МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ (БАЗОВОЙ) ПРАКТИКЕ

студента 4 курса 451 группы
направления 38.03.05 — Бизнес-информатика

механико-математического факультета
Чайковского Петра Ильича

Место прохождения: завод "Тантал"
Сроки прохождения: с 29.06.2019 г. по 26.07.2019 г.
Оценка:

Руководитель практики от СГУ

доцент, к. ф.-м. н. _____ Н. Ю. Агафонова

Руководитель практики от организации

ведущий программист _____ Д. Э. Кнутов

Саратов 2019

Тема практики:«Правила оформления курсовых и дипломных работ»

СОДЕРЖАНИЕ

1 Постановка задачи 4
2 Теоретические сведения по рассмотренным темам с их обоснованием . 5
3 Результаты работы 8
3.1 Алгоритм определения свойства рефлексивности. 8
3.2 Алгоритм определения свойства симметричности. 12
3.3 Алгоритм определения свойства антисимметричности. 16
3.4 Алгоритм определения свойства транзитивности. 20
3.5 Алгоритм классификации бинарных отношений. 24
3.6 Описание алгоритмов построения основных замыканий бинарных отношений 26
3.7 Псевдокоды рассмотренных алгоритмов 27
3.8 Коды программ, реализуюших рассмотренные алгоритмы 28
3.9 Результаты тестирования программ 29
3.10 Оценки сложности рассмотренных алгоритмов 30

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.

```
1  isReflexive(binaryRelation)
2  {
3    for element in binaryRelation
4    {
5      flag = false;
6
7      for subelement in element:
8        if (subelement == element):
9          flag = true;
10
11     if (!flag)
12       return false;
13   }
14
15   return true;
16 }
```

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, на котором 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:

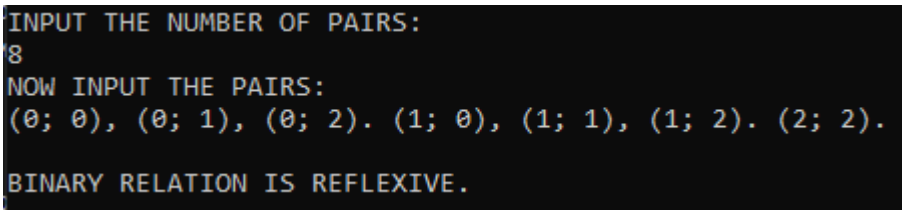

Рисунок 4 – Проверка на рефлексивность пройдена.

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

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.

```
1  isSymmetric(binaryRelation)
2  {
3    for element in binaryRelation
4      for subelement in element
5      {
6        flag = false
7
8        if element in binaryRelation[subelement]
9          flag = true
10
11       if (!flag)
12         return false;
13     }
14
15   return true;
16 }
```

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, на котором 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:

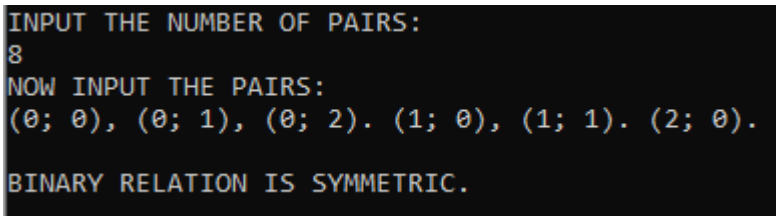

Рисунок 8 – Проверка на симметричность пройдена.

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

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.

```
1  isAntisymmetric(binaryRelation)
2  {
3    for element in binaryRelation:
4      for subelement in element
5      {
6        isAntisymmetric = true;
7
8        if subelement is not element
9        {
10         for elt in binaryRelation[subelement]
11           if (elt == element)
12             isAntisymmetric = false;
13       }
14
15       if (!isAntisymmetric)
16         return false;
17     }
18
19   return true;
20 }
```

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, на котором 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.

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.

```
1  isTransitive(binaryRelation)
2  {
3    for element in binaryRelation:
4      for subelement in element:
5        for elt in binaryRelation[subelement]
6        {
7          isTransitive = false;
8
9          for eltSup in element:
10           if (elt == eltSup)
11             isTransitive = true
12
13         if (!isTransitive)
14           return false;
15       }
16     }
17   }
18
19   return true;
20 }
```

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, на котором 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:

```
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).

BINARY RELATION IS TRANSITIVE.
```

Рисунок 16 – Проверка на транзитивность пройдена.

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

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 << "\BINARY RELATION IS AN ORDER RELATION.\n";
  else
    cout << "\BINARY RELATION IS NOT AN ORDER RELATION.\n";
}

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

```
22 {
23   if (isReflexive(binaryRelation) &&
24       isTransitive(binaryRelation))
25     cout << "\nBINARY RELATION IS A QUASI-ORDER RELATION.\n";
26   else
27     cout << "\nBINARY RELATION IS NOT A QUASI-ORDER RELATION.\n";
28 }
```

Листинг 9: Набор функций, осуществляющий классификацию бинарных отношений.

3.6 Описание алгоритмов построения основных замыканий бинарных отношений

3.7 Псевдокоды рассмотренных алгоритмов

3.8 Коды программ, реализуюших рассмотренные алгоритмы

3.9 Результаты тестирования программ

3.10 Оценки сложности рассмотренных алгоритмов