|  |  |
| --- | --- |
| 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**

**Вариант 23**

|  |  |
| --- | --- |
| **Тема: Венгерский метод решения задачи о назначениях**  **Студент: Нгуен Н.Х.**  **Группа: ИУ7И-11М**  **Преподаватель: Власов П.А.** |  |

Москва.

2021 г.

# Теоретическая часть

## Содержательная постановка задачи о назначениях

В распоряжении работодателя имеется работ и исполнителей. Стоимость выполнения -й работы -м исполнителем составляет единиц. Требуется распределить все работы между исполнителями так, чтобы:

* каждый исполнитель исполнял ровно одну работу;
* суммарная стоимость выполнения всех работ была минимальной.

## Математическая постановка задачи о назначениях

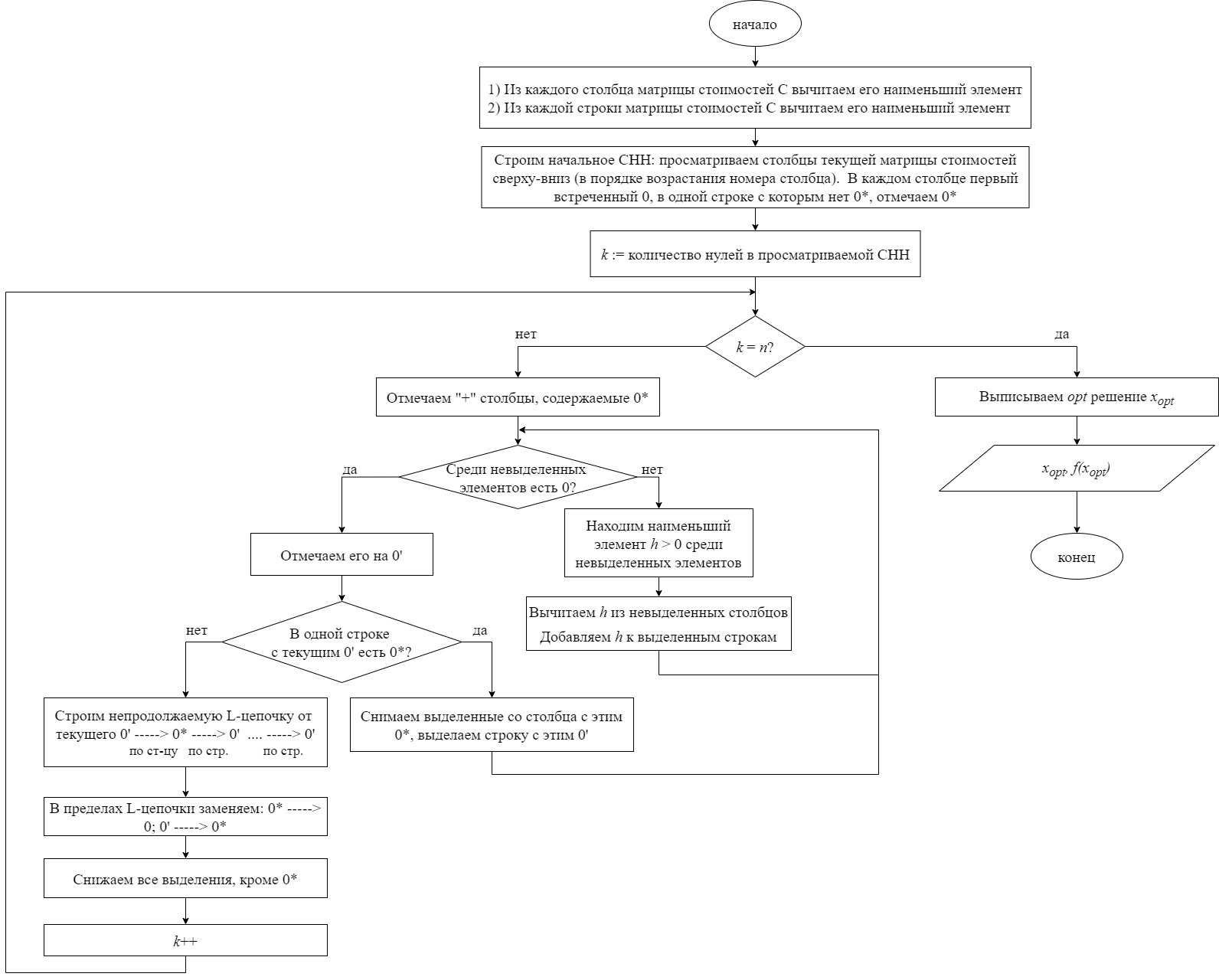
Введем управляемые переменные:

* – матрица назначения
* – матрица стоимостей

Тогда, суммарная стоимость выполнения всех работ:

Таким образом, введена математическая постановка задачи о назначениях:

## Венгерский метод решения задачи о назначениях



**Замечание**: Иногда встречается вариант задачи о назначениях в котором интерпретируются как прибыль от назначения -го работника на -ю работу. В этом случае задача о назначениях является задача максимизации:

Это задача о назначениях эквивалентна на задаче:

# Экспериментальная часть

## Исходные данные

Входные данные задаются в виде матрицы стоимостей C, для которой требуется решить задачу о назначениях в форме задач минимизации и максимизации:

## Текст программы

function lab1()

clc();

global DEBUG; %#ok<GVMIS>

global MINIMIZATION; %#ok<GVMIS>

MINIMIZATION = input('MINIMIZATION mode? y/n: [y]', 's');

if isempty(MINIMIZATION) || MINIMIZATION == 'y'

MINIMIZATION = true;

else

MINIMIZATION = false;

end

DEBUG = input('DEBUG mode? y/n: [y]', 's');

if isempty(DEBUG) || DEBUG == 'y'

DEBUG = true;

else

DEBUG = false;

end

if DEBUG

fprintf('---- DEBUG MODE ----\n');

else

fprintf('---- TOTAL MODE ----\n');

end

% 1. Initilize matrix C

CStruct = InitC();

% 2. Normalize matrix C

CStruct = NormalizeC(CStruct);

% 3. Build initial СНН (Система независимых нулей) and get k

[CStruct, k] = BuildInitSNN(CStruct);

[~, n] = size(CStruct.C);

iter = 1;

while true

if DEBUG

fprintf('Итерация %d: k = %d\n', iter, k);

end

if k >= n

break;

end

% 4. Mark columns that have 0\* with '+'

CStruct = MarkPlusCols(CStruct);

while true

% 5. Find free 0 among unmarked columns

[hasFreeZero, posZeroPrime] = FindFreeZero(CStruct);

if hasFreeZero

% 6.1. Mark free 0 to 0'

CStruct = MarkZeroPrime(CStruct, posZeroPrime);

% 7.1. Find 0\* in the row of the current 0'

[hasZeroStar, posZeroStar] = FindZeroStarInRow( ...

CStruct, posZeroPrime);

if hasZeroStar

% 8.1. Unmark column that has 0\*, Mark row that has 0'

CStruct = UpdateC(CStruct, posZeroPrime, posZeroStar);

else

% 9.1. Build L-chain from the last 0'

L = BuildL(CStruct, posZeroPrime);

% 10.1. Flip 0'<->0\*

CStruct = Flip(CStruct, L);

% 11.1. Clean C

CStruct = CleanC(CStruct);

k = k + 1;

break;

end

else

% 6.2. Find min positive value h among unmarked positions

h = FindMinH(CStruct);

% 7.2. Subtract h from unmarked cols, Add h to marked rows

CStruct = ApplyH(CStruct, h);

end

end

iter = iter + 1;

end

if DEBUG

fprintf('---- DEBUG DONE ----\n\n');

end

% Because we mark CStar with 0/1, it turns into optimal matrix

XOpt = CStruct.CStar;

fOpt = CalculateF(CStruct.COrigin, XOpt);

fprintf('X\_Opt: \n');

disp(XOpt);

fprintf('f(X\_Opt): \n');

disp(fOpt);

end

%% 1. Initialize matrix C

function CStruct = InitC()

%% 2. Normalize matrix C - Subtract every col and row by their min value

function CStruct = NormalizeC(CStruct)

%% 3. Build initial СНН and return k = |СНН|

function [CStruct, k] = BuildInitSNN(CStruct)

%% 4. Mark columns that have 0\* with '+'

function CStruct = MarkPlusCols(CStruct)

%% 5. Find free 0 among unmarked positions

function [hasFreeZero, posZeroPrime] = FindFreeZero(CStruct)

%% 6.1. Mark 0 to 0'

function CStruct = MarkZeroPrime(CStruct, posPrime)

%% 7.1. Find 0\* in the row of the current 0'

function [hasZeroStar, posZeroStar] = FindZeroStarInRow(CStruct, posPrime)

%% 8.1. Unmark column that has 0\*, Mark row that has 0'

function CStruct = UpdateC(CStruct, posPrime, posStar)

%% 9.1. Build L-chain from the last 0'

function L = BuildL(CStruct, posPrime)

%% 10.1. Flip 0'<->0\*

function CStruct = Flip(CStruct, L)

%% 11.1. Remove every mark except 0\*

function CStruct = CleanC(CStruct)

%% 6.2. Find min positive value h among unmarked positions

function h = FindMinH(CStruct)

%% 7.2. Subtract h from unmarked cols, Add h to marked rows

function CStruct = ApplyH(CStruct, h)

function f = CalculateF(C, XOpt)

%% ------------------------------ HELPER ------------------------------ %%

% Return a struct holding required parameters of C

% CStar: матрица звёзд

% СPrime: матрица штрих

function CStruct = ConstructC(COrigin, C, markedRows, markedCols, ...

CStar, CPrime, curPosPrime, curPosStar, ...

printMinCol, printMinRow)

% Print C to console

% CStruct: структура матрицы С

% msg: сообщение

function PrintC(CStruct, msg)

% Print L to console

function PrintL(L, msg)

% Compare index of 2 position A and B

function isEqual = IsEqual(posA, posB)

## Результаты

* Задача минимизации

---- TOTAL MODE ----  
Матрица стоимостей C:  
 1 4 7 9 4   
 9 3 8 7 4   
 3 4 6 8 2   
 8 2 4 6 7   
 7 6 9 8 5   
   
  
X\_Opt:   
 1 0 0 0 0  
 0 1 0 0 0  
 0 0 0 0 1  
 0 0 1 0 0  
 0 0 0 1 0  
  
f(X\_Opt):   
 18

* Задача максимации

---- TOTAL MODE ----  
Матрица стоимостей C:  
 1 4 7 9 4   
 9 3 8 7 4   
 3 4 6 8 2   
 8 2 4 6 7   
 7 6 9 8 5   
   
  
X\_Opt:   
 0 0 0 1 0  
 1 0 0 0 0  
 0 1 0 0 0  
 0 0 0 0 1  
 0 0 1 0 0  
  
f(X\_Opt):   
 38