|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования 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 «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № \_\_**1**\_\_**

**Дисциплина Методы вычислений**

|  |  |
| --- | --- |
| **Тема Венгерский метод решения задачи о назначениях**  **Вариант №19**  **Студент \_Патутин В.М.\_\_\_\_\_\_\_\_**  **Группа \_ИУ7-11М\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Власов П.А.** |  |

Москва, 2023 г.

**Теоретическая часть**

**Цель работы:** изучение венгерского метода решения задачи о назначениях.

**Содержательная и математическая постановка задачи**

В распоряжении работодателя имеется n работ и такое же число исполнителей. Стоимость выполнения i-ой работы j-ым исполнителем составляет единиц. Требуется распределить все работы между исполнителями так, чтобы каждый из них выполнял ровно 1 работу. А общая стоимость выполнения всех работ была минимальна.

Введём управляемые переменные:

, .

Здесь

- элементы матрицы стоимостей

- элементы матрицы назначений

Тогда стоимость выполнения всех работ:

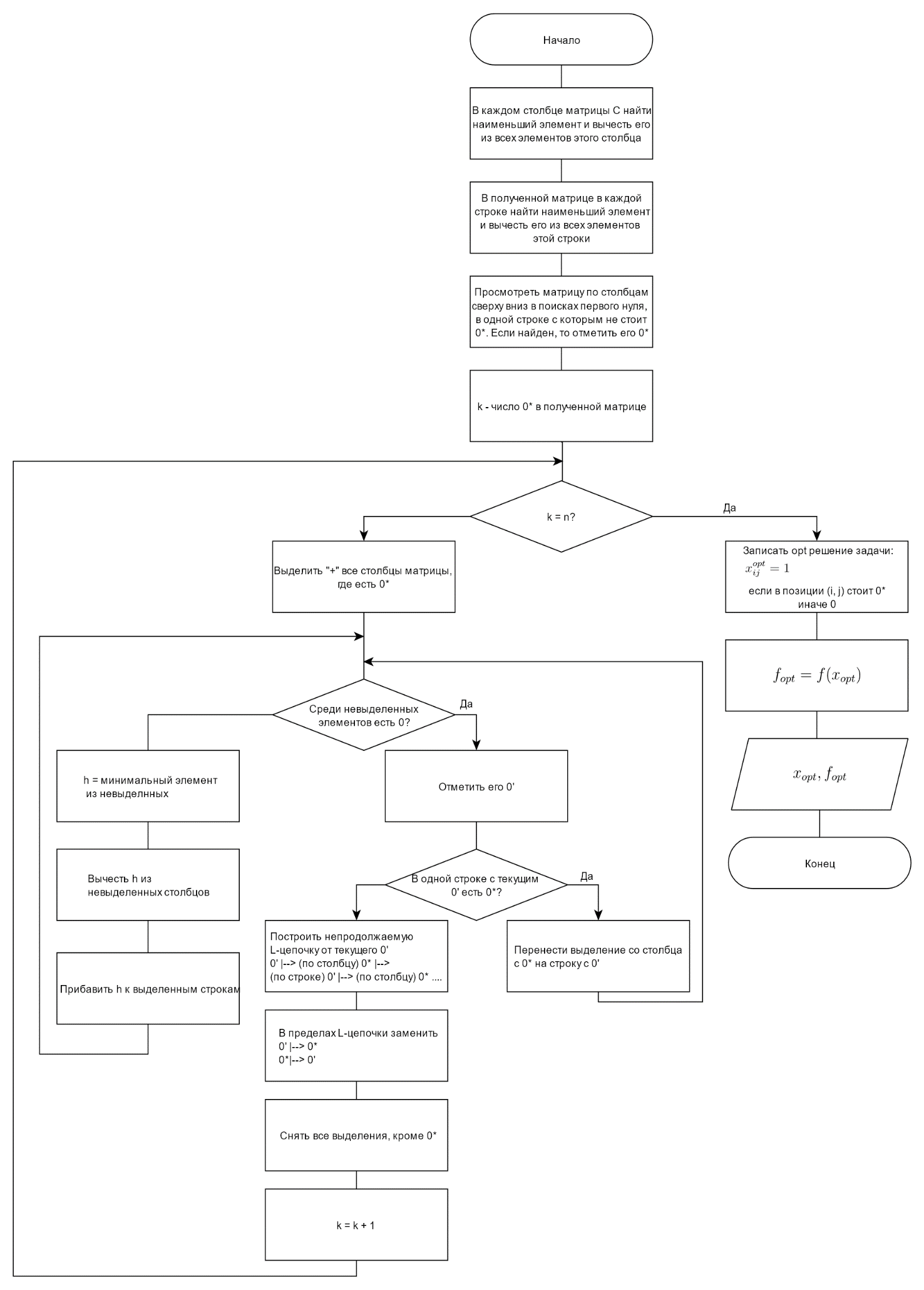
должны корректно задать распределение рабочих по задачам.  
Условие того, что j-й исполнитель выполняет ровно 1 работу:

Условие того, что i-ую работу выполняет ровно 1 исполнитель:

Таким образом, математическая постановка задачи о назначениях:

**Практическая часть**

Схема алгоритма:



C = [[9 5 10 4 8],

[5 8 9 3 7],

[8 5 2 5 10],

[7 3 4 5 9],

[10 6 8 7 9]]

Текст программы :

function lab1()

task(true);

end % lab1

function task(is\_maximization\_task)

C = [

9 5 10 4 8;

5 8 9 3 7;

8 5 2 5 10;

7 3 4 5 9;

10 6 8 7 9;

];

fprintf('Input matrix\n');

disp(C);

C\_backup = C;

size = length(C);

if is\_maximization\_task

disp('Maximization task');

max\_cost = max(max(C));

fprintf('Max cost is %i\n', max\_cost);

for i = 1:size

for j = 1:size

C(i, j) = max\_cost - C(i, j);

end

end

disp('Matrix after maximization -> minimization');

disp(C)

else

disp('Minimization task')

end

for i = 1:size

C(:, i) = C(:, i) - min(C(:, i));

end

disp('After column minimization')

disp(C);

for i = 1:size

C(i, :) = C(i, :) - min(C(i, :));

end

disp('After row minimization');

disp(C);

disp('Find stars zeros');

S = find\_stars(C);

count\_s = count\_stars(S);

mark = mark\_by\_cols(S);

print\_with\_stars(C, S, mark);

fprintf('Found %i stars\n\n', count\_s)

i\_number = 1;

found = true;

while (count\_s < size)

if (found)

fprintf('Iteration %i\n', i\_number)

i\_number = i\_number + 1;

end

fprintf('Too few zerous, %i < %i\n', count\_s, size);

[found, star\_i, star\_j] = find\_any\_pot\_star(C, S, mark);

while (found)

disp('Do one more step');

S(star\_i, star\_j) = 2;

print\_with\_stars(C, S, mark);

[switched, mark] = switch\_mark(mark, star\_i, star\_j);

print\_with\_stars(C, S, mark);

if (switched)

[found, star\_i, star\_j] = find\_any\_pot\_star(C, S, mark);

else

disp('Build L-chunk');

S = build\_l\_chunk(S, mark, star\_i, star\_j);

fprintf('\n')

S = drop\_po\_stars(S);

count\_s = count\_s + 1;

mark = mark\_by\_cols(S);

break;

end

end

if (~found)

disp('Improve matrix');

h = find\_h(C, mark);

for i = 1:size

if (mark.col(i) == 0)

C(:, i) = C(:, i) - h;

end

if (mark.row(i) ~= 0)

C(i, :) = C(i, :) + h;

end

end

end

print\_with\_stars(C, S, mark);

end

mark = mark\_by\_cols(S);

solution = zeros([size, size]);

value = 0;

for j = 1:size

solution(mark.col(j), j) = 1;

value = value + C\_backup(mark.col(j), j);

end

disp('Optimal solution');

disp(solution);

fprintf('Optimal value = %i', value);

end %task

function h = find\_h(C, mark)

max\_i = 0;

max\_j = 0;

size = length(C);

h = 0;

for i = 1:size

for j = 1:size

if (mark.col(j) == 0)

c = C(i, j);

if (c ~= 0 && (h == 0 || h > c))

h = c;

max\_i = i;

max\_j = j;

end

end

end

end

assert(h ~= 0);

fprintf('h min = h [%i, %i] = %i\n', max\_i, max\_j, h);

end % find\_h

function S = find\_stars(C)

size = length(C);

% 0 == ?

% 1 == \*

% 2 == '"'

S = zeros([size, size]);

for i = 1:size

for j = 1:size

if ((C(i, j) == 0) && (max(S(i, :)) == 0) && (max(S(:, j)) == 0))

S(i, j) = 1;

end

end

end

end % find\_stars

function S = build\_l\_chunk(S, mark, star\_i, star\_j)

from\_i = 0;

size = length(mark.row);

for i = 1:size

if (S(i, star\_j) == 1)

from\_i = i;

break;

end

end

fprintf('[%i, %i]', star\_i, star\_j);

assert(S(star\_i, star\_j) == 2);

S(star\_i, star\_j) = 1;

from\_j = star\_j;

if (from\_i == 0 || S(from\_i, from\_j) == 0)

return;

end

fprintf(' -> [%i, %i] -> ', from\_i, from\_j)

S(from\_i, from\_j) = 0;

S = build\_l\_chunk(S, mark, from\_i, mark.row(from\_i));

end % build\_l\_chunk

function S = drop\_po\_stars(S)

size = length(S);

for i = 1:size

for j = 1:size

if (S(i, j) == 2)

S(i, j) = 0;

end

end

end

end % drop\_po\_stars

function [found, i, j] = find\_any\_pot\_star(C, S, mark)

size = length(S);

found = false;

for i = 1:size

for j = 1:size

if (C(i, j) == 0 && mark.row(i) == 0 && mark.col(j) == 0)

found = true;

return;

end

end

end

end % find\_any\_pot\_star

function [switched, mark] = switch\_mark(mark, star\_i, star\_j)

mark.row(star\_i) = star\_j;

for j = 1:length(mark.col)

if (mark.col(j) == star\_i)

fprintf('Switch %i:%i with %i:%i\n', star\_i, star\_j, star\_i, j);

mark.col(j) = 0;

switched = true;

return;

end

end

fprintf('Can not do any switch\n');

j = 0;

switched = false;

end % switch\_mark

function mark = mark\_by\_cols(S)

size = length(S);

mark = struct( ...

'row', zeros([size, 1]), ...

'col', zeros([size, 1]) ...

);

for j = 1:size

for i = 1:size

if (S(i, j) == 1)

mark.col(j) = i;

break;

end

end

end

end % mark\_by\_cols

function count = count\_stars(S)

count = count\_values(S, 1);

end % count\_stars

function count = count\_values(M, value)

size = length(M);

count = 0;

for i = 1:size

for j = 1:size

if (M(i, j) == value)

count = count + 1;

end

end

end

end % count\_values

function print\_with\_stars(C, S, mark)

size = length(C);

for i = 1:size

fprintf(' ')

for j = 1:size

s = S(i, j);

if (s == 0)

fprintf('%i ', C(i, j));

elseif (s == 1)

fprintf('%i\*', C(i, j));

elseif (s == 2)

fprintf('%i"', C(i, j));

else

assert(false)

end

fprintf(' ')

end

if (mark.row(i) ~= 0)

fprintf('+')

end

fprintf('\n')

end

fprintf(' ')

for i = 1:size

if (mark.col(i) ~= 0)

fprintf('+ ')

else

fprintf(' ')

end

end

fprintf('\n')

end % print\_with\_stars

**Результаты расчетов для задач из индивидуального варианта.**

Задача минимизации

Input matrix  
 9 5 10 4 8  
 5 8 9 3 7  
 8 5 2 5 10  
 7 3 4 5 9  
 10 6 8 7 9  
  
Minimization task  
After column minimization  
 4 2 8 1 1  
 0 5 7 0 0  
 3 2 0 2 3  
 2 0 2 2 2  
 5 3 6 4 2  
  
After row minimization  
 3 1 7 0 0  
 0 5 7 0 0  
 3 2 0 2 3  
 2 0 2 2 2  
 3 1 4 2 0  
  
Find stars zeros  
 3 1 7 0\* 0   
 0\* 5 7 0 0   
 3 2 0\* 2 3   
 2 0\* 2 2 2   
 3 1 4 2 0\*   
 + + + + +   
Found 5 stars  
  
Optimal solution  
 0 0 0 1 0  
 1 0 0 0 0  
 0 0 1 0 0  
 0 1 0 0 0  
 0 0 0 0 1  
  
Optimal value = 23

Задача максимизации

Input matrix  
 9 5 10 4 8  
 5 8 9 3 7  
 8 5 2 5 10  
 7 3 4 5 9  
 10 6 8 7 9  
  
Maximization task  
Max cost is 10  
Matrix after maximization -> minimization  
 1 5 0 6 2  
 5 2 1 7 3  
 2 5 8 5 0  
 3 7 6 5 1  
 0 4 2 3 1  
  
After column minimization  
 1 3 0 3 2  
 5 0 1 4 3  
 2 3 8 2 0  
 3 5 6 2 1  
 0 2 2 0 1  
  
After row minimization  
 1 3 0 3 2  
 5 0 1 4 3  
 2 3 8 2 0  
 2 4 5 1 0  
 0 2 2 0 1  
  
Find stars zeros  
 1 3 0\* 3 2   
 5 0\* 1 4 3   
 2 3 8 2 0\*   
 2 4 5 1 0   
 0\* 2 2 0 1   
 + + + +   
Found 4 stars  
  
Iteration 1  
Too few zerous, 4 < 5  
Do one more step  
 1 3 0\* 3 2   
 5 0\* 1 4 3   
 2 3 8 2 0\*   
 2 4 5 1 0   
 0\* 2 2 0" 1   
 + + + +   
Switch 5:4 with 5:1  
 1 3 0\* 3 2   
 5 0\* 1 4 3   
 2 3 8 2 0\*   
 2 4 5 1 0   
 0\* 2 2 0" 1 +  
 + + +   
Improve matrix  
h min = h [1, 1] = 1  
 0 3 0\* 2 2   
 4 0\* 1 3 3   
 1 3 8 1 0\*   
 1 4 5 0 0   
 0\* 3 3 0" 2 +  
 + + +   
Too few zerous, 4 < 5  
Do one more step  
 0" 3 0\* 2 2   
 4 0\* 1 3 3   
 1 3 8 1 0\*   
 1 4 5 0 0   
 0\* 3 3 0" 2 +  
 + + +   
Switch 1:1 with 1:3  
 0" 3 0\* 2 2 +  
 4 0\* 1 3 3   
 1 3 8 1 0\*   
 1 4 5 0 0   
 0\* 3 3 0" 2 +  
 + +   
Do one more step  
 0" 3 0\* 2 2 +  
 4 0\* 1 3 3   
 1 3 8 1 0\*   
 1 4 5 0" 0   
 0\* 3 3 0" 2 +  
 + +   
Can not do any switch  
 0" 3 0\* 2 2 +  
 4 0\* 1 3 3   
 1 3 8 1 0\*   
 1 4 5 0" 0 +  
 0\* 3 3 0" 2 +  
 + +   
Build L-chunk  
[4, 4]  
 0 3 0\* 2 2   
 4 0\* 1 3 3   
 1 3 8 1 0\*   
 1 4 5 0\* 0   
 0\* 3 3 0 2   
 + + + + +   
Optimal solution  
 0 0 1 0 0  
 0 1 0 0 0  
 0 0 0 0 1  
 0 0 0 1 0  
 1 0 0 0 0  
  
Optimal value = 43

**Заключение.**

В результате выполнения лабораторной работы был реализован венгерский метод решения задачи о назначениях для задач минимизации и максимизации, В ходе выполнения лабораторной работы была написана программа на языке MATLAB, позволяющая применять венгерский метод решения задачи о назначениях.