

**Министерство образования и науки Российской Федерации**

**Федеральное агентство по образованию**

**Государственное образовательное учреждение высшего профессионального образования**

**«Московский государственный технический университет имени Н.Э. Баумана»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Робототехника и комплексная автоматизация» (РК)**

**Кафедра «Системы автоматизированного проектирования» (РК6)**

****

**Отчет по лабораторной работе №3 по курсу**

**«Методы комбинаторных вычислений»**

**Студент: Петраков Станислав**

**Группа:** РК6-56Б

**Преподаватель:** Волосатова Т.М.

Проверил:

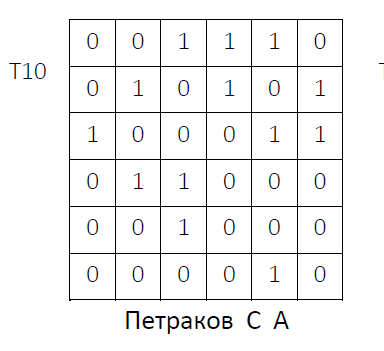
Дата:

2021 год

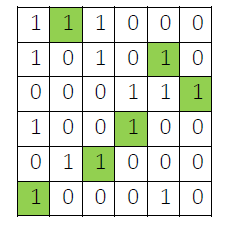
**T10 вариант**

**Задание:**

В заданной матрице размеров 6х6, требуется выбрать по одной 1 последовательно из каждой ее строки сверху вниз, которые должны иметь попарно различные индексы своих столбцов. Для решения этой 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.

****

**Описание алгоритма:**

Рассмотрим пример с матрицей:

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

𝑆1 = {123}; 𝑆2 = {135}; 𝑆3 = {456}; 𝑆4 = {14}; 𝑆5 = {23}; 𝑆6 = {15}.

0) Инициализация T : 𝑆1 = 𝑇(1); 𝑆2 = 𝑇(3); 𝑆3 = 𝑇(4); 𝑇0 = {134}.

1) Выбор представителя 𝑆4 = 𝑇( ? ) ⊂ 𝑇0

Просмотр и дополнение списка 1 из :

𝐿0 = 𝑆4 = 〈14〉 ; 𝑇(1) = 𝑆1

𝐿1 = 𝐿0 + 𝑆1 = 〈1423〉 ; 𝑇(4) = 𝑆3

𝐿2 = 𝐿1 + 𝑆3 = 〈142356〉 = 𝐿k; 2 ∉ 𝑇0

Замена представителя и расширение 𝑇0:

2 ∈ 𝐿0 ∪ 𝑆1 = 𝑇(1); {2} + 𝑇0 − {1} = {432} = ; 𝑆3 = 𝑇(2)

1 ∈ 𝐿0 = 𝑆4; {1} + = {4321} = 𝑇1; 𝑆4 = 𝑇(1)

2) Выбор представителя 𝑆5 = 𝑇( ? ) ⊂ 𝑇1

Просмотр и дополнение списка 1 из :

𝐿0 = 𝑆5 = 〈23〉 ; 𝑇(2) = 𝑆1

𝐿1 = 𝐿0 + 𝑆1 = 〈231〉 ; 𝑇(4) = 𝑆1

𝐿2 = 𝐿0 + 𝑆1 = 〈2315〉 ; 𝑇(4) = 𝑆1

𝐿3 = 𝐿1 + 𝑆2 = 〈23154〉= 𝐿k; 6 ∉ 𝑇1

Замена представителя и расширение 𝑇1:

5 ∈ 𝐿1 ∪ 𝑆1 = 𝑇(6); {6} + 𝑇1 − {4} = {2541} = ; 𝑆1 = 𝑇(6)

3 ∈ 𝐿0 ∪ 𝑆2 = 𝑇(4); {4} + 𝑇1− {3} = {25413} = 𝐿k; 3 ∉ 𝑇2

3) Выбор представителя 𝑆6 = 𝑇( ? ) ⊂ 𝑇2

Просмотр и дополнение списка 1 из:

𝐿0 = 𝑆6 = 〈12〉 ; 𝑇(1) = 𝑆4

𝐿1 = 𝐿0 + 𝑆4 = 〈124〉 ; 𝑇(2) = 𝑆3

𝐿2 = 𝐿1 + 𝑆3 = 〈1243〉; 𝑇(4) = 𝑆2

𝐿3 = 𝐿2 + 𝑆2 = 〈12436〉 ; 𝑇(3) = 𝑆5

𝐿4 = 𝐿3 + 𝑆5 = 〈124365〉 = 𝐿k; 5 ∉ 𝑇2

Замена представителя и расширение 𝑇2:

5 ∈ 𝐿2 ∪ 𝑆3 = 𝑇(4); {6} + 𝑇2 − {4} = {25613} = 𝑇2; 𝑆3 = 𝑇(6)

6 ∈ 𝐿0 ∪ 𝑆4 = 𝑇(1); {4} + 𝑇2− {1} = {25643} = 𝑇2; 𝑆4 = 𝑇(4)

2 ∈ 𝐿0 ∪ 𝑆4 = 𝑇(1); {4} + 𝑇2− {1} = {25643} = 𝑇2; 𝑆4 = 𝑇(4)

1 ∈ 𝐿0 = 𝑆6; {1} + 𝑇2 = {256431} = 𝑇3; 𝑆7 = 𝑇(1)

Ответ: 𝑇 = {562431}.

**Исходный код:**

**main.cpp**

#include <fstream>

#include <new>

#include <iostream>

#include <vector>

#include <map>

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 << "Matrix:" << endl;

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

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

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

}

cout << endl;

}

cout << "Amount of rows:" << ' ' << rows << endl;

cout << "Amount of cols:" << ' ' << cols << endl;

}

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 << "Families:" << 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 << " Iteration:" << endl;

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

}

delete\_matrix(matrix, rows);

return 0;

}

**matrix.txt:**

6

6

0 0 1 1 1 0

0 1 0 1 0 1

1 0 0 0 1 1

0 1 1 0 0 0

0 0 1 0 0 0

0 0 0 0 1 0

**Результат работы программы:**

Matrix:

0 0 1 1 1 0

0 1 0 1 0 1

1 0 0 0 1 1

0 1 1 0 0 0

0 0 1 0 0 0

0 0 0 0 1 0

Amount of rows: 6

Amount of cols: 6

Families:

S1: 3 4 5

S2: 2 4 6

S3: 1 5 6

S4: 2 3

S5: 3

S6: 5

T0:

3 2 1

1 Iteration:

L0:

2 3

L1:

2 3 4 6

L2:

2 3 4 6 5

T

4 2 1 3

2 Iteration:

L0:

3

L1:

3 2

L2:

3 2 4 6

L3:

3 2 4 6 5

T

4 6 1 2 3

3 Iteration:

L0:

5

T

4 6 1 2 3 5

**Используемая литература:**

## Т.М. Волосатова, С.В. Родионов. Методы комбинаторных вычислений, учебное пособие. Издательство МГТУ им. Н.Э. Баумана. Москва. 2011