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

ПРЕОБРАЗОВАТЕЛИ КОДОВ

ОТЧЕТ

студента 3 курса 331 группы
специальности 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Бородина Артёма Горовича

Проверил
аспирант ______________ А. А. Мартышкин

Саратов 2022

СОДЕРЖАНИЕ

1 Цель работы и порядок выполнения

Цель работы — изучение основных понятий теории полугрупп.

Порядок выполнения работы

1. Рассмотреть понятия полугруппы, подполугруппы и порождающего множества. Разработать алгоритм построения подполугрупп по таблице Кэли.
2. Разработать алгоритм построения полугруппы бинарных отношений по заданному порождающему множеству.
3. Рассмотреть понятия подгруппы, порождающего множества и определяющих соотношений. Разработать алгоритм построения полугруппы по порождающему множеству и определяющим соотношениям.

2 Теоретические сведения по рассмотренным темам с их обоснованием

Определение. *Полугруппа* – это алгебра $S = (S, \cdot)$ с одной ассоциативной бинарной операцией $\cdot$, т.е. выполняется $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ для любых $x, y, z \in S$. Если полугрупповая операция называется умножением (соответственно, сложением), то полугруппу называют *мультипликативной* (соответственно, *аддитивной*).

Определение. Подмножество X полугруппы S называется *подполугруппой*, если X устойчиво относительно операции умножения, т.е. для любых $x, y \in X$ выполняется свойство: $x \cdot y \in X$. В этом случае множество X с ограничением на нем операции умножения исходной полугруппы S образует полугруппу.

Определение (*Порождающее множество*). В силу общего свойства подалгебр пересечение любого семейства X_i $(i \in I)$ подполугрупп полугруппы S является подполугруппой S и, значит, множество $Sub(S)$ всех подполугрупп полугруппы S является системой замыканий. Следовательно, для любого подмножества X полугруппы S существует наименьшая подполугруппа S, содержащая множество X. Такая полугруппа обозначается символом $<X>$ и называется подполугруппой S, порождённой множеством X. При этом множество X называется также порождающим множеством подполугруппы $<X>$.

В частности, если $<X> = S$, то X называется *порождающим множеством полугруппы* S и говорят, что множество X порождает полугруппу S.

Видно, что полугруппа $<X>$ состоит из всевозможных конечных произведений $x_1 \cdot ... \cdot x_n$ элементов $x_1, \ldots, x_n \in X$, т.е. выполняется равенство:

$$<X> = \{x_1 \cdot ... \cdot x_n : n \in N \text{ и } x_1, \ldots, x_n \in X\}$$

Алгоритм вычисления подполугруппы $<X> \subset S$:

1. Положим $i = 0,\ X_0 = X$.
2. Для X_i вычислим $\overline{X}_l = \{x \cdot y : x \in X_i \wedge y \in X\}$ и положим $X_{i+1} = X_i \cup \overline{X}_l$.
3. Вычисляем $<X> = \bigcup\limits_{i=0}^{\infty} X_i$.

Определение (*Определяющее соотношение и копредставление полугруппы S*). Для любой конечной полугруппы S найдется такой конечный алфавит A, что для некоторого отображения $\varphi : A \to S$ выполняется равенство <$\varphi(A)$> = S и, значит, $S \cong A^+/\ker\varphi$. В это случае множество A называется *множеством порождающих символов* полугруппы S (относительно отображения $\varphi : A \to S$). Если при этом для слов $w_1, w_2 \in A^+$ выполняется равенство $\varphi(w_1) = \varphi(w_2)$, т.е. $w_1 \equiv w_2(\ker\varphi)$, то говорят, что на S *выполняется соотношение* $w_1 = w_2$ (относительно отображения $\varphi : A \to S$).

В некоторых случаях можно выбрать подмножество $\rho \subset \ker\varphi$, которое однозначно определяет конгруэнцию $\ker\varphi$ как наименьшую конгруэнцию полугруппы A^+, содержащую отношение ρ, т.е. $\ker\varphi = f_{con}(\rho) = f_{eq}(f_{reg}(\rho))$. В случае $(w_1, w_2) \in \rho$ будет выполняться равенство $\varphi(w_1) = \varphi(w_2)$ – будем называть такие выражения *определяющими соотношениями*. Из таких соотношений конгруэнция $\ker\varphi$ строится с помощью применения следующих процедур к словам $u, v \in A^+$:

1. слово v *непосредственно выводится* из слова u, если v получается из u заменой некоторого подслова w_1 на слово w_2, удовлетворяющее определяющему соотношению $w_1 = w_2$, т.е. $(u, v) = (xw_1y, xw_2y)$ для некоторых $x, y \in A^*$;
2. слово v *выводится* из слова u, если v получается из u с помощью конечного числа применения процедуры (1).

Если все выполняющиеся на S соотношения выводятся из определяющих соотношений совокупности ρ, то конгруэнция $\ker\varphi$ полностью определяется отношением ρ и выражение

$$< A : \{w_1 = w_2 : (w_1, w_2) \in \rho\} >$$

называется *копредставлением полугруппы S*.

3 Результаты работы

3.1 Алгоритм построения подполугруппы по заданному порождающему множеству

Описание алгоритма построения подполугруппы по заданному порождающему множеству.

Вход: полугруппа S с таблицей Кэли размерности $N \times N$, а также порождающее множество $X \subset S$.

Выход: построенная подполугруппа $<X> \subset S$.

Метод: вычисляем подполугруппу $<X>$ в соответствии с упомянутым алгоритмом. На каждом шаге новые элементы добавляем в структуру set, содержащую только уникальные элементы. Если после вычисления $\overline{X}_l$, а затем $X_{i+1} = X_i \cup \overline{X}_l$ (X_i определяется структурой set) размер контейнера не увеличился, это означает, что в него не было добавлено новых элементов, что в свою очередь свидетельствует о том, что процесс вычисления подполугруппы $<X> \subset S$ закончен.

Псевдокод алгоритма построения подполугруппы по заданному порождающему множеству.

```
get_subsemigroup(<N×N matrix> CayleyTable, semigroupSubset[])
{
  <set> newElements, X_i = semigroupSubset;
  while (true)
  {
    currentSize = X_i.size();
    for object in X_i:
      for diffObject in X_i:
        newElements.insert(
          CayleyTable[row_in_table(object)]
                     [col_in_table(diffObject)]);
    for newItem in newElements:
      X_i.insert(newItem);
    if (X_i.size() == currentSize)
      break;
    newElements.clear();
  }
  return X_i;
}
```

Листинг 1: Псевдокод алгоритма.

Код программы построения подполугруппы по заданному порождающему множеству.

```
void getSubsemigroupMachinerie (vector<vector<int>> CayleyTable,
                                set<int>& subsemigroup)
{
  set<int> ::iterator i, j;
  set<int> newElements, X_i = semigroupSubset;

  while (true)
    {
      unsigned short curSize = X_i.size ();

      for (i = X_i.begin (); i != X_i.end (); ++i)
        for (j = X_i.begin (); j != X_i.end (); ++j)
          newElements.insert (CayleyTable[*i][*j]);
      for (i = newElements.begin(); i != newElements.end(); ++i)
        X_i.insert (*i);
      if (X_i.size () - curSize == 0)
        break;
      newElements.clear ();
    }
  subsemigroup = X_i;
}
```

Листинг 2: Код программы.

Результат тестирования программы построения подполугруппы по заданному порождающему множеству.

Рассмотрим полугруппу $S = \{a, b, c, d\}$ с таблицей Кэли размерности 4×4 и порождающим множеством $X \subset S = \{b\}$. Построим подполугруппу <X> $\subset S$ по заданным входным данным.

```
INPUT CAYLEY TABLE DIMENSION:
4
INPUT ELEMENTS OF THE CAYLEY TABLE:
a b c d
b c d a
c d a b
d a b c
INPUT SUBSET SIZE:
1
INPUT SUBSET ELEMENTS:
b
RESULTING SEMIGROUP IS:
<a, b, c, d>
```

Рисунок 1 – Результат построения подполугруппы <X> по заданному порождающему множеству.

Получилось, что <X> = S, и $X = \{b\}$ является *порождающим множеством полугруппы S*.

Оценка сложности алгоритма построения подполугруппы по заданному порождающему множеству.

Для генерации новых элементов ищутся все попарные произведения элементов из контейнера X_i. Значения этих произведений хранятся в заданной таблице Кэли. Количество элементов в X_i на каждом шаге не превышает размера заданной полугруппы S, т.е. числа строк или столбцов в заданной таблице Кэли (N). Внешний цикл $while$ сработает не более N раз, т.к. алгоритм будет работать, пока на каждом шаге в X_i будет добавляться один или более элемент. Таким образом, оценка сложности алгоритма построения подполугруппы по заданному порождающему множеству принимает вид: $O(N^3)$.

3.2 Алгоритм построения полугруппы бинарных отношений по заданному порождающему множеству

Описание алгоритма построения полугруппы бинарных отношений по заданному порождающему множеству.

Вход: порождающее множество X матриц бинарных отношений.

Выход: полугруппа $<X>$ и таблица Кэли полученной полугруппы.

Метод: аналогичен методу построения подполугруппы по заданному порождающему множеству. Если на некотором шаге после вычисления попарных композиций бинарных отношений не было сформировано новых элементов, то процесс построения полугруппы окончен, и программа переходит к построению таблицы Кэли полученной полугруппы.

Псевдокод алгоритма построения полугруппы бинарных отношений по заданному порождающему множеству.

```
get_semigroup(<set <N×N matrix>> binaryRelationSet)
{
  <set <N×N matrix>> newElements, Xi = binaryRelationSet;
  while (true)
  {
    currentSize = Xi.size();
    for binRelMat in Xi:
      for diffBinRelMat in Xi:
        newElements.insert(
          get_binary_relation_composition(
            binRelMat, diffBinRelMat);
    for newItem in newElements:
      Xi.insert(newItem);
    if (Xi.size() == currentSize)
      break;
    newElements.clear();
  }
  return Xi;
}
```

Листинг 3: Псевдокод алгоритма.

Код программы построения полугруппы бинарных отношений по заданному порождающему множеству.

```
vector<vector<unsigned short>> boolMatricesMultiplication (
  vector<vector<unsigned short>> fM,
  vector<vector<unsigned short>> sM)
{
  unsigned short i, j, k, matSize = fM.size (), product = 0;
  vector<vector<unsigned short>> resM (
    matSize, vector<unsigned short> (matSize, 0));

  for (i = 0; i < matSize; ++i)
    for (j = 0; j < matSize; ++j)
      {
        for (k = 0; k < matSize; ++k)
          product += fM[i][k] * sM[k][j];
        resM[i][j] = (product > 0 ? 1 : 0);
        product = 0;
      }
  return (resM);
}

vector<pair<vector<vector<unsigned short>>, char>>
  matrixMappings;
char symbolToMap = 'a';

void getSemigroupMachinerie (
  set<vector<vector<unsigned short>>> binaryRelationSet,
  set<vector<vector<unsigned short>>>& semigroup)
{
  set<vector<vector<unsigned short>>> ::iterator i, j;
  set<vector<vector<unsigned short>>> newElements,
                                      X_i = binaryRelationSet;

  while (true)
    {
      unsigned short curSize = X_i.size ();

      for (i = X_i.begin (); i != X_i.end (); ++i)
        for (j = X_i.begin (); j != X_i.end (); ++j)
          newElements.insert (
            boolMatricesMultiplication (*i, *j));
```

```
      for (i = newElements.begin(); i != newElements.end(); ++i)
        X_i.insert (*i);
      if (X_i.size () - curSize == 0)
        break;
      newElements.clear ();
    }
  semigroup = X_i;
}

char find_corresponding_letter (
  vector<pair<vector<vector<unsigned short>>, char>> matMaps,
  vector<vector<unsigned short>> matToCheck)
{
  int i;
  for (i = 0; i < matMaps.size (); ++i)
    if (matMaps[i].first == matToCheck)
      return (matMaps[i].second);
  return ('.');
}

void display_matrix_Cayley_table (
  vector<pair<vector<vector<unsigned short>>, char>> matMaps)
{
  int i, j;

  cout << endl << "CAYLEY TABLE:\n";
  cout << "    ";
  for (i = 0; i < matMaps.size (); ++i)
    cout << setw (4) << matMaps[i].second;
  cout << "\n";

  for (i = 0; i < matMaps.size (); ++i)
    {
      cout << setw (4) << matMaps[i].second << setw (4);
      for (j = 0; j < matMaps.size (); ++j)
        cout << find_corresponding_letter (
          matMaps,
          boolMatricesMultiplication (matMaps[i].first,
                                      matMaps[j].first))
           << setw (4);
      cout << "\n";
```

```
    }
  cout << endl;
}

void getSemigroup ()
{
  unsigned short i = 0, j, k = 0, binaryRelationsNumber;
  set<vector<vector<unsigned short>>> ::iterator it;
  set<vector<vector<unsigned short>>> binaryRelationSet,
                                      semigroup;
  cout << "INPUT THE NUMBER OF BINARY RELATIONS:\n";
  cin >> binaryRelationsNumber;
  cout << "INPUT THE DIMENSION OF MATRICES:\n";
  cin >> matrixDimension;

  while (i < binaryRelationsNumber)
    {
      vector<vector<unsigned short>>
        binaryRelationMatrix (
          matrixDimension, vector<unsigned short> (
                             matrixDimension, 0));
      cout << "INPUT THE BINARY RELATION MATRIX:\n";
      getMatrix (binaryRelationMatrix);

      if (binaryRelationSet.empty ())
        {
          binaryRelationSet.insert (binaryRelationMatrix);
          ++i;
          continue;
        }
      for (it = binaryRelationSet.begin ();
           it != binaryRelationSet.end (); ++it)
        if (*it == binaryRelationMatrix)
          cout << "THIS MATRIX IS ALREADY IN THE SET.
                   TRY ANOTHER ONE.\n";
        else
          {
            binaryRelationSet.insert (binaryRelationMatrix);
            ++i;
          }
    }
```

```
  getSemigroupMachinerie (binaryRelationSet, semigroup);

  cout << "THE RESULTING SEMIGROUP IS:\n";

  for (it = semigroup.begin (); it != semigroup.end (); ++it)
    {
      cout << symbolToMap << ":\n";
      matrixMappings.push_back (make_pair (*it, symbolToMap++));
      for (i = 0; i < matrixDimension; ++i)
        {
          for (j = 0; j < matrixDimension; ++j)
            cout << (*it)[i][j] << " ";
          cout << '\n';
        }
      cout << '\n';
    }
  display_matrix_Cayley_table (matrixMappings);
}
```

Листинг 4: Код программы.

ЗАКЛЮЧЕНИЕ