невыполнение свойства рефлексивности (в таком случае программа преждевременно завершает своё выполнение). Посчитаем число элементарных операций, выполняемых алгоритмом. Пусть число уникальных элементов, входящих в бинарное отношение с другими элементами b_i, равно n. Определим число n_i как количество элементов в списке смежности i-го по счёту элемента. Тогда *верхняя граница* числа элементарных операций, выполняемых алгоритмом, будет равна:

$$n_1 + n_2 + \dots + n_n = \sum_{i=1}^{n} n_i \leq n^2 \implies T(isReflexive, n) = O(n^2).$$

3.2 Алгоритм определения свойства симметричности.

Описание алгоритма определения свойства симметричности.

Вход: список смежности бинарного отношения ρ.

Выход: строка «Бинарное отношение является/не является симметричным.» и bool значение **true** или **false**.

Метод: для каждого элемента a, находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. Для каждого элемента b_i из списка смежности элемента a просматривается его список смежности, и в нём ищется элемент a. Алгоритм прекращает свою работу, если был найден элемент b_i из списка смежности элемента a, список смежности которого не содержит элемента a, или, если для всякого элемента a списки смежностей элементов b_i содержат a.

Псевдокод алгоритма определения свойства симметричности.

```
isSymmetric(binaryRelation)
{
  for element in binaryRelation
    for subelement in element
    {
      flag = false

      if element in binaryRelation[subelement]
        flag = true

      if (!flag)
        return false
    }

  return true
}
```

Листинг 3: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства симметричности.

```
bool isSymmetric(map<int, set<int>> binaryRelation)
{
  bool isSymmetric = false;
  set<int> ::iterator it, itHelper;

  for (auto element : binaryRelation)
  {
    it = element.second.begin();

    for (; it != element.second.end(); ++it)
    {
      isSymmetric = false;

      if (!binaryRelation[*it].empty())
      {
        itHelper = binaryRelation[*it].begin();

        for (; itHelper != binaryRelation[*it].end(); ++itHelper)
          if (element.first == *itHelper)
          {
            isSymmetric = !isSymmetric;
            break;
          }
      }

      if (!isSymmetric)
      {
        cout << "\nBINARY RELATION IS NOT SYMMETRIC.\n";
        return false;
      }
    }
  }

  cout << "\nBINARY RELATION IS SYMMETRIC.\n";
  return true;
}
```

Листинг 4: Код программы.

Результат тестирования программы определения свойства симметричности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ, для одгого из которых свойство симметричности выполняется, а для другого нет.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(7; 16). (9; 77). (19; 99). (33; 14). (35; 15). (57; 82). (79; 94). (89; 17).
```

Рисунок 5 – Пары элементов, находящихся в несимметричном бинарном отношении.

Как видно, данное отношение не является симметричным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(7; 16). (9; 77). (19; 99). (33; 14). (35; 15). (57; 82). (79; 94). (89; 17).

BINARY RELATION IS NOT SYMMETRIC.
```

Рисунок 6 – Проверка на симметричность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является симметричным. Теперь рассмотрим бинарное отношение δ, для которого свойство симметричности выполняется. Рассмотрим 8 следующих пар (число сгенерированных пар элементов может быть меньше, поскольку некоторые пары при генерации могли совпасть):

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 1), (0; 2). (1; 0), (1; 1). (2; 0).
```

Рисунок 7 – Пары элементов, находящихся в симметричном бинарном отношении.

Как видно, отношение является симметричным. Посмотрим на выход программы:

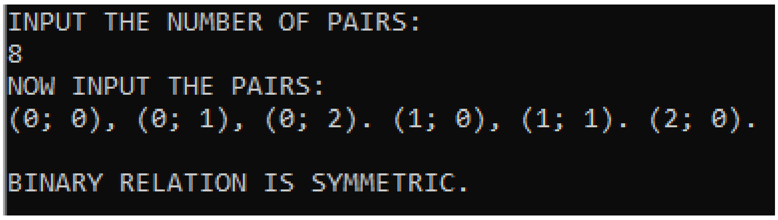


Рисунок 8 – Проверка на симметричность пройдена.

Выход программы совпадает с тем фактом, что отношение является симметричным.

Оценка временной сложности алгоритма определения свойства симметричности.

Будем считать, что алгоритм отрабатывает полностью, а не встречает элемент, который гарантирует невыполнение свойства симметричности (в таком случае программа преждевременно завершает своё выполнение). Посчитаем число элементарных операций, выполняемых алгоритмом. Пусть число уникальных элементов, входящих в бинарное отношение с другими элементами b_i, равно n. Определим число n_i как количество элементов в списке смежности i-го по счёту элемента, а n_{ij} как j-ый по счёту элемент в таком списке смежности. Тогда *верхняя граница* числа элементарных операций, выполняемых алгоритмом, будет равна:

$$n_{n_{11}}+\cdots+n_{n_{1n}}+n_{n_{21}}+\cdots+n_{n_{nn}} \leq n\sum_{i=1}^{n} n_i \leq n^3 \implies T(isSymmetric,\, n) = O(n^3).$$

3.3 Алгоритм определения свойства антисимметричности.

Описание алгоритма определения свойства антисимметричности. Вход: список смежности бинарного отношения ρ.

Выход: строка «Бинарное отношение является/не является антисимметричным.» и bool значение **true** или **false**.

Метод: для каждого элемента a, находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. Для каждого элемента b_i из списка смежности элемента a просматривается его список смежности. В этом списке смежности не должно содержаться элемента a, за исключением ситуации, когда a и b_i - один и тот же элемент, и переход из a в b_i - петля.

Псевдокод алгоритма определения свойства антисимметричности.

```
isAntisymmetric(binaryRelation)
{
  for element in binaryRelation:
    for subelement in element
    {
      isAntisymmetric = true

      if subelement is not element
      {
        for elt in binaryRelation[subelement]
          if (elt == element)
            isAntisymmetric = false
      }

      if (!isAntisymmetric)
        return false
    }

  return true
}
```

Листинг 5: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства антисимметричности.

```
bool isAntisymmetric(map<int, set<int>> binaryRelation)
{
  bool isAntisymmetric = false;
  set<int> ::iterator it, itHelper;

  for (auto element : binaryRelation)
  {
    it = element.second.begin();

    for (; it != element.second.end(); ++it)
    {
      isAntisymmetric = true;

      if (!binaryRelation[*it].empty() && *it != element.first)
      {
        itHelper = binaryRelation[*it].begin();

        for (; itHelper != binaryRelation[*it].end(); ++itHelper)
          if (*itHelper == element.first)
            isAntisymmetric = false;
      }

      if (!isAntisymmetric)
      {
        cout << "\nBINARY RELATION IS NOT ANTISYMMETRIC.\n";
        return false;
      }
    }
  }

  cout << "\nBINARY RELATION IS ANTISYMMETRIC.\n";
  return true;
}
```

Листинг 6: Код программы.

Результат тестирования программы определения свойства антисимметричности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ, для одгого из которых свойство антисимметричности выполняется, а для другого нет.

Сгенерируем 8 пар элементов (число сгенерированных пар элементов может быть меньше, поскольку некоторые пары при генерации могли совпасть), соответствующих бинарному отношению ρ. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 2). (1; 1). (2; 0), (2; 2).
```

Рисунок 9 – Пары элементов, находящихся в неантисимметричном бинарном отношении.

Как видно, данное отношение не является антисимметричным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(0; 0), (0; 2). (1; 1). (2; 0), (2; 2).

BINARY RELATION IS NOT ANTISYMMETRIC.
```

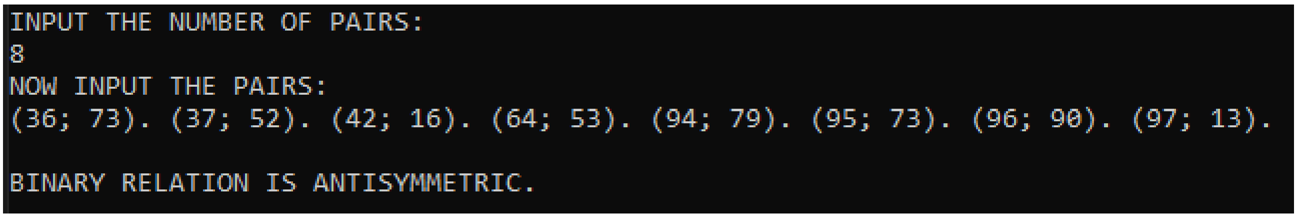
Рисунок 10 – Проверка на антисимметричность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является антисимметричным. Теперь рассмотрим бинарное отношение δ, для которого свойство антисимметричности выполняется. Рассмотрим 8 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(36; 73). (37; 52). (42; 16). (64; 53). (94; 79). (95; 73). (96; 90). (97; 13).
```

Рисунок 11 – Пары элементов, находящихся в симметричном бинарном отношении.

Как видно, отношение является антисимметричным. Посмотрим на выход программы:



```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(36; 73). (37; 52). (42; 16). (64; 53). (94; 79). (95; 73). (96; 90). (97; 13).

BINARY RELATION IS ANTISYMMETRIC.
```

Рисунок 12 – Проверка на симметричность пройдена.

Выход программы совпадает с тем фактом, что отношение является антисимметричным.

Оценка временной сложности алгоритма определения свойства антисимметричности.

Будем считать, что алгоритм отрабатывает полностью, а не встречает элемент, который гарантирует невыполнение свойства антисимметричности (в таком случае программа преждевременно завершает своё выполнение). Поскольку принцип работы этого алгоритма аналогичен тому, что используется в алгоритм определения свойства симметричности, то для этого алгоритма выполняется оценка сложности, полученная ранее:

$$n_{n_{11}} + \cdots + n_{n_{1n}} + n_{n_{21}} + \cdots + n_{n_{nn}} \leq n \sum_{i=1}^{n} n_i \leq n^3 \implies T(isAntisymmetric, n) = O(n^3).$$

3.4 Алгоритм определения свойства транзитивности.

Описание алгоритма определения свойства транзитивности.
Вход: список смежности бинарного отношения ρ.

Выход: строка «Бинарное отношение является/не является транзитивным.» и bool значение **true** или **false**.

Метод: для каждого элемента a, находящегося в бинарном отношении ρ (с некоторыми элементами b_i), просматривается его список смежности. Для каждого элемента b_i из списка смежности элемента a просматривается его список смежности. Для удовлетворения условия транзитивности каждый элемент из списка смежности b_i должен содержаться в списке смежности элемента a. Алгоритм завершает свою работу, если находится такой элемент, что это условие для него не выполняется, или после проверки и выполнения этого условия для всякого элемента a.

Псевдокод алгоритма определения свойства транзитивности.

```
isTransitive(binaryRelation)
{
  for element in binaryRelation:
    for subelement in element:
      for elt in binaryRelation[subelement]
      {
        isTransitive = false

        for eltSup in element:
          if (elt == eltSup)
            isTransitive = true

        if (!isTransitive)
          return false
      }

  return true
}
```

Листинг 7: Псевдокод алгоритма.

Код программы, реализующей алгоритм определения свойства транзитивности.

```
bool isTransitive(map<int, set<int>> binaryRelation)
{
  bool isTransitive = false;
  set<int> ::iterator it, itHelper, itSup;

  for (auto element : binaryRelation)
  {
    it = element.second.begin();

    for (; it != element.second.end(); ++it)
    {
      if (!binaryRelation[*it].empty())
      {
        itHelper = binaryRelation[*it].begin();

        for (; itHelper != binaryRelation[*it].end(); ++itHelper)
        {
          itSup = element.second.begin();
          isTransitive = false;

          for (; itSup != element.second.end(); ++itSup)
            if (*itHelper == *itSup)
              isTransitive = !isTransitive;

          if (!isTransitive)
          {
            cout << "\nBINARY RELATION IS NOT TRANSITIVE.\n";
            return false;
          }
        }
      }
    }
  }
  cout << "\nBINARY RELATION IS TRANSITIVE.\n";
  return true;
}
```

Листинг 8: Код программы.

Результат тестирования программы определения свойства транзитивности.

Для демонстрации работы программы рассмотрим два произвольных бинарных отношения ρ и δ, для одгого из которых свойство транзитивности выполняется, а для другого нет.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 67). (17; 14), (17; 16). (18; 54). (19; 88). (26; 90). (90; 24). (92; 97).
```

Рисунок 13 – Пары элементов, находящихся в нетранзитивном бинарном отношении.

Как видно, данное отношение не является транзитивным. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 67). (17; 14), (17; 16). (18; 54). (19; 88). (26; 90). (90; 24). (92; 97).

BINARY RELATION IS NOT TRANSITIVE.
```

Рисунок 14 – Проверка на транзитивность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является транзитивным. Теперь рассмотрим бинарное отношение δ, для которого свойство транзитивности выполняется. Перейдём на ручной ввод и введём 8 пар элементов:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
1 1 1 2 2 3 1 3 4 5 5 6 4 6 2 2

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 2), (2; 3). (4; 5), (4; 6). (5; 6).
```

Рисунок 15 – Пары элементов, находящихся в транзитивном бинарном отношении.

Как видно, отношение является транзитивным. Посмотрим на выход программы:

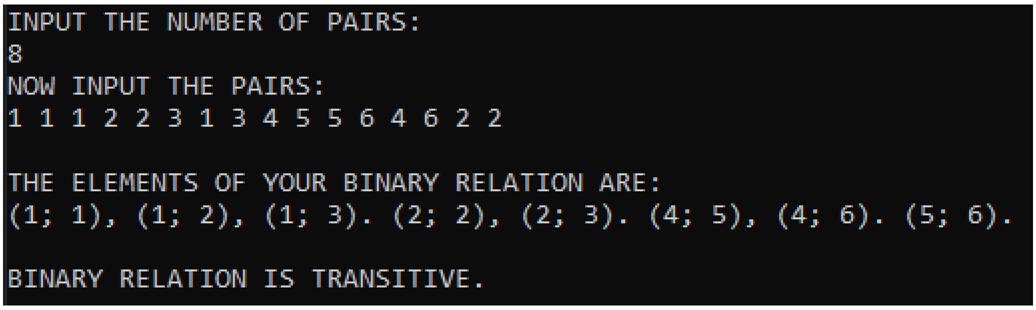


Рисунок 16 – Проверка на транзитивность пройдена.

Выход программы совпадает с тем фактом, что отношение является транзитивным.

Оценка временной сложности алгоритма определения свойства транзитивности.

Будем считать, что алгоритм отрабатывает полностью, а не встречает элемент, который гарантирует невыполнение свойства транзитивности (в таком случае программа преждевременно завершает своё выполнение). Определим n как число уникальных элементов, встречающихся в бинарном отношении, n_i как число элементов в списке смежности элемента с i-ым номером, а n_{ij} как j-ый элемент в таком списке смежности. Тогда число элементарных операций для одного элемента (например, первого) может быть вычислено как:

$$n_1 n_{n_{11}} + n_1 n_{n_{12}} + \cdots + n_1 n_{n_{1n}} = n_1 \sum_{i=1}^{n} n_{n_{1i}}.$$

А для n элементов:

$$n_1 \sum_{i=1}^{n} n_{n_{1i}} + n_2 \sum_{i=1}^{n} n_{n_{2i}} + \cdots + n_n \sum_{i=1}^{n} n_{n_{ni}} = \sum_{j=1}^{n} (n_j \sum_{i=1}^{n} n_{n_{ji}}),$$

$$\sum_{i=1}^{n} n_{n_{ji}} \le \sum_{i=1}^{n} n_i \le n^2 \implies \sum_{j=1}^{n} (n_j \sum_{i=1}^{n} n_{n_{ji}}) \le n^2 \sum_{i=1}^{n} n_i \le n^4.$$

Получаем, что $T(isTransitive, n) = O(n^4)$.

3.5 Алгоритм классификации бинарных отношений.

Отдельно рассмотрим три функции, определяющих класс бинарного отношения, а именно: отношение эквивалентности, отношение порядка или отношение квазипорядка. Тогда алгоритм, содержащий в себе эти три функции, может выглядеть следующим образом:

Вход: список смежности бинарного отношения ρ.

Выход: строки «Бинарное отношение является/не является отношением эквивалентности.», «Бинарное отношение является/не является отношением порядка.», «Бинарное отношение является/не является отношением квазипорядка.»

Метод: последовательно запускаем функции проверки принадлежности бинарного отношение к соответствующему классу.

Код программы, реализующей алгоритм классификации бинарных отношений.

```
void isEquivalence(map<int, set<int>> binaryRelation)
{
  if (isReflexive(binaryRelation) &&
      isSymmetric(binaryRelation) &&
      isTransitive(binaryRelation))
    cout << "\nBINARY RELATION IS AN EQUIVALENT RELATION.\n";
  else
    cout << "\nBINARY RELATION IS NOT AN EQUIVALENT RELATION.\n";
}

void isOrder(map<int, set<int>> binaryRelation)
{
  if (isReflexive(binaryRelation) &&
      isAntisymmetric(binaryRelation) &&
      isTransitive(binaryRelation))
    cout << "\nBINARY RELATION IS AN ORDER RELATION.\n";
  else
    cout << "\nBINARY RELATION IS NOT AN ORDER RELATION.\n";
}

void isQuasi(map<int, set<int>> binaryRelation)
```

```
{
  if (isReflexive(binaryRelation) &&
      isTransitive(binaryRelation))
    cout << "\nBINARY RELATION IS A QUASI-ORDER RELATION.\n";
  else
    cout << "\nBINARY RELATION IS NOT A QUASI-ORDER RELATION.\n";
}
```

Листинг 9: Набор функций, осуществляющий классификацию бинарных отношений.

Результат тестирования программы классификации бинарных отношений.

Для демонстрации работы программы рассмотрим шесть произвольных бинарных отношений ρ_i, $i = \overline{1,6}$. В каждой паре одно отношение будет принадлежать какому-нибудь классу, а другое - нет. Начнём с рассмотрения отношения эквивалентности.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ_1. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 37). (23; 69). (29; 82). (31; 78). (56; 33). (60; 88). (66; 58). (91; 77).
```

Рисунок 17 – Пары элементов, находящихся в неэквивалентном бинарном отношении.

Как видно, данное отношение не является эквивалентностью. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(6; 37). (23; 69). (29; 82). (31; 78). (56; 33). (60; 88). (66; 58). (91; 77).

BINARY RELATION IS NOT REFLEXIVE.

BINARY RELATION IS NOT AN EQUIVALENT RELATION.
```

Рисунок 18 – Проверка на эквивалентность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является эквивалентностью. Также программа сообщила, почему данное отношение не является эквивалентностью - оно нерефлексивно. Теперь рассмотрим бинарное отношение ρ_2, которое является эквивалентностью. Перейдём на ручной ввод и введём 9 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
9
NOW INPUT THE PAIRS:
1 1 2 2 3 3 1 2 2 1 2 3 3 2 1 3 3 1

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 1), (3; 2), (3; 3).
```

Рисунок 19 – Пары элементов, находящихся в эквивалентном бинарном отношении.

Как видно, отношение является эквивалентностью. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
9
NOW INPUT THE PAIRS:
1 1 2 2 3 3 1 2 2 1 2 3 3 2 1 3 3 1

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 1), (3; 2), (3; 3).

BINARY RELATION IS REFLEXIVE.

BINARY RELATION IS SYMMETRIC.

BINARY RELATION IS TRANSITIVE.

BINARY RELATION IS AN EQUIVALENT RELATION.
```

Рисунок 20 – Проверка на эквивалентность пройдена.

Выход программы совпадает с тем фактом, что отношение является эквивалентностью, поскольку каждое из необходимых свойств выполняется.

Теперь перейдём к рассмотрению отношения порядка.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ_3. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(8; 38). (39; 34). (41; 84). (43; 42). (46; 54). (64; 49). (73; 30). (86; 86).
```

Рисунок 21 – Пары элементов, не находящихся в бинарном отношении порядка.

Как видно, данное отношение не является порядком. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(8; 38). (39; 34). (41; 84). (43; 42). (46; 54). (64; 49). (73; 30). (86; 86).

BINARY RELATION IS NOT REFLEXIVE.

BINARY RELATION IS NOT AN ORDER RELATION.
```

Рисунок 22 – Проверка на упорядоченность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является отношением порядка. Также программа сообщила, почему данное отношение не является отношением порядка - оно нерефлексивно. Теперь рассмотрим бинарное отношение ρ_4, которое является отношением порядка. Перейдём на ручной ввод и введём 6 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 2 2 3 3 3

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 2), (2; 3). (3; 3).
```

Рисунок 23 – Пары элементов, находящихся в упорядоченном бинарном отношении.

Как видно, отношение является отношением порядка. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 2 2 3 3 3

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 2), (2; 3). (3; 3).

BINARY RELATION IS REFLEXIVE.

BINARY RELATION IS ANTISYMMETRIC.

BINARY RELATION IS TRANSITIVE.

BINARY RELATION IS AN ORDER RELATION.
```

Рисунок 24 – Проверка на упорядоченность пройдена.

Выход программы совпадает с тем фактом, что отношение является порядком, поскольку каждое из необходимых свойств выполняется.

Теперь перейдём к рассмотрению отношения квазипорядка.

Сгенерируем 8 пар элементов, соответствующих бинарному отношению ρ_5. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(11; 36). (20; 91). (23; 44). (45; 81). (63; 50). (74; 34). (81; 36). (82; 71).
```

Рисунок 25 – Пары элементов, не находящихся в бинарном отношении квазипорядка.

Как видно, данное отношение не является квазипорядком. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
8
NOW INPUT THE PAIRS:
(11; 36). (20; 91). (23; 44). (45; 81). (63; 50). (74; 34). (81; 36). (82; 71).

BINARY RELATION IS NOT REFLEXIVE.

BINARY RELATION IS NOT A QUASI-ORDER RELATION.
```

Рисунок 26 – Проверка на упорядоченность не пройдена.

Выход программы совпадает с тем фактом, что отношение не является отношением квазипорядка. Также программа сообщила, почему данное отношение не является отношением квазипорядка - оно нерефлексивно. Теперь рассмотрим бинарное отношение ρ_6, которое является отношением квазипорядка. Перейдём на ручной ввод и введём 7 следующих пар:

```
INPUT THE NUMBER OF PAIRS:
7
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 1 2 2 2 3 3 3

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 3).
```

Рисунок 27 – Пары элементов, находящихся в квазиупорядоченном бинарном отношении.

Как видно, отношение является отношением квазипорядка. Посмотрим на выход программы:

```
INPUT THE NUMBER OF PAIRS:
7
NOW INPUT THE PAIRS:
1 1 1 2 1 3 2 1 2 2 2 3 3 3

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(1; 1), (1; 2), (1; 3). (2; 1), (2; 2), (2; 3). (3; 3).

BINARY RELATION IS REFLEXIVE.

BINARY RELATION IS TRANSITIVE.

BINARY RELATION IS A QUASI-ORDER RELATION.
```

Рисунок 28 – Проверка на квазиупорядоченность пройдена.

Выход программы совпадает с тем фактом, что отношение является квазипорядком, поскольку каждое из необходимых свойств выполняется.

Оценка временной сложности алгоритма классификации бинарных отношений.

Поскольку каждый из алгоритмов классификации использует уже готовые функции, то для них мы сразу можем дать оценку временной сложности:

Для каждого алгоритма классификации бинарного отношения:
$T(isEquivalence, n) = T(isOrder, n) = T(isQuasi, n) = O(n^4)$.

3.6 Алгоритм построения рефлексивного замыкания бинарного отношения.

Описание алгоритма построения рефлексивного замыкания бинарного отношения. Вход: список смежности бинарного отношения ρ.
Выход: замкнутое относительно свойства рефлексивности бинарное отношение ρ.
Метод: для каждого элемента a, находящегося в бинарном отношении ρ (с некоторыми элементами b_i), в его список смежности добавляется элемент a.

Псевдокод алгоритма построения рефлексивного замыкания бинарного отношения.

```
reflexiveClosure(binaryRelation)
{
  for element in binaryRelation
  {
    placeholder = element
    binaryRelation[element].insert(element);
  }

  return binaryRelation;
}
```

Листинг 10: Псевдокод алгоритма.

Код программы, реализующей алгоритм построения рефлексивного замыкания бинарного отношения.

```
map<int, set<int>>
    reflexiveClosure(map<int, set<int>> binaryRelation)
{
  int i, placeholder;
  set<int>::iterator it;

  for (auto element : binaryRelation)
  {
    placeholder = element.first;

    binaryRelation[element.first].insert(element.first);
  }

  return binaryRelation;
}
```

Листинг 11: Код программы.

Результат тестирования программы рефлексивного замыкания бинарного отношения.

Для демонстрации работы программы рассмотрим произвольное отношение δ.

Сгенерируем 3 пары элементов, соответствующих бинарному отношению δ. Получаем следующие пары:

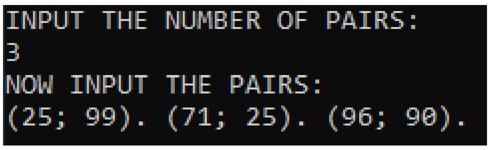


Рисунок 29 – Бинарное отношение, не обладающее свойством рефлексивности.

Как видно, данное отношение не является рефлексивным. Построим рефлексивное замыкание этого бинарного отношения:

```
INPUT THE NUMBER OF PAIRS:
3
NOW INPUT THE PAIRS:
(25; 99). (71; 25). (96; 90).

THE ELEMENTS OF YOUR BINARY RELATION AFTER REFLEXIVE CLOSURE ARE:
(25; 25), (25; 99). (71; 25), (71; 71). (90; 90). (96; 90), (96; 96). (99; 99).

BINARY RELATION IS REFLEXIVE.
```

Рисунок 30 – Бинарное отношение после рефлексивного замыкания.

Как можно видеть, получившееся бинарное отношение является замкнутым относительно свойства рефлексивности, что подтверждает выход программы *isReflexive.*

Оценка временной сложности алгоритма рефлексивного замыкания бинарного отношения.

Определим n как число уникальных элементов, встречающихся в бинарном отношении. Для каждого такого элемента в его список смежности должен быть добавлен он сам. В зависимости от реализации контейнера, являющегося списком смежности, сложность добавления элемента будет отличаться. Мы используем структуру set, асимптотическая сложность добавления элемента в него равна $O(\log n)$. Тогда $T(\mathit{reflexiveClosure},\, n) = O(n \log n)$.

3.7 Алгоритм построения симметричного замыкания бинарного отношения.

Описание алгоритма построения симметричного замыкания бинарного отношения.

Вход: список смежности бинарного отношения ρ.

Выход: замкнутое относительно свойства симметричности бинарное отношение ρ.

Метод: для каждого элемента a, находящегося в бинарном отношении ρ с некоторыми элементами b_i, в список смежности каждого такого элемента b_i добавляется элемент a.

Псевдокод алгоритма построения симметричного замыкания бинарного отношения.

```
symmetricClosure(binaryRelation)
{
  for element in binaryRelation:
    for subelement in element:
      binaryRelation[subelement].insert(element)

  return binaryRelation;
}
```

Листинг 12: Псевдокод алгоритма.

Код программы, реализующей алгоритм построения симметричного замыкания бинарного отношения.

```
map<int, set<int>>
    symmetricClosure(map<int, set<int>> binaryRelation)
{
  set<int> ::iterator it;
  for (auto element : binaryRelation)
  {
    it = element.second.begin();
    for (; it != element.second.end(); ++it)
      binaryRelation[*it].insert(element.first);
  }
  return binaryRelation;
}
```

Листинг 13: Код программы.

Результат тестирования программы симметричного замыкания бинарного отношения.

Для демонстрации работы программы рассмотрим произвольное отношение δ.

Сгенерируем 3 пары элементов, соответствующих бинарному отношению δ. Получаем следующие пары:

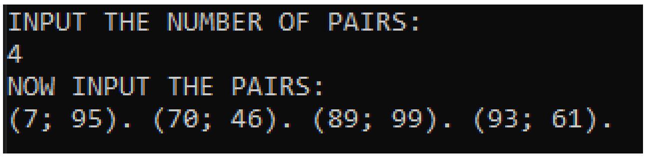
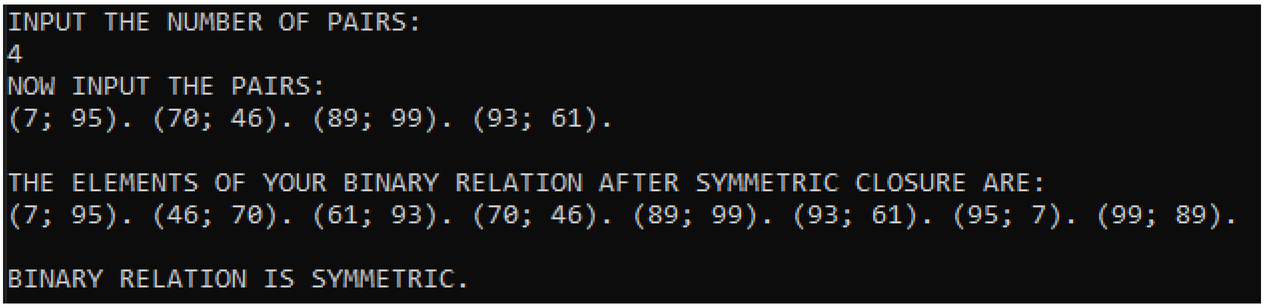


Рисунок 31 – Бинарное отношение, не обладающее свойством симметричности.

Как видно, данное отношение не является симметричным. Построим симметричное замыкание этого бинарного отношения:



```
INPUT THE NUMBER OF PAIRS:
4
NOW INPUT THE PAIRS:
(7; 95). (70; 46). (89; 99). (93; 61).

THE ELEMENTS OF YOUR BINARY RELATION AFTER SYMMETRIC CLOSURE ARE:
(7; 95). (46; 70). (61; 93). (70; 46). (89; 99). (93; 61). (95; 7). (99; 89).

BINARY RELATION IS SYMMETRIC.
```

Рисунок 32 – Бинарное отношение после симметричного замыкания.

Как можно видеть, получившееся бинарное отношение является замкнутым относительно свойства симметричности, что подтверждает выход программы *isSymmetric.*

Оценка временной сложности алгоритма симметричного замыкания бинарного отношения.

Определим n как число уникальных элементов, встречающихся в бинарном отношении, n_i как число элементов в списке смежности i-го элемента, а n_{ij} как j-ый элемент в таком списке смежности. Для каждого элемента n_{ij} в список смежности $n_{n_{ij}}$ должен быть добавлен i-ый элемент. Тогда число элементарных операций для одного элемента (например, первого), составляет: n_1, а временная сложность с учётом стоимости вставки элемента не превышает: $O(n_1 \log n)$. Тогда число элементарных операций для n элементов будет составлять:

$$\log n \sum_{i=1}^{n} n_i \leq n^2 \log n \implies T(symmetricClosure, n) = O(n^2 \log n).$$

3.8 Алгоритм построения транзитивного замыкания бинарного отношения.

Описание алгоритма построения транзитивного замыкания бинарного отношения.

Вход: список смежности бинарного отношения ρ.

Выход: замкнутое относительно свойства транзитивности бинарное отношение ρ.

Метод: для каждого элемента a, находящегося в бинарном отношении ρ с некоторыми элементами b_i, все элементы, находящиеся в списке смежности элемента b_i, должны быть добавлены в список смежности элемента a. Но поскольку после всего одного применения функции *transitiveClosure* получившееся бинарное отношение может оказаться незамкнутым относительно свойства транзитивности, применение функции на ρ надо продолжать до тех пор, пока функция *isTransitive* не вернёт **true**.

Псевдокод алгоритма построения транзитивного замыкания бинарного отношения.

```
transitiveClosureMachinery(binaryRelation)
{
  for element in binaryRelation:
    for subelement in element:
      for elt in binaryRelation[subelement]:
        binaryRelation[element].insert(elt);

  return binaryRelation;
}

transitiveClosure(binaryRelation)
{
    while (!isTransitive(binaryRelation))
        binaryRelation =
          transitiveClosureMachinery(binaryRelation);
    return binaryRelation;
}
```

Листинг 14: Псевдокод алгоритма.

Код программы, реализующей алгоритм построения транзитивного замыкания бинарного отношения.

```
map<int, set<int>>
    transitiveClosure(map<int, set<int>> binaryRelation)
{
  set<int> ::iterator it, itHelper;

  for (auto element : binaryRelation)
  {
    it = element.second.begin();

    for (; it != element.second.end(); ++it)
    {
      itHelper = binaryRelation[*it].begin();

      for (; itHelper != binaryRelation[*it].end(); ++itHelper)
        binaryRelation[element.first].insert(*itHelper);
    }
  }

  return binaryRelation;
}

map<int, set<int>>
    transitiveClosure(map<int, set<int>> binaryRelation)
{
    while (!isTransitive(binaryRelation))
        binaryRelation =
          transitiveClosureMachinery(binaryRelation);

    return binaryRelation;
}
```

Листинг 15: Код программы.

Результат тестирования программы транзитивного замыкания бинарного отношения.

Для демонстрации работы программы рассмотрим произвольное отношение δ.

Перейдём на ручной ввод и сгенерируем 6 пар элементов, соответствующих бинарному отношению δ. Получаем следующие пары:

```
INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
0 2 1 4 2 2 2 4 4 3 4 4

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(0; 2). (1; 4). (2; 2), (2; 4). (4; 3), (4; 4).
```

Рисунок 33 – Бинарное отношение, не обладающее свойством транзитивности.

Как видно, данное отношение не является транзитивным. Построим транзитивное замыкание этого бинарного отношения:

```
INPUT THE NUMBER OF PAIRS:
6
NOW INPUT THE PAIRS:
0 2 1 4 2 2 2 4 4 3 4 4

THE ELEMENTS OF YOUR BINARY RELATION ARE:
(0; 2). (1; 4). (2; 2), (2; 4). (4; 3), (4; 4).

BINARY RELATION IS NOT TRANSITIVE.

BINARY RELATION IS NOT TRANSITIVE.

BINARY RELATION IS TRANSITIVE.

THE ELEMENTS OF YOUR BINARY RELATION AFTER TRANSITIVE CLOSURE ARE:

(0; 2), (0; 3), (0; 4). (1; 3), (1; 4). (2; 2), (2; 3), (2; 4). (4; 3), (4; 4).
```

Рисунок 34 – Бинарное отношение после транзитивного замыкания.

Заметим, что $(f \circ f)(\delta)$ (где f - функция *transitiveClosureMachinery*) оказалось недостаточно для замыкания исходного бинарного отношения.

Получившееся бинарное отношение является замкнутым относительно свойства транзитивности, что при третьем применении функции *transitiveClosureMachinery* подтверждает выход программы *isTransitive.*

Оценка временной сложности алгоритма построения транзитивного замыкания.

Определим n как число уникальных элементов, встречающихся в бинарном отношении, n_i как число элементов в списке смежности i-го элемента, а n_{ij} как j-ый элемент в таком списке смежности. Посчитаем число элементарных операций для того, чтобы свойство транзитивности было выполнено для одного элемента (например, первого):

$$n_{n_{11}} + n_{n_{12}} + \cdots + n_{n_{1n}} = \sum_{i=1}^{n} n_{n_{1i}} \leq n^2.$$

Тогда временная сложность выполнения операций для одного элемента с учётом сложности вставки элементов не превышает $O(n^2 \log n)$. Тогда временная сложность для n элементов будет равна $O(n^3 \log n)$.

Но также необходимо учесть, что построение транзитивного замыкания может не завершиться за один вызов функции, поэтому окончательная сложность алгоритма:

$$T(transitiveClosure,\, n) = O(Cn^3 \log n),\ 1 \leq C < \infty.$$

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы мною были изучены основные свойства бинарных отношений и операций их замыкания. В качестве практического задания мною были построены алгоритмы определения свойств рефлексивности, симметричности и транзитивности, классификации бинарных отношений и построения основных замыканий бинарных отношений с использованием структуры set, были представлены псевдокоды написанных по построенным алгоритмам программ, а также проведена оценка временной сложности представленных алгоритмов.