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 M, размеров 6x6, 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 T семейства S подмножеств номеров единиц в каждой строке матрицы M от 1 до 6. Последовательное выполнение итераций алгоритма Холла должна сопровождать спецификация операций его формальных процедур.

Пример алгоритма

1	1	1	0	0	0
1	0	1	0	1	0
0	0	0	1	1	1
1	0	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	0

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

$S_1 = \{123\};\ S_2 = \{135\};\ S_3 = \{456\};\ S_4 = \{14\};\ S_5 = \{23\};\ S_6 = \{15\}.$

0) Инициализация Т : $S_1 = T(1);\ S_2 = T(3);\ S_3 = T(4);\ T_0 = \{134\}.$

1) Выбор представителя $S_4 = T(?) \subset T_0$

Просмотр и дополнение списка 1 из $S_{i\leq 4}$:

$L_0 = S_4 = \langle \underline{1}4\rangle ; T(1) = S_1$

$L_1 = L_0 + S_1 = \langle 1\underline{4}23\rangle ; T(4) = S_3$

$L_2 = L_1 + S_3 = \langle 14\underline{2}356\rangle = L_K;\ 2 \notin T_0$

Замена представителя и расширение T_0:

$2 \in L_0 \cup S_1 = T(1); \{2\} + T_0 - \{1\} = \{\underline{2}34\} = T_0';\ S_1 = T(2)$

$1 \in L_0 = S_4; \{1\} + T_0' = \{234\underline{1}\} = T_1;\ S_4 = T(1)$

2) Выбор представителя $S_5 = T(?) \subset T_1$

Просмотр и дополнение списка 1 из $S_{i\leq 5}$:

$L_0 = S_5 = \langle \underline{2}3\rangle ; T(2) = S_1$

$L_1 = L_0 + S_1 = \langle 2\underline{3}1\rangle ; T(3) = S_2$

$L_2 = L_1 + S_2 = \langle 23\underline{1}5\rangle ; T(1) = S_4$

$L_3 = L_2 + S_4 = \langle 231\underline{5}4\rangle = L_K;\ 5 \notin T_1$

Замена представителя и расширение T_1:

$5 \in L_1 \cup S_2 = T(3); \{5\} + T_1 - \{3\} = \{\underline{2}541\} = T_1';\ S_2 = T(5)$

$3 \in L_0 = S_5; \{3\} + T_1' = \{2541\underline{3}\} = T_2;\ S_5 = T(3)$

3) Выбор представителя $S_6 = T(?) \subset T_2$

Просмотр и дополнение списка 1 из $S_{i\leq 6}$:

$L_0 = S_6 = \langle \underline{1}5\rangle ; T(1) = S_4$

$L_1 = L_0 + S_4 = \langle 1\underline{5}4\rangle ; T(5) = S_2$

$L_2 = L_1 + S_2 = \langle 15\underline{4}3\rangle ; T(4) = S_3$

$L_3 = L_2 + S_3 = \langle 154\underline{3}6\rangle ; T(3) = S_5$

$L_4 = L_3 + S_5 = \langle 1543\underline{6}2\rangle = L_K;\ 6 \notin T_2$

Замена представителя и расширение T_2:

$6 \in L_2 \cup S_3 = T(4); \{6\} + T_2 - \{4\} = \{25\underline{6}13\} = T_2';\ S_3 = T(6)$

$4 \in L_0 \cup S_4 = T(1); \{4\} + T_2' - \{1\} = \{256\underline{4}3\} = T_2'';\ S_4 = T(4)$

$1 \in L_0 = S_6; \{1\} + T_2'' = \{25643\underline{1}\} = T_3;\ S_6 = T(1)$

Ответ: $T = \{256431\}.$

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

```
Входная матрица:
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

Семейства S:
S1: 1 5 6
S2: 2 3 4
S3: 1 2 3
S4: 3
S5: 1 2 5
S6: 3 5

T0:
1 2 3

1 итерация:

L0:
3

L1:
3 1 2

L2:
3 1 2 5 6

L3:
3 1 2 5 6 4

T
5 2 1 3
```

```
2 итерация:

L0:
1 2 5

L1:
1 2 5 3

L2:
1 2 5 3 4

L3:
1 2 5 3 4 6

L4:
1 2 5 3 4 6

T
5 4 1 3 2
```

```
3 итерация:

L0:
3 5

L1:
3 5

L2:
3 5 1 6

L3:
3 5 1 6 2

T
6 4 1 3 2 5
```

Список использованных источников

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