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

**Идеалы полугрупп**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ПРИКЛАДНАЯ УНИВЕРСАЛЬНАЯ АЛГЕБРА»

студента 3 курса 331 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Норикова Павла Сергеевича

|  |  |  |
| --- | --- | --- |
| Преподаватель  профессор, д.ф.-м.н. | \_\_\_\_\_\_\_\_\_\_ | В. А. Молчанов |
|  | подпись, дата |  |

Саратов 2022

ВВЕДЕНИЕ

Цель данной работы — изучение строения полугрупп с помощью отношений Грина, разработка алгоритмов построения идеалов полугруппы по таблице Кэли, вычисление отношений Грина и построения «egg-box»-картины конечной полугруппы.

ИДЕАЛЫ ПОЛУГРУППЫ

Пусть – произвольная полугруппа. Непустое подмножество называется правым (соответственно, левым) идеалом полугруппы , если для любых выполняется условие: (соответственно ), т.е. (соответственно, ). Если – одновременно левый и правый идеал полугруппы , то называется двусторонним идеалом (или просто идеалом) полугруппы . Ясно, что в коммутативной полугруппе все эти определения совпадают.

Множество всех идеалов (соответственно, левых идеалов или правых идеалов ) любой полугруппы является системой замыкания.

Пусть – подмножество полугруппы . Тогда наименьший правый идеал полугруппы , содержащий подмножество , равен , наименьший левый идеал полугруппы , содержащий подмножество , равен и наименьший идеал полугруппы , содержащий подмножество , равен .

В частности, любой элемент определяет наименьшие правый, левый и двусторонний идеалы: , которые называются главными (соответственно, правыми, левыми и двусторонними) идеалами.

Минимальные относительно теоретико-множественного включения идеалы (соответственно, левые или правые идеалы) называются минимальными идеалами (соответственно, минимальными левыми или правыми идеалами).

Алгоритм 1 – построение идеалов полугруппы по таблице Кэли.

*Вход*. Таблица Кэли v полугруппы S.

*Выход*. Множество правых r, левых l и двусторонних идеалов b.

Шаг 1. Для всех подмножеств all[k] множества S

Шаг 1.1. Создаём set\_r = all[k] и для всех элементов x подмножества all[k]

Шаг 1.1.1. Для всех i от 1 до n

Шаг 1.1.1.1. Добавляем в set\_r v[x][i].

Шаг 1.2. Если all[k] = set\_r, добавляем set\_r в r.

Шаг 1.3. Создаём set\_l = all[k] и для всех элементов x подмножества all[k]

Шаг 1.3.1. Для всех i от 1 до n

Шаг 1.3.1.1. Добавляем в set\_l v[i][x].

Шаг 1.4. Если all[k] = set\_l, добавляем set\_l в l.

Шаг 1.5. Если set\_r = set\_l = all[k], добавляем set\_l в b.

Шаг 2. Выводим r, l и b.

ПОНЯТИЯ И СВОЙСТВА ОТНОШЕНИЙ ГРИНА НА ПОЛУГРУППАХ

Отображения  определяют ядра ℐ, ℛ ℒ по формулам:

ℐ ⟺

ℛ ⟺

ℒ ⟺

Все эти отношения, а также отношения 𝔇 = ℛ ∨ ℒ, ℋ = ℛ ∩ ℒ являются эквивалентностями на множестве , которые называются отношениями Грина полугруппы . Классы этих эквивалентностей, порожденные элементом , обозначаются , , , и , соответственно.

Отношения Грина полугруппы S удовлетворяют следующим свойствам:

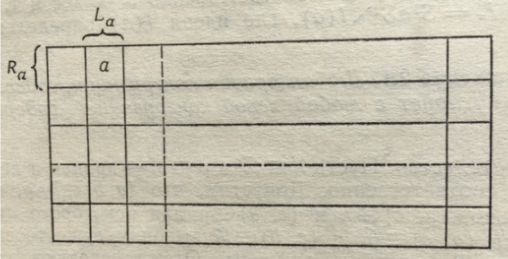
1) эквивалентность ℛ регулярна слева и эквивалентность ℒ регулярна справа, те. ℛ ⟹ ℛ и ℒ ⟹ ℒ для любых ,

2) эквивалентности ℛ, ℒ коммутируют,

3) 𝔇 = ℛ ∙ ℒ = ℒ ∙ ℛ,

4) если полугруппа конечна, то 𝔇 = ℐ,

5) любой класс эквивалентности 𝔇 можно изобразить с помощью следующей egg-box-диаграммы, клетки которой являются классами эквивалентности ℋ, лежащими в .



Алгоритм 2 – вычисление отношений Грина и построение «egg-box»-картины конечной полугруппы.

*Вход*. Таблица Кэли v полугруппы S.

*Выход*. Отношения Грина R, L, J, H, D и «egg-box»-картины.

Шаг 1. Для всех i от 1 до n

Шаг 1.1. Для всех j от 1 до n

Шаг 1.1.1. Создаём множества R1, R2, L1, L2.

Шаг 1.1.2. Добавляем в R1 и L1 S[i], а в R2 и L2 S[j].

Шаг 1.1.3. Для всех k от 1 до n

Шаг 1.1.3.1. Добавляем в R1 v[i][k], в R2 v[j][k], в L1 v[k][i], в L2 v[k][j].

Шаг 1.1.4. Если R1 = R2, добавляем в R пару (S[i], S[j]).

Шаг 1.1.5. Если L1 = L2, добавляем в L пару (S[i], S[j]).

Шаг 1.1.6. Создаём множества J1, J2.

Шаг 1.1.7. Добавляем в J1 S[i], а в J2 S[j].

Шаг 1.1.8. Для всех k от 1 до n

Шаг 1.1.8.1. Добавляем в J1 v[k][i], в J2 v[k][j].

Шаг 1.1.9. Для всех элементов el из J1

Шаг 1.1.9.1. Для всех k от 1 до n

Шаг 1.1.9.1.1. Добавляем в J1 v[el][k]

Шаг 1.1.10. Для всех элементов el из J2

Шаг 1.1.10.1. Для всех k от 1 до n

Шаг 1.1.10.1.1. Добавляем в J2 v[el][k]

Шаг 1.1.11. Если J1 = J2, добавляем в J пару (S[i], S[j]).

Шаг 2. Выводим L, R, J, D = J, H = L ∩ R.

Шаг 3. Выводим L, R, J, D = J, H = L ∩ R.

Шаг 4. Находим классы эквивалентностей отношений R, L, D.

Шаг 5. Для каждого класса эквивалентности x из D

Шаг 5.1. Для каждого .

Шаг 5.1. Для каждого .

Шаг 5.1. Выводим .

ВРЕМЕННАЯ СЛОЖНОСТЬ АЛГОРИТМОВ

Построение идеалов полугруппы по таблице Кэли – .

Вычисление отношений Грина и построение «egg-box»-картины конечной полугруппы – .

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

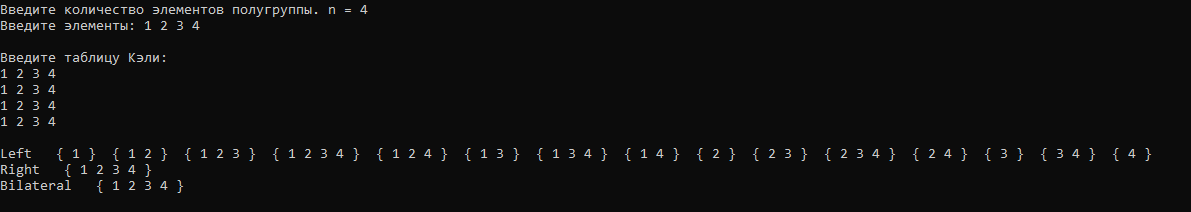


Рис.1 – первый тест

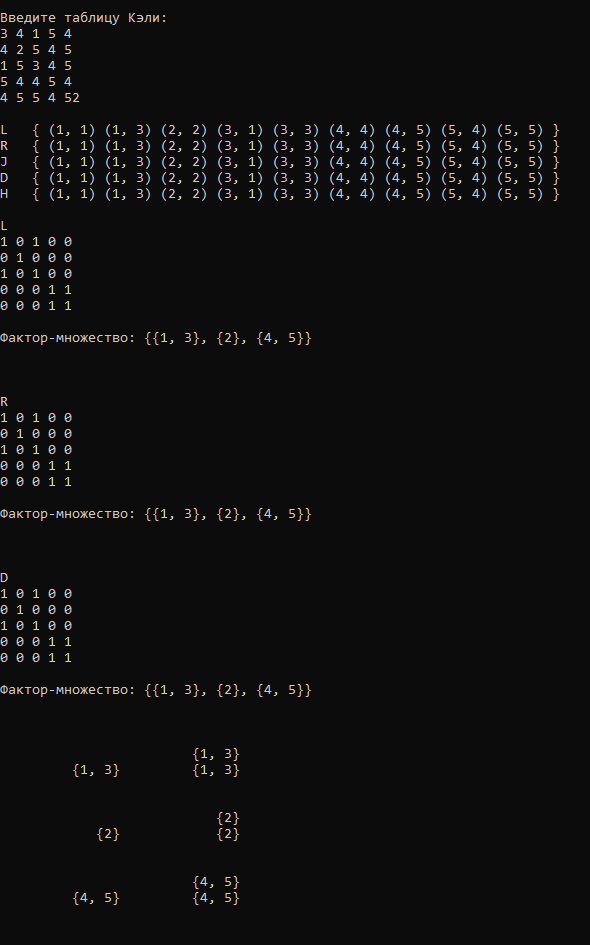
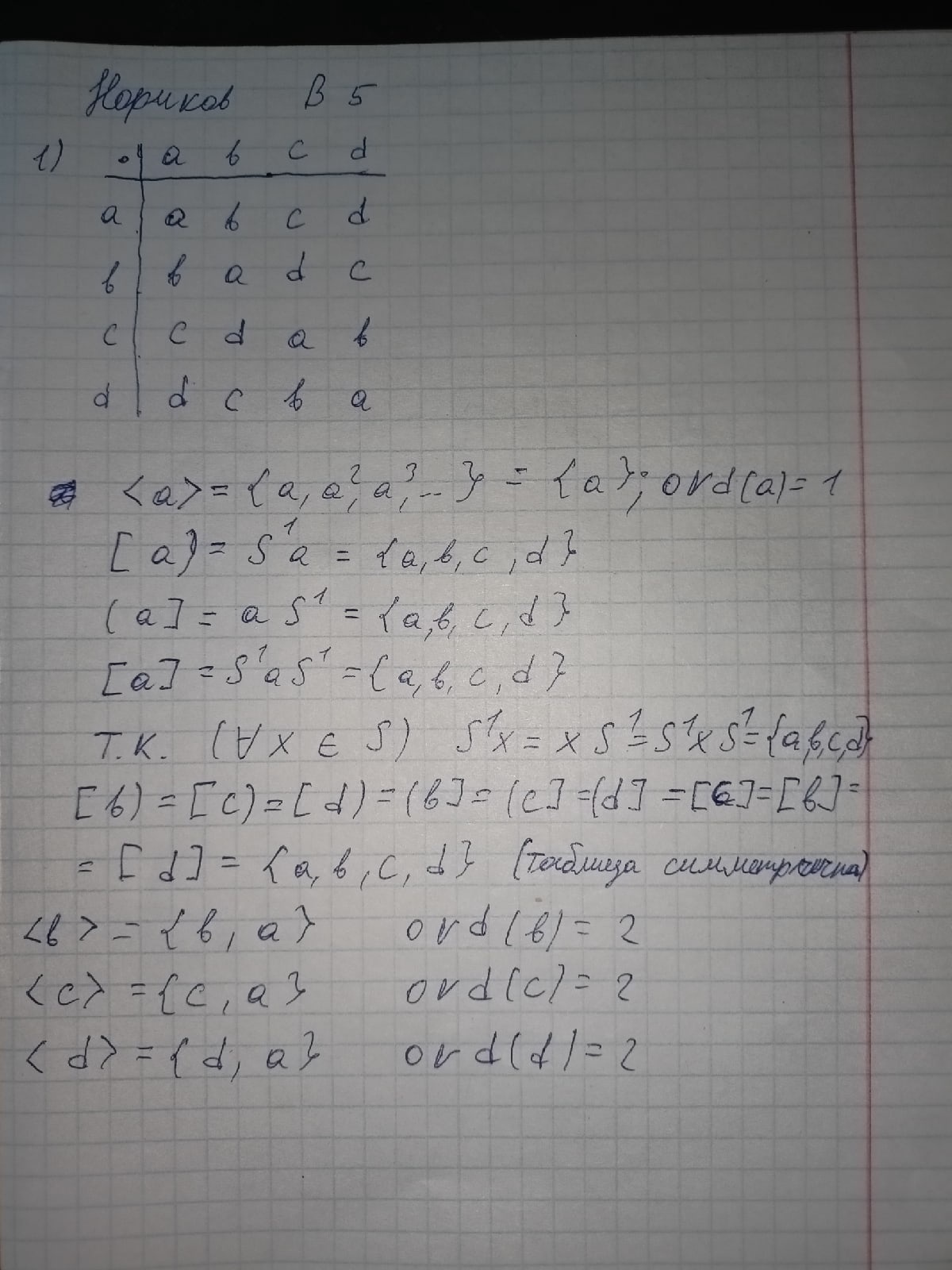
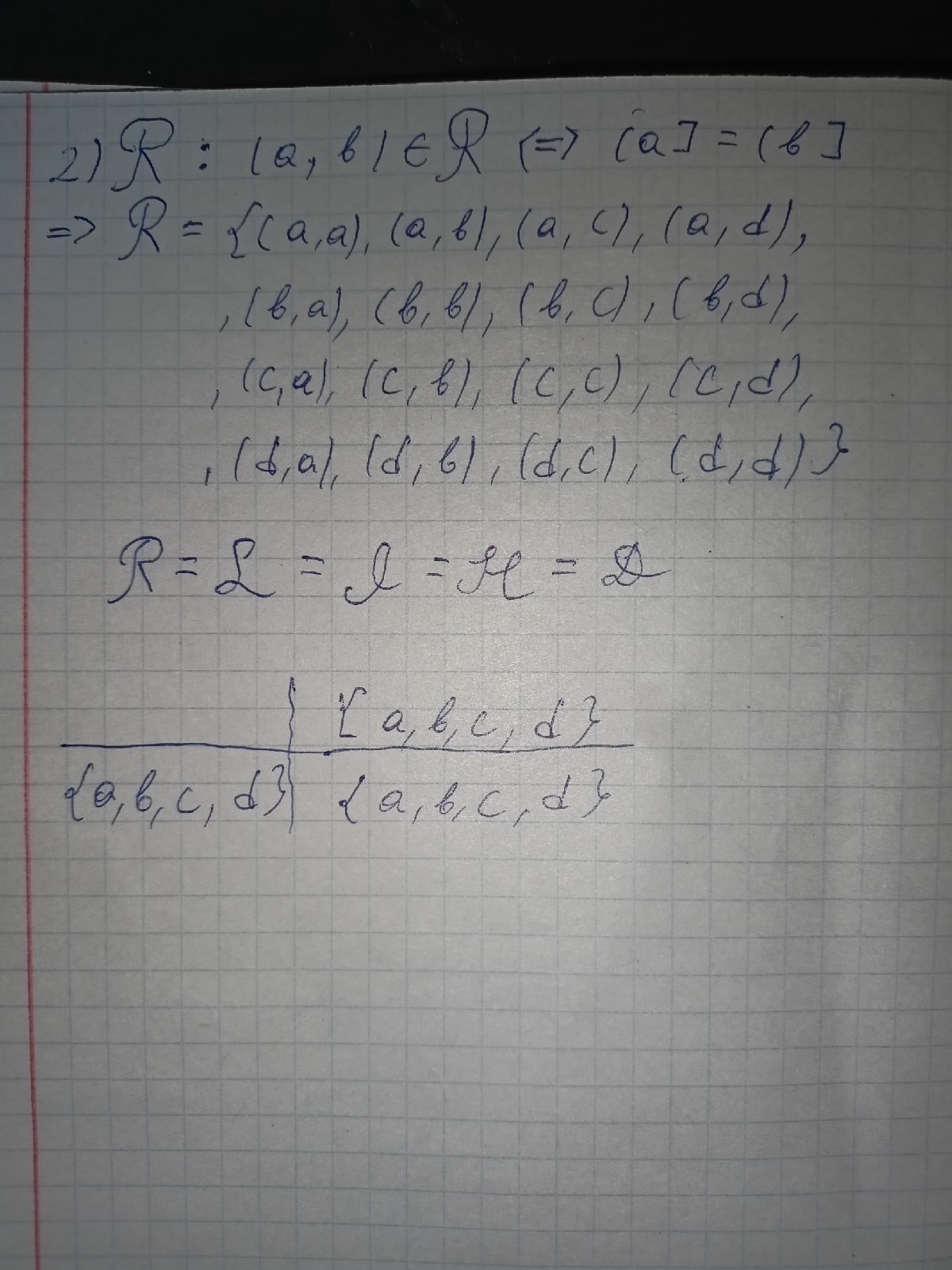
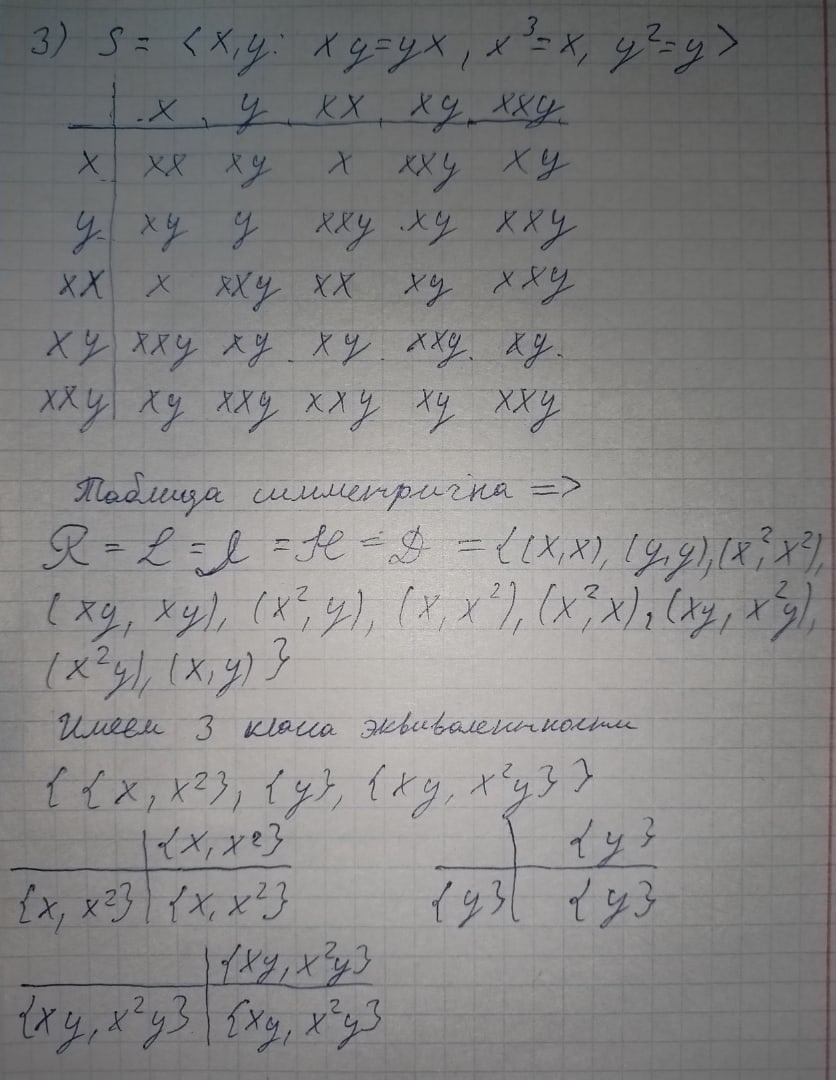


Рис.2 – второй тест

ПИСЬМЕННОЕ ЗАДАНИЕ







КОД ПРОГРАММЫ

#include <iostream>

#include <iomanip>

#include <vector>

#include <algorithm>

#include <set>

#include<map>

#include<string>

using namespace std;

int find\_number(char\* els, int n, char x) {

for (int i = 0; i < n; i++) {

if (els[i] == x)

return i;

}

}

bool assoc(char\*\* v, char\* els, int n) {

int checker = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

if (v[find\_number(els, n, v[i][j])][k] != v[i][find\_number(els, n, v[j][k])])

return false;

}

}

}

return true;

}

set<set<char>> all\_gen(char\* els, int n, int x) {

set<set<char>> res;

while (x < n) {

if (x == 1) {

for (int i = 0; i < n; i++) {

set<char> t1;

t1.insert(els[i]);

res.insert(t1);

}

}

else {

for (int i = 0; i < n; i++) {

for (set<char> t1 : res) {

set<char> t2 = t1;

t2.insert(els[i]);

res.insert(t2);

}

}

}

x++;

}

return res;

}

void alg\_1(char\*\* v, int n, char\* els) {

set<set<char>> all1 = all\_gen(els, n, 1);

set<set<char>> r\_res;

set<set<char>> l\_res;

set<set<char>> b\_res;

for (set<char> x : all1) {

set<char> res1 = x;

for (char el : x)

for (int i = 0; i < n; i++)

res1.insert(v[find\_number(els, n, el)][i]);

if (x.size() == res1.size())

r\_res.insert(x);

set<char> res2 = x;

for (char el : x)

for (int i = 0; i < n; i++)

res2.insert(v[i][find\_number(els, n, el)]);

if (x.size() == res2.size())

l\_res.insert(x);

if (res1 == res2 && res1.size() == x.size())

b\_res.insert(x);

}

cout << "\nLeft ";

for (set<char> x : l\_res) {

cout << "{ ";

for (char y : x) {

cout << y << " ";

}

cout << "} ";

}

cout << "\nRight ";

for (set<char> x : r\_res) {

cout << "{ ";

for (char y : x) {

cout << y << " ";

}

cout << "} ";

}

cout << "\nBilateral ";

for (set<char> x : b\_res) {

cout << "{ ";

for (char y : x) {

cout << y << " ";

}

cout << "} ";

}

cout << endl;

}

set<pair<char, char>> my\_union(set<pair<char, char>> x, set<pair<char, char>> y) {

set<pair<char, char>> res;

for (pair<char, char> el1 : x)

if (y.find(el1) != y.end())

res.insert(el1);

return res;

}

set<int> my\_intersection(vector<int> x, vector<int> y) {

set<int> res;

for (int el1 : x)

for (int el2 : y)

if (el1 == el2)

res.insert(el1);

return res;

}

vector<int> my\_intersection2(vector<int> x, vector<int> y) {

vector<int> res;

for (int el1 : x)

for (int el2 : y)

if (el1 == el2)

res.push\_back(el1);

return res;

}

vector<vector<int>> eq(int\*\* v2, int n, char\* els) {

vector<vector<int>> res;

int\* count = new int[n];

for (int i = 0; i < n; i++) {

count[i] = 1;

}

for (int i = 0; i < n; i++) {

if (count[i]) {

vector<int> srez;

for (int j = i; j < n; j++) {

if (v2[i][j] && count[j]) {

count[j] = 0;

srez.push\_back(j);

}

}

count[i] = 0;

res.push\_back(srez);

}

}

cout << endl << "Фактор-множество: {";

for (int i = 0; i < res.size(); i++) {

cout << "{";

for (int j = 0; j < res[i].size(); j++) {

cout << els[res[i][j]];

if (j != res[i].size() - 1)

cout << ", ";

}

cout << "}";

if (i != res.size() - 1)

cout << ", ";

}

cout << "}" << endl;

return res;

}

string print\_inter(set<int> v, char\* els) {

string res = "";

res += "{";

for (int x : v) {

res = res + els[x] + ", ";

}

res.pop\_back();

res.pop\_back();

res += "}";

return res;

}

string print\_inter\_vec(vector<int> v, char\* els) {

string res = "";

res += "{";

for (int x : v) {

res = res + els[x] + ", ";

}

res.pop\_back();

res.pop\_back();

res += "}";

return res;

}

void alg\_2(char\*\* v, int n, char\* els) {

set<pair<char, char>> R;

set<pair<char, char>> L;

set<pair<char, char>> J;

set<pair<char, char>> H;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

set<char> R1;

R1.insert(els[i]);

set<char> R2;

R2.insert(els[j]);

set<char> L1;

L1.insert(els[i]);

set<char> L2;

L2.insert(els[j]);

for (int k = 0; k < n; k++) {

R1.insert(v[i][k]);

R2.insert(v[j][k]);

L1.insert(v[k][i]);

L2.insert(v[k][j]);

}

if (R1 == R2)

R.insert(make\_pair(els[i], els[j]));

if (L1 == L2)

L.insert(make\_pair(els[i], els[j]));

set<char> J1;

J1.insert(els[i]);

set<char> J2;

J2.insert(els[j]);

for (int k = 0; k < n; k++) {

J1.insert(v[k][i]);

J2.insert(v[k][j]);

}

for (char el : J1) {

for (int k = 0; k < n; k++) {

J1.insert(v[find\_number(els, n, el)][k]);

}

}

for (char el : J2) {

for (int k = 0; k < n; k++) {

J2.insert(v[find\_number(els, n, el)][k]);

}

}

if (J1 == J2)

J.insert(make\_pair(els[i], els[j]));

}

}

cout << "\nL { ";

for (pair<char, char> x : L) {

cout << "(" << x.first << ", " << x.second << ") ";

}

cout << "}\nR { ";

for (pair<char, char> x : R) {

cout << "(" << x.first << ", " << x.second << ") ";

}

cout << "}\nJ { ";

for (pair<char, char> x : J) {

cout << "(" << x.first << ", " << x.second << ") ";

}

//Полугруппа конечна, поэтому D = J

cout << "}\nD { ";

for (pair<char, char> x : J) {

cout << "(" << x.first << ", " << x.second << ") ";

}

H = my\_union(L, R);

cout << "}\nH { ";

for (pair<char, char> x : H) {

cout << "(" << x.first << ", " << x.second << ") ";

}

cout << "}\n\n";

int\*\* v1;

v1 = new int\* [n];

for (int i = 0; i < n; i++) {

v1[i] = new int[n];

for (int j = 0; j < n; j++) {

v1[i][j] = 0;

}

}

for (pair<char, char> x : L) {

v1[find\_number(els, n, x.first)][find\_number(els, n, x.second)] = 1;

}

cout << "L \n";

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cout << v1[i][j] << " ";

}

cout << endl;

}

vector<vector<int>> l\_eq = eq(v1, n, els);

cout << "\n\n\n";

int\*\* v2;

v2 = new int\* [n];

for (int i = 0; i < n; i++) {

v2[i] = new int[n];

for (int j = 0; j < n; j++) {

v2[i][j] = 0;

}

}

for (pair<char, char> x : R) {

v2[find\_number(els, n, x.first)][find\_number(els, n, x.second)] = 1;

}

cout << "R \n";

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cout << v2[i][j] << " ";

}

cout << endl;

}

vector<vector<int>> r\_eq = eq(v2, n, els);

cout << "\n\n\n";

int\*\* v3;

v3 = new int\* [n];

for (int i = 0; i < n; i++) {

v3[i] = new int[n];

for (int j = 0; j < n; j++) {

v3[i][j] = 0;

}

}

for (pair<char, char> x : J) {

v3[find\_number(els, n, x.first)][find\_number(els, n, x.second)] = 1;

}

cout << "D \n";

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cout << v3[i][j] << " ";

}

cout << endl;

}

vector<vector<int>> d\_eq = eq(v3, n, els);

cout << "\n\n\n";

for (vector<int> cl : d\_eq) {

vector<vector<int>> res\_l;

vector<vector<int>> res\_r;

for (vector<int> el : l\_eq) {

if (my\_intersection2(cl, el) == el)

res\_l.push\_back(el);

}

for (vector<int> el : r\_eq)

if (my\_intersection2(cl, el) == el)

res\_r.push\_back(el);

cout << setw(15) << "";

for (vector<int> ans : res\_l) {

string str1 = print\_inter\_vec(ans, els);

cout << setw(15) << str1;

}

cout << endl;

for (vector<int> ans : res\_r) {

string str1 = print\_inter\_vec(ans, els);

cout << setw(15) << str1 << setw(15);

for (vector<int> ans2 : res\_l) {

string str2 = print\_inter(my\_intersection(ans, ans2), els);

cout << str2 << setw(15);

}

cout << endl;

}

cout << endl;

cout << endl;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int w = 999;

while (w > 0) {

cout << "1 - алгоритмы построения идеалов полугруппы по таблице Кэли" << endl;

cout << "2 - алгоритмы вычисления отношений Грина и построения «egg-box»-картины конечной полугруппы" << endl;

cout << "0 - выход" << endl;

cin >> w;

if (w == 1) {

int n;

cout << "Введите количество элементов полугруппы. n = ";

cin >> n;

char\* els = new char[n];

cout << "Введите элементы: ";

for (int i = 0; i < n; i++)

cin >> els[i];

char\*\* v1;

v1 = new char\* [n];

cout << endl;

cout << "Введите таблицу Кэли:" << endl;

for (int i = 0; i < n; i++) {

v1[i] = new char[n];

for (int j = 0; j < n; j++) {

cin >> v1[i][j];

}

}

if (assoc(v1, els, n))

alg\_1(v1, n, els);

else

cout << "Операция не ассоциативна" << endl;

}

if (w == 2) {

int n;

cout << "Введите количество элементов полугруппы. n = ";

cin >> n;

char\* els = new char[n];

cout << "Введите элементы: ";

for (int i = 0; i < n; i++)

cin >> els[i];

char\*\* v1;

v1 = new char\* [n];

cout << endl;

cout << "Введите таблицу Кэли:" << endl;

for (int i = 0; i < n; i++) {

v1[i] = new char[n];

for (int j = 0; j < n; j++) {

cin >> v1[i][j];

}

}

if (assoc(v1, els, n))

alg\_2(v1, n, els);

else

cout << "Операция не ассоциативна" << endl;

}

cout << endl;

}

return 0;

}

ЗАКЛЮЧЕНИЕ

В ходе данной работы были изучены и реализованы алгоритмы построения идеалов полугруппы по таблице Кэли, вычисления отношений Грина и построения «egg-box»-картины.