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	Никифорова Ирина Андреевна
Группа	РК6-71б
Тип задания	лабораторная работа
Тема лабораторной работы	Системы различных представителей
Вариант	Т14

Студент _____ **Никифорова И. А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Родионов С. В.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2019 г.

Оглавление

Задание на лабораторную работу

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

В заданной матрице M, состоящей из нулей и единиц, размером 6x6 требуется выбрать по одной единице последовательно из каждой ее строки сверху вниз так, чтобы они имели попарно различные индексы своих столбцов.

Для решения этой проблемы выбора следует применять 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.

Алгоритм Холла

Пусть задано множество E из n произвольных элементов. В решаемой задаче это n номеров столбцов, в которых могут стоять единицы:

$E = \{ 1, 2, 3, 4, 5, 6 \}, | E | = n = 6$

Пусть также задано семейство S из $m \leq n$ его непустых и необязательно различных подмножеств. В задаче, каждое подмножество семейства состоит из номеров столбцов с единицами в каждой строке, $m = 6$ - число строк.

Необходимо найти систему различных представителей (СРП) или, по-другому, трансверсаль, где представители выбираются для каждого подмножества (строки).

Перед применением алгоритма можно проверить выполнение теоремы Холла, определяющей, может ли быть построена полная трансверсаль: сочетание любых k подмножеств семейства S должно содержать не менее, чем k различных элементов базового множества E - это необходимое и достаточное условие. Обычно теорему Холла не проверяют, потому что ее проверка слишком затратна, а алгоритм Холла может быть применен и без этого.

Алгоритм Холла заключается в итерационном выборе и перевыборе представителей подмножеств семейства S. Его итерации можно описать следующим образом:

0. Инициализация начальной трансверсали T_0 . В нее выбираются поочередно представители подмножеств, начиная с первого и далее. Выбираются всегда минимально допустимые представители так, чтобы они не были использованы ранее в этой трансверсали. Как только представителя выбрать нельзя, если все элементы соответствующего подмножества уже задействованы, инициализация прекращается, а номер текущего подмножества запоминается.

1. Выбор следующего представителя. На этой итерации выбирается представитель для множества, на котором остановился предыдущий шаг.

1.1. Просмотр и дополнение списка L. В список L заносятся все элементы текущего множества. Далее, все его элементы поочередно проверяются на то, представлены ли они уже в трансверсали предыдущего шага. Если представлены, то множество, представителем которого этот элемент является, включается в конец списка L, без повторения уже внесенных туда элементов. Если же рассматриваемого элемента нет в предыдущей трансверсали, то он запоминается начинается следующий этап.

1.2. Замена представителя и расширение трансверсали T. Запомненный на предыдущем шаге элемент включается в T на место представителя того множества, с включением которого он попал в L. Исключенный элемент таким же образом снова включается в T, но уже на место другого представителя и так далее, пока не будет встречен элемент, изначально входящий в L (L_0). Когда он будет встречен, он просто добавляется в конец трансверсали.

2. Итерация 1 повторяется поочередно для всех следующих представителей до тех пор, пока не будет получена итоговая полная трансверсаль либо не будет достигнута итерация на которой формирование продолжить не удастся. Частичная трансверсаль будет итоговой, если например, в списке L (в конечном его варианте) окажутся только представители предыдущей трансверсали.

Решение задачи

	1	2	3	4	5	6
1	0	0	0	1	1	1
2	1	1	1	0	0	0
3	1	1	0	0	1	0
4	0	0	0	1	0	0
5	1	0	0	1	1	0
6	0	1	0	0	0	0

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

$S_1 = \{4, 5, 6\}$, $S_2 = \{1, 2, 3\}$,

$S_3 = \{1, 2, 5\}$, $S_4 = \{4\}$,

$S_5 = \{1, 4, 5\}$, $S_6 = \{2\}$

0. Инициализация T: $S_1 = T(4)$, $S_2 = T(1)$, $S_3 = T(2)$; $\underline{T_0 = \{4, 1, 2\}}$

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

1.1. Просмотр и дополнение списка 1 из $S_{i \le 4}$:

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

$L_1 = L_0 + S_1 = \{4\underline{5}6\} = L_k$; $5 \notin T_0$

1.2. Замена представителя и расширение T_0:

$5 \in L_0 \cup S_1 = T(4)$; $\{5\} + T_0 - \{4\} = \{\underline{5}12\} = T_0'$; $S_1 = T(5)$

$4 \in L_0 = S_4$; $\{4\} + T_0' = \underline{\{5124\} = T_1}$; $S_4 = T(4)$

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

2.1. Просмотр и дополнение списка 1 из $S_{i \le 5}$:

$L_0 = S_5 = \{\underline{1}45\}$; $T(1) = S_2$

$L_1 = L_0 + S_2 = \{1\underline{4}523\}$; $T(4) = S_4$

$L_2 = L_1 + S_4 = \{14\underline{5}23\}$; $T(5) = S_1$

$L_3 = L_2 + S_1 = \{145\underline{2}36\}$; $T(2) = S_3$

$L_4 = L_3 + S_3 = \{145236\} = L_k;\ 3 \notin T_1$

2.2. Замена представителя и расширение T_1:

$3 \in L_0 \cup S_2 = T(1);\ \{3\} + T_1 - \{1\} = \{5324\} = T_1';\ S_2 = T(3)$

$1 \in L_0 = S_5;\ \{1\} + T_1' = \underline{\{53241\} = T_2};\ S_5 = T(1)$

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

3.1. Просмотр и дополнение списка 1 из $S_{i \le 6}$:

$L_0 = S_6 = \{2\};\ T(2) = S_3$

$L_1 = L_0 + S_3 = \{215\};\ T(1) = S_5$

$L_2 = L_1 + S_5 = \{2154\};\ T(5) = S_1$

$L_3 = L_2 + S_1 = \{21546\};\ T(4) = S_4$

$L_4 = L_3 + S_4 = \{215466\} = L_k;\ 6 \notin T_2$

3.2. Замена представителя и расширение T_2:

$6 \in L_1 \cup S_1 = T(5);\ \{6\} + T_2 - \{5\} = \{63241\} = T_2';\ S_1 = T(6)$

$5 \in L_0 \cup S_3 = T(2);\ \{5\} + T_2' - \{2\} = \{63541\} = T_2'';\ S_3 = T(5)$

$2 \in L_0 = S_6;\ \{2\} + T_2'' = \underline{\{635412\} = T_3};\ S_6 = T(2)$

Ответ: $\underline{T = \{635412\}}$

	1	2	3	4	5	6
1	0	0	0	1	1	**1**
2	1	1	**1**	0	0	0
3	1	1	0	0	**1**	0
4	0	0	0	**1**	0	0
5	**1**	0	0	1	1	0
6	0	**1**	0	0	0	0

Листинг программы

```
#include "matrix.hpp"
#include <iostream>
#include <vector>
#include <algorithm>
#include <map>

#define rows 6
#define cols 6

using namespace std;

bool is_in_vector(int x, vector<int>& v) {
    if (find(v.begin(), v.end(), x) == v.end()) {
        return false;
    }

    return true;
}

int index_in_vector(int x, vector<int>& v) {
    auto it = find(v.begin(), v.end(), x);

    if (it == v.end()) {
        return -1;
    }

    return distance(v.begin(), it);
}

vector<int> sum_vectors(vector<int> original, vector<int>& additional) {
    vector<int> sum = original;
```

```
    for (int a : additional) {
        if (!is_in_vector(a, original)) {
            sum.push_back(a);
        }
    }

    return sum;
}

void print_vector(vector<int>& v, string name) {
    cout << name << " = { ";
    int i;
    for (i = 0; i < v.size() - 1; i++) {
        cout << v[i] << ", ";
    }
    cout << v[i] << " }" << endl;
}

Matrix<int> create_input_matrix() {
    int input_matrix[rows][cols] = {
        {0, 0, 0, 1, 1, 1},
        {1, 1, 1, 0, 0, 0},
        {1, 1, 0, 0, 1, 0},
        {0, 0, 0, 1, 0, 0},
        {1, 0, 0, 1, 1, 0},
        {0, 1, 0, 0, 0, 0},
    };

    Matrix<int> A(*input_matrix, rows, cols);
    A.print("Input matrix:");

    return A;
}
```

```
vector< vector<int> > turn_to_families(Matrix<int>& A) {
    vector< vector<int> > S;

    for (int i = 0; i < A.get_rows(); i++) {
        vector<int> current_family;
        for (int j = 0; j < A.get_cols(); j++) {
            if (A.get(i, j) == 1) {
                current_family.push_back(j + 1);
            }
        }
        S.push_back(current_family);
    }

    cout << "Families:\n";
    for (int i = 0; i < S.size(); i++) {
        print_vector(S[i], "S_" + to_string(i));
    }
    cout << endl;

    return S;
}

vector<int> create_T_0(vector <vector<int> >& S) {
    vector<int> T_0;

    bool cant_set_next = false;
    for (int i = 0; i < S.size() && !cant_set_next; i++) {
        int current_agent_i = 0;
        while(!cant_set_next && is_in_vector(S[i][current_agent_i], T_0)) {
            current_agent_i++;
            if(current_agent_i >= S[i].size()) {
                cant_set_next = true;
```

```
            }
        }
        if (!cant_set_next) {
            T_0.push_back(S[i][current_agent_i]);
        }
    }

    cout << "First transversal: \n";
    print_vector(T_0, "T_0");
    cout << endl;

    return T_0;
}

int index_in_add_families(map<int, vector<int> >& add_families, int x, int
after = -1) {
    int i = 0;
    for (auto it = add_families.begin(); it != add_families.end(); it++) {
        i = it->first;
        if (i > after) {
            if (is_in_vector(x, add_families[i])) {
                return i;
            }
        }
    }
    return -1;
}

vector<int> find_next_agent(vector<int>& T, vector< vector<int> >& S) {
    int next_agent_i = T.size();

    cout << endl << "1. Viewing and forming of L-list:" << endl;

    vector<int> L = S[next_agent_i];
```

```
    print_vector(L, "L_0");

    int j = 0;
    int p = index_in_vector(L[j], T);
    map<int, vector<int> > add_families;
    while (p != -1) {
        add_families[p] = S[p];
        L = sum_vectors(L, S[p]);
        print_vector(L, "L_" + to_string(j + 1));

        j++;
        p = index_in_vector(L[j], T);
    }
    cout << L[j] << " ∉ T" << endl;
    cout << endl;

    cout << "2. Replacement of agent:" << endl;
    vector<int> T_new = T;


    int agent = L[j], agent_swap = 0;
    int replace_i = -1;
    while (!is_in_vector(agent, S[next_agent_i])) {
        replace_i = index_in_add_families(add_families, agent, replace_i);
        agent_swap = T_new[replace_i];
        T_new[replace_i] = agent;

        cout << "{ " << agent << " } + T - { " << agent_swap << " }" <<
endl;

        agent = agent_swap;

        print_vector(T_new, "T");
        cout << endl;
```

```
        }

        cout << "{ " << agent << " } + T" << endl;
        T_new.push_back(agent);

        print_vector(T_new, "T");
        cout << endl;

        return T_new;
    }

    int main() {
        Matrix<int> A = create_input_matrix();
        vector< vector<int> > S = turn_to_families(A);
        vector<int> T = create_T_0(S);

        for (int i = 1; T.size() != rows; i++) {
            cout << "Iteration #" << i << ':' << endl;;
            T = find_next_agent(T, S);
        }

        A.print("Result:", &T);

        return 0;
    }
```

Результаты работы программы.

Input matrix:
| 0 0 0 1 1 1 |
| 1 1 1 0 0 0 |
| 1 1 0 0 1 0 |
| 0 0 0 1 0 0 |
| 1 0 0 1 1 0 |
| 0 1 0 0 0 0 |

Families:
S_0 = { 4, 5, 6 }
S_1 = { 1, 2, 3 }
S_2 = { 1, 2, 5 }
S_3 = { 4 }
S_4 = { 1, 4, 5 }
S_5 = { 2 }

First transversal:
T_0 = { 4, 1, 2 }

Iteration #1:

1. Viewing and forming of L-list:
L_0 = { 4 }
L_1 = { 4, 5, 6 }
5 ∉ T

2. Replacement of agent:
{ 5 } + T - { 4 }
T = { 5, 1, 2 }

{ 4 } + T
T = { 5, 1, 2, 4 }

Iteration #2:

1. Viewing and forming of L-list:
L_0 = { 1, 4, 5 }
L_1 = { 1, 4, 5, 2, 3 }
L_2 = { 1, 4, 5, 2, 3 }
L_3 = { 1, 4, 5, 2, 3, 6 }
L_4 = { 1, 4, 5, 2, 3, 6 }
3 ∉ T

2. Replacement of agent:
{ 3 } + T - { 1 }
T = { 5, 3, 2, 4 }

{ 1 } + T
T = { 5, 3, 2, 4, 1 }

Iteration #3:

1. Viewing and forming of L-list:
L_0 = { 2 }
L_1 = { 2, 1, 5 }
L_2 = { 2, 1, 5, 4 }
L_3 = { 2, 1, 5, 4, 6 }
L_4 = { 2, 1, 5, 4, 6 }
6 ∉ T

2. Replacement of agent:
{ 6 } + T - { 5 }
T = { 6, 3, 2, 4, 1 }

{ 5 } + T - { 2 }
T = { 6, 3, 5, 4, 1 }

{ 2 } + T
T = { 6, 3, 5, 4, 1, 2 }

Result:
| 0 0 0 1 1 _1_ |
| 1 1 _1_ 0 0 0 |
| 1 1 0 0 _1_ 0 |
| 0 0 0 _1_ 0 0 |
| _1_ 0 0 1 1 0 |
| 0 _1_ 0 0 0 0 |