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

**Комбинаторная теория 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 основных понятий теории полугрупп, а также разработка алгоритмов построения подполугрупп по таблице Кэли, алгоритмов построения полугруппы бинарных отношений по заданному порождающему множеству и алгоритмов построения полугруппы по порождающему множеству и определяющим соотношениям.

ПОЛУГРУППЫ, ПОДПОЛУГРУППЫ И ПОРОЖДАЮЩИЕ МНОЖЕСТВА

Полугруппа – это алгебра с одной ассоциативной бинарной операцией , т.е. выполняется

для любых

Если полугрупповая операция называется умножением (соответственно, сложением), то полугруппу называют мультипликативной (соответственно, аддитивной).

Для любого непустого множества множество всех бинарных отношений на множестве с операцией композиции является полугруппой. Такая полугруппа называется симметрической полугруппой бинарных отношений на множестве и обозначается символом .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | … |  |
|  |  |  | … |  |
|  | … | … | … | … |
| ⋮ | … | … | … | … |
|  |  |  | … |  |

В случае конечного n-элементного множества операция умножения на задаётся таблицей Кэли размерности , строки и столбцы которой помечены элементами множества и в которой на пересечении –ой строки и–го столбца стоит произведение элементов .

Элемент полугруппы называется:

1) нулевым, если для всех (при мультипликативной записи операции полугруппы такой элемент обозначается символом 0 и называется нулем);

2) нейтральным (единичным), если для всех (при мультипликативной записи операции полугруппы такой элемент обозначается символом 1 и называется единицей);

3) обратимым, если для некоторого выполняется свойство: (такой элемент называется обратным для элемента и обозначается символом );

4) идемпотентом, если , т.е. .

Если в 1) выполняется лишь равенство для всех , то называется правым нулем. Аналогично определяется левый нуль и односторонние единицы.

Подмножество полугруппы называется подполугруппой, если устойчиво относительно операции умножения, т.е. для любых выполняется свойство: .

В силу общего свойства подалгебр пересечение любого семейства подполугрупп полугруппы является подполугруппой и, значит, множество всех подполугрупп полугруппы является системой замыканий. Следовательно, для любого подмножества полугруппы существует наименьшая подполугруппа , содержащая множество . Такая полугруппа обозначается символом и называется подполугруппой , порождённой множеством . При этом множество называется также порождающим множеством подполугруппы .

В частности, если , то называется порождающим множеством полугруппы и говорят, что множество порождает полугруппу .

Легко видеть, что полугруппа состоит из всевозможных конечных произведений элементов ,..., , т.е. выполняется равенство: и.

Алгоритм 1 – построение подполугруппы по таблице Кэли.

*Вход*. Таблица Кэли полугруппы S и подмножество .

*Выход*. Подполугруппа .

Шаг 1. .

Шаг 2. Для вычисляем .

Шаг 3. Вычисляем .

Шаг 4. Если не равен , , возвращаемся к шагу 2.

Шаг 5. Выводим .

Алгоритм 2 – построение полугрупп бинарных отношений по заданному порождающему множеству.

*Вход*. Конечное множество бинарных отношений (булевых матриц).

*Выход*. Полугруппа с таблицей Кэли.

Шаг 1. .

Шаг 2. Для вычисляем и заносим вычисленные композиции на пересечение элементов x и y в таблицу Кэли.

Шаг 3. Вычисляем .

Шаг 4. Если не равен , , возвращаемся к шагу 2.

Шаг 5. Выводим и таблицу Кэли.

ПОДГРУППЫ И ОПРЕДЕЛЯЮЩИЕ СООТНОШЕНИЯ

Полугруппа с единичным элементом называется моноидом. Другими словами, моноид – это алгебра с ассоциативной бинарной операцией и выделенным единичным элементом 1. При этом полугруппа называется полугруппой моноида . Моноид называется группой, если в нем все элементы обратимы.

Для любого моноида множество всех обратимых элементов с операцией умножения моноида является группой, т.е. для любых выполняется условие .

Подгруппой полугруппы назовем подполугруппу из S, являющуюся группой относительно бинарной операции, определенной в . Это эквивалентно тому, что есть подполугруппа из , в которой для любых существуют такие, что и . Отсюда легко получить, что подмножество полугруппы является подгруппой тогда и только тогда, когда для любого .

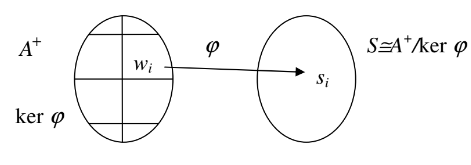
Пусть – произвольное множество, называемое алфавитом. Элементы называются буквами. Словом над алфавитом называется конечная последовательность букв алфавита .

Обозначим символом множество всех непустых слов над алфавитом и символом - множество слов .

Любая полугруппа является фактор-полугруппой некоторой полугруппы слов , т.е. для некоторой конгруэнции полугруппы .

Для любой конечной полугруппы найдется такой конечный алфавит , что для некоторого отображения выполняется равенство и, значит,. В этом случае множество называется множеством порождающих символов полугруппы (относительно отображения ).

Если при этом для слов выполняется равенство , т.е. , то говорят, что на выполняется соотношение (относительно отображения ).



Очевидно, что в общем случае множество таких соотношений для всех пар будет бесконечным и не представляется возможности эффективно описать полугруппу в виде полугруппы классов конгруэнции . Однако в некоторых случаях можно выбрать такое сравнительно простое подмножество , которое однозначно определяет конгруэнцию как наименьшую конгруэнцию полугруппы , содержащую отношение , т.е.

.

Так как в случае по-прежнему выполняется равенство , то будем писать и называть такие выражения определяющими соотношениями.

Алгоритм 3 – построение полугруппы по порождающему множеству и определяющим соотношениям:

*Вход*. Конечное множество символов и конечное множество определяющих соотношений

*Выход*. Полугруппа с таблицей Кэли.

Шаг 1. res = , = , = . Создаём словарь k.

Шаг 2. Для каждого слова w из res

Шаг 2.1. Для каждой буквы a из А создадим массив и добавим в него w + a

Шаг 2.1.1. Для каждого определяющего соотношения r из R

Шаг 2.1.1.1. Если возможно, преобразуем с помощью r слово w + a и обозначим его nw.

Шаг 2.1.1.2. Если nw есть в , выходим из цикла.

Шаг 2.1.1.3. Если длина nw меньше, чем длина w + a, добавляем nw в , k[w + a] = nw. Переходим к шагу 2.1.1 со словом nw.

Шаг 2.1.1.4. Иначе, если nw есть в sOfr, для всех слов x из : k[x] = k[nw];

Шаг 2.1.1.5. Иначе добавляем nw в и переходим к шагу 2.1.1 со словом nw.

Шаг 2.1.2. Все элементы из поместить в sOfr.

Шаг 2.1.3. Для всех элементов x из : k[x] = k[minw], где minw – слово из минимальной длины.

Шаг 2.1.4. w + a поместить в sOfr.

Шаг 2.1.5. minw поместить в res.

Шаг 3. Выводим полугруппу res.

Шаг 4. Заводим maxw = максимальная длина слова из res, s\_cur = res, s\_plus.

Шаг 5. maxw раз

Шаг 5.1. для каждого слова w из s\_cur

Шаг 5.1.1. Выполняем шаг 2.1.

Шаг 5.1.2. Добавляем w + a в s\_plus.

Шаг 5.2. Очищаем s\_cur.

Шаг 5.3. s\_cur = s\_plus.

Шаг 6. Выводим таблицу Кэли с помощью словаря k.

ВРЕМЕННАЯ СЛОЖНОСТЬ АЛГОРИТМОВ

Построение подполугруппы по таблице Кэли - O(n3)

Построение полугрупп бинарных отношений по заданному порождающему множеству -

Построение полугруппы по порождающему множеству и определяющим соотношениям – сложность алгоритма зависит от того, сколько элементов окажется в полугруппе, а так как у нас нет границы для количества этих элементов, сложность оценить не получится.

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

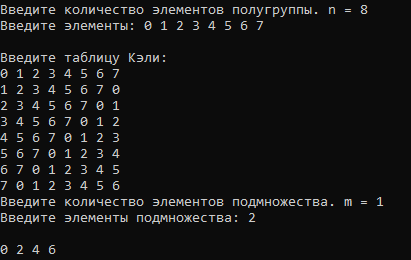


Рис.1 – первый тест

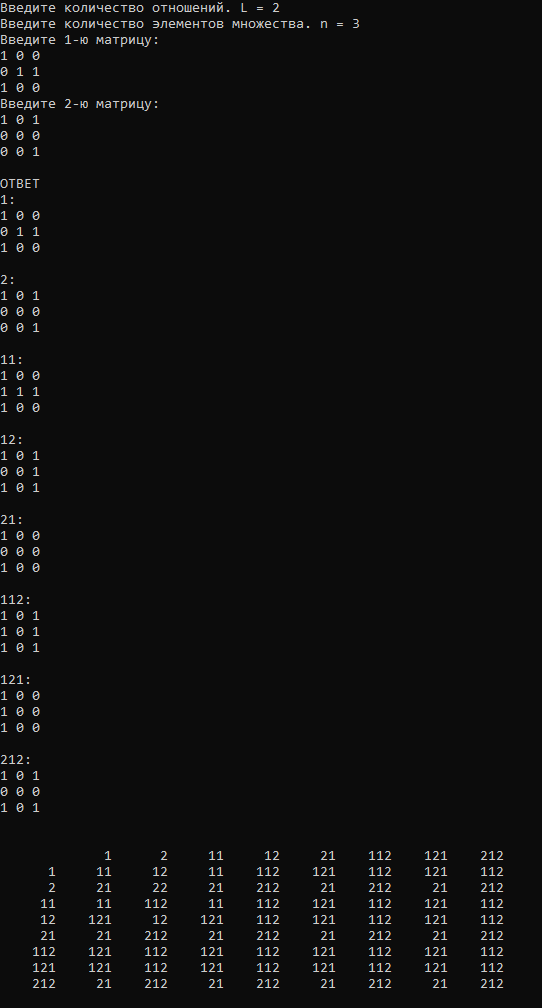


Рис.2 – второй тест

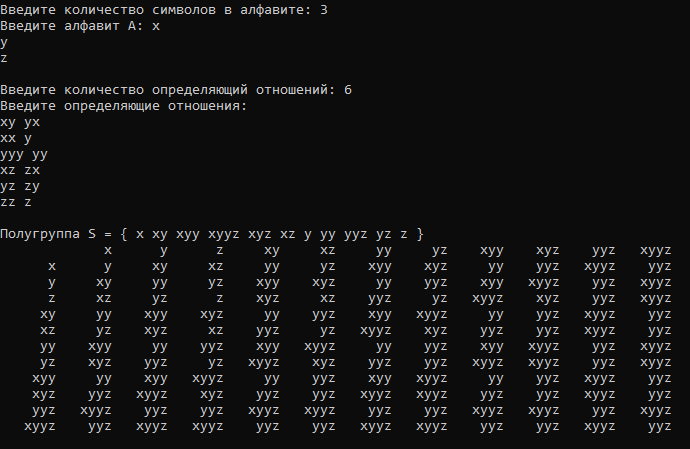
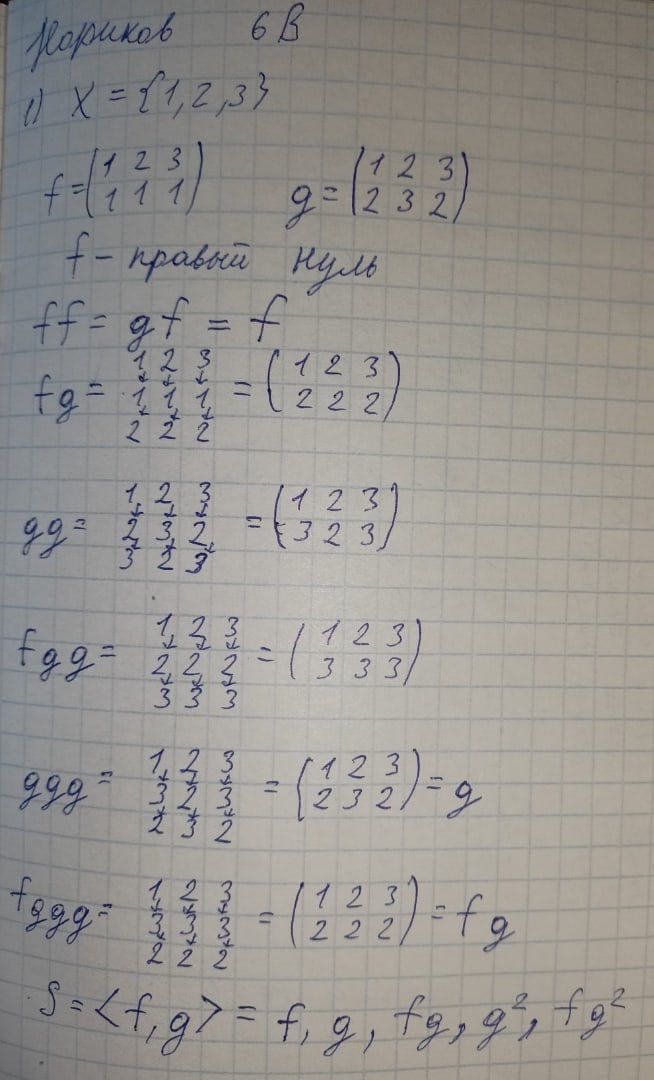
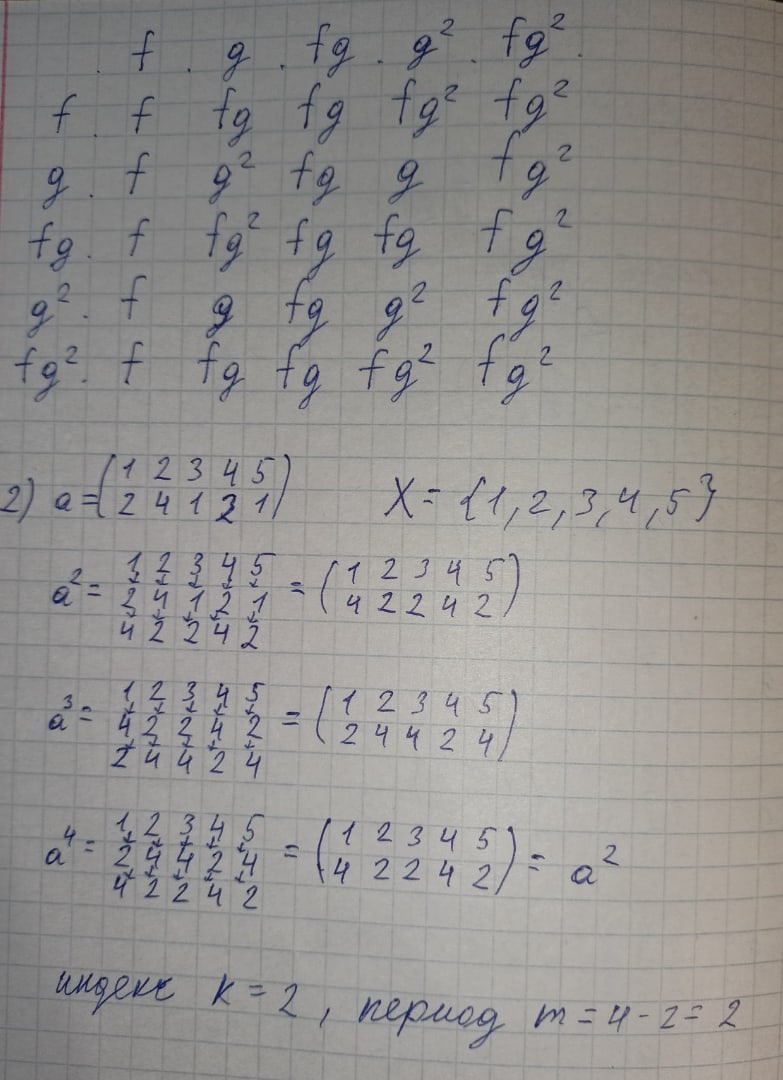
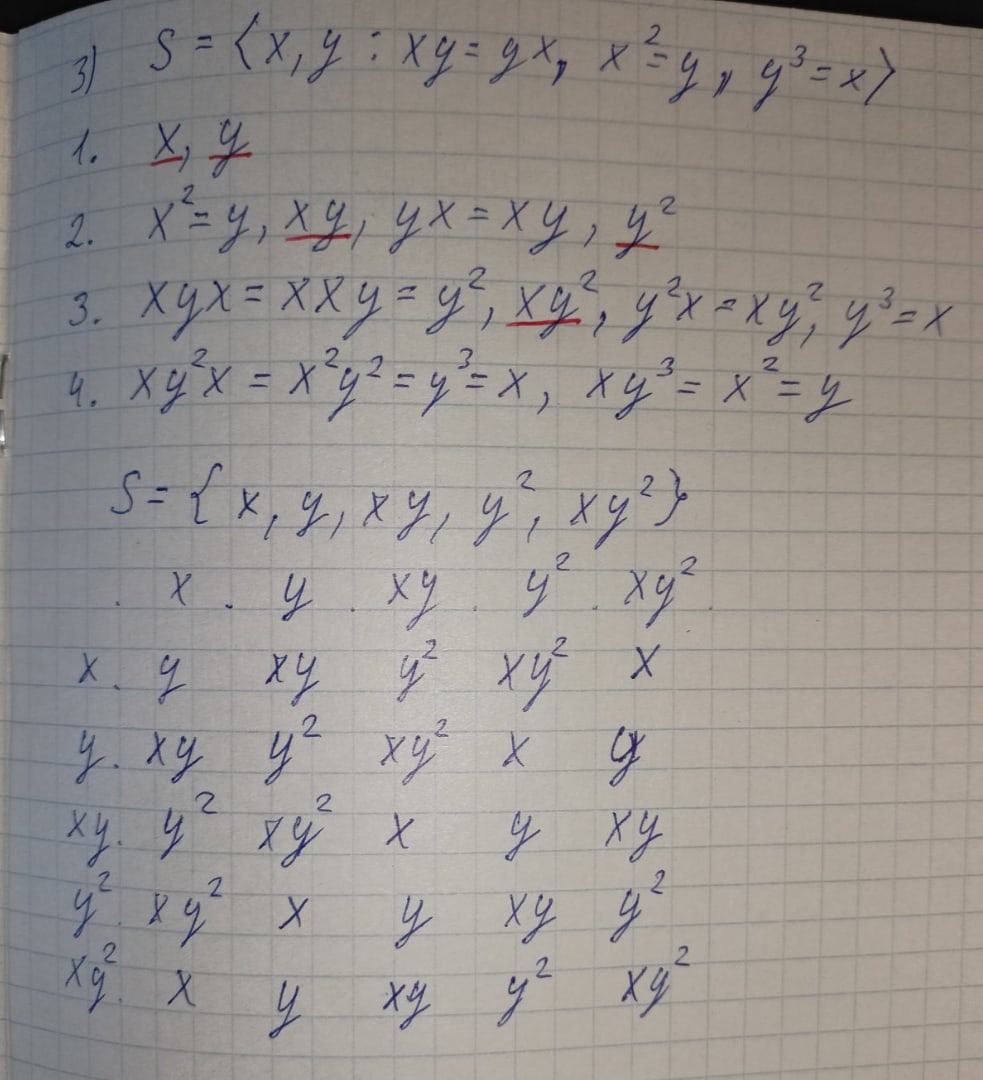


Рис.3 – третий тест

ПИСЬМЕННОЕ ЗАДАНИЕ







КОД ПРОГРАММЫ

#include <iostream>

#include <iomanip>

#include <vector>

#include <algorithm>

#include <set>

#include<map>

#include<string>

using namespace std;

bool comp(string s1, string s2) {

if (s1.size() < s2.size()) return true;

else if (s1.size() == s2.size() && s1 < s2) return true;

else return false;

}

bool comp2(pair <vector <vector <int>>, string> s1, pair <vector <vector <int>>, string> s2) {

if (s1.second.size() < s2.second.size()) return true;

else if (s1.second.size() == s2.second.size() && s1.second < s2.second) return true;

else return false;

}

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;

}

bool check\_uniq(vector <char> v, int n, char x) {

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

if (v[i] == x)

return false;

return true;

}

vector <char> find\_xi\_(char\*\* v, int n, vector <char> xi, int m, char\* els) {

vector <char> res;

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

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

char vij = v[find\_number(els, n, xi[i])][find\_number(els, n, xi[j])];

if (check\_uniq(res, res.size(), vij)) {

res.push\_back(vij);

}

}

}

return res;

}

vector <char> xi\_union(vector <char> xi, int n, vector <char> xi\_, int m) {

vector <char> res(n);

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

res[i] = xi[i];

}

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

if (check\_uniq(res, res.size(), xi\_[i]))

res.push\_back(xi\_[i]);

}

return res;

}

void alg\_1(char\*\* v, int n, vector <char> v0, int m, char\* els) {

vector <char> xi(m);

vector <char> xi\_;

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

xi[i] = v0[i];

while (true) {

xi\_ = find\_xi\_(v, n, xi, xi.size(), els);

vector <char> x\_new = xi\_union(xi, xi.size(), xi\_, xi\_.size());

if (xi.size() == x\_new.size())

break;

else

xi = x\_new;

}

sort(xi.begin(), xi.end());

cout << endl;

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

cout << xi[i] << " ";

}

}

map <pair<string, string>, string> kely;

vector <vector <int>> my\_multiplication(vector <vector <int>> v1, vector <vector <int>> v2, int n) {

vector <vector <int>> res(n);

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

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

int count = 0;

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

if (v1[i][k] \* v2[k][j]) {

count = 1;

break;

}

}

res[i].push\_back(count);

}

}

return res;

}

bool checker\_uniq(vector <pair <vector <vector <int>>, string>> v, int l, vector <vector <int>> x) {

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

if (v[i].first == x)

return false;

}

return true;

}

bool checker\_uniq\_kely(vector <pair <vector <vector <int>>, string>> v, int l, vector <vector <int>> x, string xi, string xj) {

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

if (v[i].first == x) {

kely[make\_pair(xi, xj)] = v[i].second;

return false;

}

}

return true;

}

vector <pair <vector <vector <int>>, string>> find\_multi(vector <pair <vector <vector <int>>, string>> v, int l, int n) {

vector <pair <vector <vector <int>>, string>> res;

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

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

vector <vector <int>> mult = my\_multiplication(v[i].first, v[j].first, n);

if (checker\_uniq\_kely(res, res.size(), mult, v[i].second, v[j].second)) {

res.push\_back(make\_pair(mult, v[i].second + v[j].second));

kely[make\_pair(v[i].second, v[j].second)] = v[i].second + v[j].second;

}

}

}

return res;

}

vector <pair <vector <vector <int>>, string>> find\_union\_matr(vector <pair <vector <vector <int>>, string>> v1, vector <pair <vector <vector <int>>, string>> v2, int l1, int l2, int n) {

vector <pair <vector <vector <int>>, string>> res = v1;

for (int i = 0; i < l2; i++) {

if (checker\_uniq(res, res.size(), v2[i].first)) {

res.push\_back(v2[i]);

}

}

return res;

}

void alg\_2(vector <pair <vector <vector <int>>, string>> matr, int l, int n) {

vector <pair <vector <vector <int>>, string>> xi = matr;

while (true) {

vector <pair <vector <vector <int>>, string>> xi\_ = find\_multi(xi, xi.size(), n);

vector <pair <vector <vector <int>>, string>> x\_new = find\_union\_matr(xi, xi\_, xi.size(), xi\_.size(), n);

if (xi.size() == x\_new.size())

break;

else

xi = x\_new;

}

sort(xi.begin(), xi.end(), comp2);

cout << endl << "ОТВЕТ" << endl;

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

cout << xi[i].second << ":" << endl;

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

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

cout << xi[i].first[j][k] << " ";

}

cout << endl;

}

cout << endl;

}

cout << endl;

cout << setw(7) << "";

for (pair <vector <vector <int>>, string> ans : xi)

cout << setw(7) << ans.second;

cout << endl;

for (pair <vector <vector <int>>, string> ans : xi) {

cout << setw(7) << ans.second << setw(7);

for (pair <vector <vector <int>>, string> ans2 : xi) {

pair<string, string> res = make\_pair(ans.second, ans2.second);

cout << kely[res] << setw(7);

}

cout << endl;

}

cout << endl;

}

set<string> setOfr;

map<string, string> r;

set<string> s, s\_cur, s\_tmp, s\_i\_tmp;

int mx = 0;

int mn = 0;

int cnt;

string cur\_word;

map<string, string> help;

void handler(string word, int check)

{

cur\_word = "";

if (word.size() >= 0) {

if (setOfr.find(word) != setOfr.end()) {

s\_i\_tmp.insert(word);

for (string x : s\_i\_tmp) {

if (x != word)

help[x] = word;

}

for (int i = 0; i < 1; i++) {

map <string, string> ::iterator it = help.begin();

while (it != help.end()) {

if (help[it->second] != "")

help[it->first] = help[it->second];

it++;

}

}

if (s.find(word) != s.end()) {

help[word] = word;

}

return;

}

s\_i\_tmp.insert(word);

help[word] = word;

string word\_ = word;

int tmp = mx;

while (tmp >= mn)

{

for (int i = word\_.size() - 1; i >= 0; i--)

{

cur\_word = "";

string part = "";

if (tmp > word.size() || tmp > i + 1)

break;

int j, j\_Start, j\_end;

for (j = i; j > i - tmp; j--) {

part = word\_[j] + part;

j\_Start = i - tmp + 1;

}

if (r.find(part) != r.end()) {

for (int k = 0; k < j\_Start; ++k)

cur\_word = cur\_word + word\_[k];

cur\_word = cur\_word + r[part];

for (int k = i + 1; k < word\_.size(); k++)

cur\_word = cur\_word + word\_[k];

if (s\_i\_tmp.find(cur\_word) == s\_i\_tmp.end()) {

if (cur\_word.size() < word.size())

{

s\_i\_tmp.insert(cur\_word);

help[word] = cur\_word;

for (int i = 0; i < 1; i++) {

map <string, string> ::iterator it = help.begin();

while (it != help.end()) {

if (help[it->second] != "")

help[it->first] = help[it->second];

it++;

}

}

handler(cur\_word, check + 1);

return;

}

else if (setOfr.find(cur\_word) != setOfr.end()) {

for (string x : s\_i\_tmp) {

help[x] = cur\_word;

for (int i = 0; i < 1; i++) {

map <string, string> ::iterator it = help.begin();

while (it != help.end()) {

if (help[it->second] != "")

help[it->first] = help[it->second];

it++;

}

}

}

s\_i\_tmp.insert(cur\_word);

return;

}

else {

s\_i\_tmp.insert(cur\_word);

handler(cur\_word, check + 1);

if (check > 0)

return;

else continue;

}

}

}

}

tmp--;

}

vector<string> res;

for (string ans : s\_i\_tmp) {

res.push\_back(ans);

}

sort(res.begin(), res.end(), comp);

for (string x : res) {

setOfr.insert(x);

}

string max\_s;

for (string x : res) {

max\_s = x;

break;

}

for (int i = 0; i < 1; i++) {

map <string, string> ::iterator it = help.begin();

while (it != help.end()) {

if (help[it->second] != "")

help[it->first] = help[it->second];

it++;

}

}

for (string x : res) {

if (x != max\_s)

help[x] = max\_s;

}

for (int i = 0; i < 2; i++) {

map <string, string> ::iterator it = help.begin();

while (it != help.end()) {

if (help[it->second] != "")

help[it->first] = help[it->second];

it++;

}

}

if (help[max\_s] != "")

s.insert(help[max\_s]);

setOfr.insert(word);

return;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int w = 999;

while (w > 0) {

cout << endl;

cout << "1 - алгоритм построения подполугрупп по таблице Кэли" << endl;

cout << "2 - алгоритм построения полугруппы бинарных отношений по заданному порождающему множеству" << endl;

cout << "3 - алгоритм построения полугруппы по порождающему множеству и определяющим соотношениям" << 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];

}

}

int m;

cout << "Введите количество элементов подмножества. m = ";

cin >> m;

vector <char> podmn(m);

cout << "Введите элементы подмножества: ";

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

cin >> podmn[i];

if (assoc(v1, els, n))

alg\_1(v1, n, podmn, m, els);

else

cout << "Операция НЕ обладает свойством ассоциативности" << endl;

}

if (w == 2) {

int l, n;

cout << "Введите количество отношений. L = ";

cin >> l;

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

cin >> n;

vector <pair <vector <vector <int>>, string>> matr;

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

cout << "Введите " << i + 1 << "-ю матрицу:" << endl;

vector <vector <int>> help1;

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

vector <int> help2;

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

int x;

cin >> x;

help2.push\_back(x);

}

help1.push\_back(help2);

}

matr.push\_back(make\_pair(help1, to\_string(i + 1)));

}

alg\_2(matr, l, n);

}

if (w == 3) {

int n, m;

set<string> a;

cout << "Введите количество символов в алфавите: ";

cin >> n;

cout << "Введите алфавит А: ";

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

{

string ai;

cin >> ai;

a.insert(ai);

s.insert(ai);

setOfr.insert(ai);

help[ai] = ai;

}

cout << endl << "Введите количество определяющий отношений: ";

cin >> m;

puts("Введите определяющие отношения:");

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

{

string r1, r2;

cin >> r1 >> r2;

if (r1.size() > mx)

mx = r1.size();

if (r2.size() > mx)

mx = r2.size();

if (r1.size() < r2.size()) {

r[r2] = r1;

help[r2] = r1;

s.insert(r1);

setOfr.insert(r1);

setOfr.insert(r2);

}

else if (r1.size() > r2.size()) {

r[r1] = r2;

help[r1] = r2;

s.insert(r2);

setOfr.insert(r2);

setOfr.insert(r1);

}

else

{

if (r1 < r2) {

r[r1] = r2;

r[r2] = r1;

help[r1] = r1;

help[r2] = r1;

s.insert(r1);

setOfr.insert(r1);

setOfr.insert(r2);

}

else {

r[r1] = r2;

r[r2] = r1;

help[r1] = r2;

help[r2] = r2;

s.insert(r2);

setOfr.insert(r2);

setOfr.insert(r1);

}

}

}

cnt = 1;

s\_cur = a;

while (!s\_cur.empty())

{

cnt++;

for (string si : s\_cur)

{

for (string letter : a)

{

s\_i\_tmp.clear();

string si\_ = si + letter;

cur\_word = "";

handler(si\_, 0);

}

}

s\_cur.clear();

for (auto ss : s)

if (ss.size() == cnt)

s\_cur.insert(ss);

}

vector<string> res;

for (string ans : s) {

res.push\_back(ans);

}

sort(res.begin(), res.end(), comp);

cout << endl << "Полугруппа S = { ";

for (string ans : s)

cout << ans << " ";

cout << "}" << endl;

map <string, string> ::iterator it = help.begin();

it = help.begin();

int max\_size = 0;

for (string str : res) {

max\_size = str.size();

}

s\_cur = s;

set<string> s\_plus;

while (max\_size > 0) {

s\_plus.clear();

for (string si : s\_cur)

{

for (string letter : a)

{

s\_i\_tmp.clear();

string si\_ = si + letter;

handler(si\_, 0);

s\_plus.insert(si\_);

}

}

s\_cur.clear();

for (auto ss : s\_plus) {

if (max\_size > 0) {

s\_cur.insert(ss);

}

}

max\_size--;

}

int for\_res = 0;

cout << setw(7) << "";

for (string ans : res)

cout << setw(7) << ans;

cout << endl;

for (string ans : res) {

cout << setw(7) << ans << setw(7);

for (string ans2 : res) {

cur\_word = (ans + ans2);

cout << help[cur\_word] << setw(7);

}

cout << endl;

}

cout << endl;

}

}

return 0;

}

ЗАКЛЮЧЕНИЕ

В ходе данной работы работе были определены понятия полугруппы, подполугруппы и порождающего множества. Были разработаны алгоритмы построения подполугрупп по таблице Кэли и построения полугруппы бинарных отношений по заданному бинарному отношению. Также был изучен материал о подгруппах и определяющих соотношениях. По этой теме был разработан алгоритм построения полугруппы по порождающему множеству и определяющим соотношениям.