| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |
| --- | --- |

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

Студент Гусаров Аркадий Андреевич

Группа РК6-53Б

Тип задания Лабораторная работа №3

Тема лабораторной работы Системы различных представлений

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гусаров А.А.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волосатова Т.М**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Родионов С.В**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2021 г.*

# Задание на лабораторную работу

В заданной [0,1] – матрице М, размеров 6х6, требуется выбрать по одной 1 последовательно из каждой ее строки сверху вниз, которые должны иметь попарно различные индексы своих столбцов.

# Цель выполнения лабораторной работы

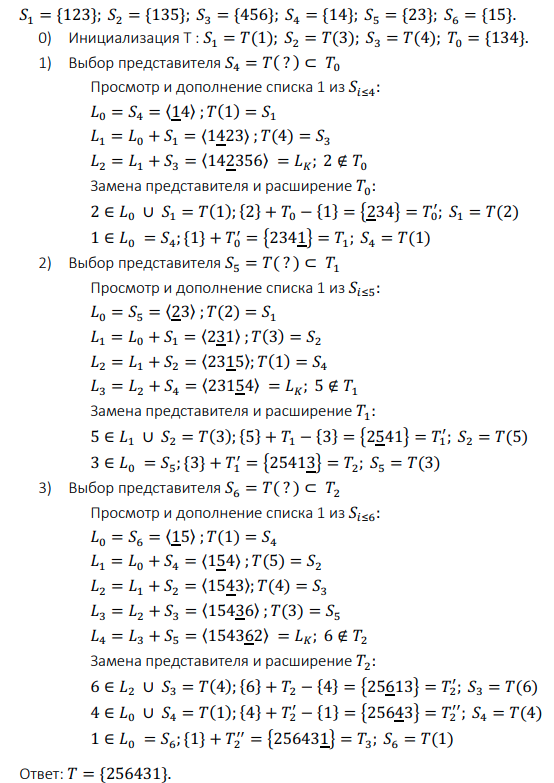
**Цель** – ознакомление и применение на практике алгоритма Холла.

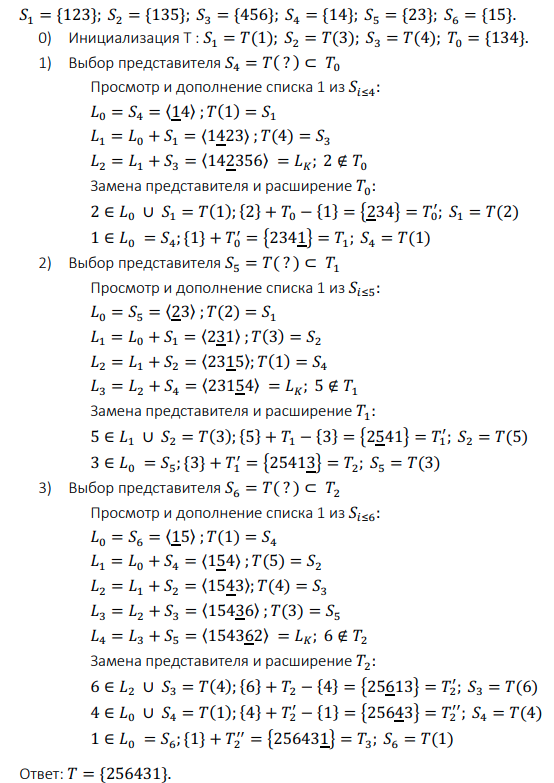
# Теоретическая часть

Для решения проблемы выбора следует применить алгоритм Холла, чтобы найти систему различных представителей Т семейства S подмножеств номеров единиц в каждой строке матрицы М от 1 до 6. Последовательное выполнение итераций алгоритма Холла должна сопровождать спецификация операций его формальных процедур.

# Пример алгоритма

Семейство подмножеств номеров 1 по строкам [0,1] матрицы М:





# 

# Код программы

*Файл main.cpp:*

#include "matrix.cpp"

#include <iostream>

#include <vector>

#include <map>

using namespace std;

vector<vector<int>> create\_families(int \*\*matrix, int rows, int cols)

{

vector<vector<int>> S;

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

{

vector<int> family;

for (int j = 0; j < cols; j++)

{

if (matrix[i][j] == 1)

{

family.push\_back(j + 1);

}

}

S.push\_back(family);

}

return S;

}

bool elem\_in\_vector(int x, vector<int> v)

{

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

{

if (x == v[i])

{

return true;

}

}

return false;

}

vector<int> create\_T0(vector<vector<int>> S)

{

vector<int> T0;

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

{

bool set\_new\_elem = true;

bool can\_have\_elem = false;

for (int j = 0; j < S[i].size() && set\_new\_elem; j++)

{

if (!elem\_in\_vector(S[i][j], T0))

{

T0.push\_back(S[i][j]);

set\_new\_elem = false;

can\_have\_elem = true;

}

else

{

can\_have\_elem = false;

}

}

if (!can\_have\_elem)

{

return T0;

}

}

return T0;

}

void print\_vector(vector<int> v)

{

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

{

cout << v[i] << ' ';

}

cout << endl;

}

vector<int> stick\_vectors(vector<int> L, vector<int> S)

{

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

{

if (!elem\_in\_vector(S[i], L))

{

L.push\_back(S[i]);

}

}

return L;

}

int index\_in\_vector(int x, vector<int> v)

{

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

{

if (x == v[i])

{

return i;

}

}

return -1;

}

int index\_in\_added\_families(map<int, vector<int>> &added\_families, int x, int after = -1)

{

int i = 0;

for (auto it = added\_families.begin(); it != added\_families.end(); it++)

{

i = it->first;

if (i > after)

{

if (elem\_in\_vector(x, added\_families[i]))

{

return i;

}

}

}

return -1;

}

vector<int> finding\_of\_next\_elem(vector<int> &T, vector<vector<int>> &S)

{

vector<int> L;

vector<int> T\_new = T;

int next\_agent = T.size();

L = S[next\_agent]; // L\_0

cout << endl

<< "L0:" << endl;

print\_vector(L);

int i = 0;

int j = index\_in\_vector(L[i], T);

map<int, vector<int>> added\_families;

while (j != -1)

{

added\_families[j] = S[j];

L = stick\_vectors(L, S[j]);

cout << endl

<< "L" << i + 1 << ":" << endl;

print\_vector(L);

i++;

j = index\_in\_vector(L[i], T);

}

int new\_agent = L[i];

int agent\_to\_swap = 0;

int replace\_index = -1;

while (!elem\_in\_vector(new\_agent, S[next\_agent]))

{

replace\_index = index\_in\_added\_families(added\_families, new\_agent, replace\_index);

agent\_to\_swap = T\_new[replace\_index];

T\_new[replace\_index] = new\_agent;

new\_agent = agent\_to\_swap;

}

T\_new.push\_back(new\_agent);

cout << endl

<< "T" << endl;

print\_vector(T\_new);

return T\_new;

}

int main()

{

int rows = 0, cols = 0;

int \*\*matrix = read\_matrix\_from\_file(rows, cols);

print\_matrix(matrix, rows, cols);

vector<vector<int>> S = create\_families(matrix, rows, cols);

cout << endl

<< "Семейства S:" << endl;

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

{

cout << "S" << i + 1 << ": ";

print\_vector(S[i]);

}

vector<int> T = create\_T0(S);

cout << endl

<< "T0:" << endl;

print\_vector(T);

for (int i = 0; T.size() != rows; i++)

{

cout << endl

<< i + 1 << " итерация:" << endl;

T = finding\_of\_next\_elem(T, S);

}

delete\_matrix(matrix, rows);

return 0;

}

*Файл matrix.cpp:*

#include <iostream>

#include <fstream>

#include <new>

using namespace std;

#define filename "matrix.txt"

int \*\*create\_matrix(int rows, int cols)

{

int \*\*matrix = new int \*[rows];

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

{

matrix[i] = new int[cols];

}

return matrix;

}

void delete\_matrix(int \*\*matrix, int rows)

{

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

{

delete[] matrix[i];

}

}

// Формат файла: число строк, число столбцов, матрица

int \*\*read\_matrix\_from\_file(int &rows, int &cols)

{

ifstream fin(filename);

if (!fin.is\_open())

{

cout << "Неверное имя файла" << endl;

}

fin >> rows >> cols;

int \*\*matrix = create\_matrix(rows, cols);

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

{

for (int j = 0; j < cols; j++)

{

fin >> matrix[i][j];

}

}

fin.close();

return matrix;

}

void print\_matrix(int \*\*matrix, int rows, int cols)

{

cout << "Входная матрица:" << endl;

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

{

for (int j = 0; j < cols; j++)

{

cout << matrix[i][j] << ' ';

}

cout << endl;

}

}

*Файл matrix.txt:*

6

6

1 0 0 0 1 1

0 1 1 1 0 0

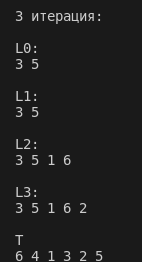
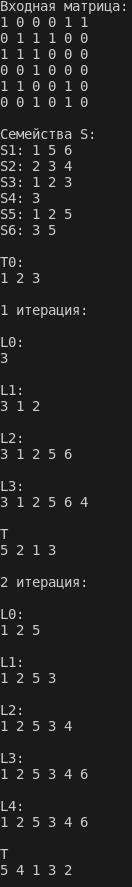
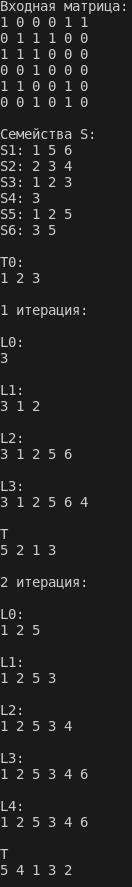
1 1 1 0 0 0

0 0 1 0 0 0

1 1 0 0 1 0

0 0 1 0 1 0

# Результат работы программы



# Список использованных источников

1. *Волосатова Т.М. курс лекций по дисциплине «Методы комбинаторных вычислений».*