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

*По предмету: «Методы вычислений»*

***Тема: «Венгерский метод решения задачи о назначениях»***

Преподаватель:

Власов П. А.,

Студент: Доманов К. И.,

Группа: ИУ7-11М

Москва, 2021 г.

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

**Содержательная постановка задачи о назначениях:**

В распоряжении работодателя имеется *n* работ и столько же исполнителей. Стоимость выполнения *i*-ой работы *j*-ым исполнителем составляет *сij ≥ 0* единиц. Требуется распределить все работы между исполнителями таким образом, чтобы:

* Каждый исполнитель исполнял ровно одну работу;
* Суммарная стоимость выполнения всех работ должна быть минимальной.

**Математическая постановка задачи о назначениях:**

**Исходные данные:**

Входные данные задаются в виде матрицы стоимостей C в соответствии с вариантом 5, для которой требуется решить задачу о назначениях в форме задач минимизации и максимизации.

**Описание венгерского метода**



**Текст программы**

C\_matr1 = cost\_matrx();

matrx\_prnt(C\_matr1, 'Матрица стоимостей');

max = false;

if max

C = max\_task(C\_matr1);

matrx\_prnt(C, 'Решение задачи максимизации')

else

C = C\_matr1;

end

C = subtract\_cols(C);

matrx\_prnt(C, 'Из каждого столбца вычитается минимальный элемент');

C = subtrack\_rows(C);

matrx\_prnt(C, 'Из каждой строки вычитается минимальный элемент');

C = starMatrix(C);

matrx\_prnt(C, 'Строим начальную СНН: первый в столбце 0, в одной строке с которым нет 0\*, отмечаем с помощью \*');

iter = 0;

while C.numberOfStars < C.sizeMatrix

fprintf('Итерация: %d\n', iter);

iter = iter + 1;

C = highlighting\_columns(C);

matrx\_prnt(C, 'Отмечаем столбцы с 0\*');

[C,rowIndex, columnIndex] = markWithDash(C);

C = buildLchain(C, rowIndex, columnIndex);

matrx\_prnt(C, "В L-цепочке меняем все 0\* на 0 и 0' на 0\*");

C = removeIcons(C);

matrx\_prnt(C, "Снимаем все выделения, кроме \*");

end

C = optimizedMatrix(C);

fprintf('Количество независимых нулей = n \n');

matrx\_prnt(C, 'Записываем оптимальное решение x\*: xij = 1, если в позиции (i, j) матрицы стоимостей стоит 0\*, иначе xij = 0');

matrix = C\_matr1.matrix;

sizeMatrix = C\_matr1.sizeMatrix;

fopt = 0;

for r = 1:sizeMatrix

for c = 1:sizeMatrix

fopt = fopt + matrix(r,c) \* C.matrix(r,c);

end

end

fprintf('f(x\*) = %d\n', fopt(1));

function C = cost\_matrx()

matrix = dlmread('5var.txt');

[rows, cols] = size(matrix);

starMatrix = zeros(rows,cols);

dashMatrix = zeros(rows,cols);

Lchain = zeros(rows,cols);

C = struct('matrix', matrix, 'sizeMatrix', rows, 'starMatrix', starMatrix , 'dashMatrix', dashMatrix, 'Lchain', Lchain, 'markedRows', starMatrix(:,1), 'markedColumns', starMatrix(1,:), 'numberOfStars', 0);

end

function result = max\_task(C)

matrix = C.matrix;

matrix = -matrix + max(max(matrix));

result = C;

result.matrix = matrix;

end

function matrx\_prnt(C, msg)

matrix = C.matrix;

sizeMatrix = C.sizeMatrix;

fprintf('%s:\n', msg);

for r = 1:sizeMatrix

for c = 1:sizeMatrix

if C.markedRows(r) || C.markedColumns(c)

fprintf('<strong>');

end

fprintf('%2g', matrix(r,c));

if C.starMatrix(r,c)

fprintf('\*');

elseif C.dashMatrix(r,c)

fprintf("'");

else

fprintf(' ');

end

if C.markedRows(r) || C.markedColumns(c)

fprintf(2, '</strong>');

end

end

fprintf('\n');

end

fprintf('\n');

end

function result = subtract\_cols(C)

matrix = C.matrix;

sizeMatrix = C.sizeMatrix;

for i = 1:sizeMatrix

col = matrix(:,i);

matrix(:,i) = col - min(col);

end

result = C;

result.matrix = matrix;

end

function result = subtrack\_rows(C)

matrix = C.matrix;

sizeMatrix = C.sizeMatrix;

for i = 1:sizeMatrix

row = matrix(i,:);

matrix(i,:) = row - min(row);

end

result = C;

result.matrix = matrix;

end

function result = starMatrix(C)

matrix = C.matrix;

sizeMatrix = C.sizeMatrix;

starMatrix = C.starMatrix;

stars\_cols = 0;

for c = 1:sizeMatrix

for r = 1:sizeMatrix

if matrix(r,c) == 0

flag = false;

for c2 = 1:sizeMatrix

if starMatrix(r,c2)

flag = true;

end

end

if ~flag

starMatrix(r,c)=true;

stars\_cols=stars\_cols+1;

break

end

end

end

end

result = C;

result.starMatrix = starMatrix;

result.numberOfStars = stars\_cols;

end

function result = highlighting\_columns(C)

sizeMatrix = C.sizeMatrix;

starMatrix = C.starMatrix;

markedColumns = C.markedColumns;

for c = 1:sizeMatrix

for r = 1:sizeMatrix

if starMatrix(r,c)

markedColumns(c) = true;

break;

end

end

end

result = C;

result.markedColumns = markedColumns;

end

function [result, rowIndex,columnIndex] = markWithDash(C)

matrix = C.matrix;

sizeMatrix = C.sizeMatrix;

starMatrix = C.starMatrix;

dashMatrix = C.dashMatrix;

markedColumns = C.markedColumns;

markedRows = C.markedRows;

rowIndex = 0;

columnIndex = 0;

cont\_flag = true;

while cont\_flag

flag = false;

for c = 1:sizeMatrix

for r = 1:sizeMatrix

if (~markedRows(r) && ~markedColumns(c) && matrix(r,c) == 0)

dashMatrix(r,c) = true;

rowIndex=r;

columnIndex=c;

flag = true;

break

end

end

if flag

break;

end

end

if flag

C4 = C;

C4.markedRows = markedRows;

C4.markedColumns = markedColumns;

C4.dashMatrix = dashMatrix;

matrx\_prnt(C4, "Среди невыделенных элементов есть 0, отмечаем этот 0 с помощью '");

flag2 = false;

for c = 1:sizeMatrix

if starMatrix(rowIndex, c)

markedColumns(c) = false;

markedRows(rowIndex) = true;

flag2= true;

break;

end

end

if ~flag2

cont\_flag = false;

else

C2 = C;

C2.markedRows = markedRows;

C2.markedColumns = markedColumns;

C2.dashMatrix = dashMatrix;

matrx\_prnt(C2, "Снимаем выделение со столбца с 0\* и выделяем строку с 0'");

end

else

C2 = C;

C2.markedRows = markedRows;

C2.markedColumns = markedColumns;

C2.dashMatrix = dashMatrix;

[C3, minNumber] = subtract\_min(C2);

fprintf('Ищем наименьший элемент среди невыделенных элементов в матрице = %d\n', minNumber);

matrx\_prnt(C3, 'Вычитаем его из невыделеных столбцов и добавляем к выделенным строкам');

C = C3;

matrix = C3.matrix;

end

end

result = C;

result.markedColumns = markedColumns;

result.markedRows = markedRows;

result.starMatrix = starMatrix;

result.dashMatrix = dashMatrix;

end

function [result, minNumber] = subtract\_min(C)

matrix = C.matrix;

sizeMatrix = C.sizeMatrix;

markedRows = C.markedRows;

markedColumns = C.markedColumns;

minNumber = intmax;

for r = 1:sizeMatrix

for c = 1:sizeMatrix

if (~markedRows(r) && ~markedColumns(c))

minNumber = min(minNumber, matrix(r,c));

end

end

end

for r = 1:sizeMatrix

for c = 1:sizeMatrix

if (markedRows(r) && markedColumns(c))

matrix(r,c) = matrix(r,c) + minNumber;

end

if (~markedRows(r) && ~markedColumns(c))

matrix(r,c) = matrix(r,c) - minNumber;

end

end

end

result = C;

result.matrix = matrix;

result.markedRows = markedRows;

result.markedColumns = markedColumns;

end

function result = buildLchain(C, rowIndex, columnIndex)

sizeMatrix = C.sizeMatrix;

starMatrix = C.starMatrix;

dashMatrix = C.dashMatrix;

markedColumns = C.markedColumns;

markedRows = C.markedRows;

Lchain = C.Lchain;

Lchain(rowIndex,columnIndex) = true;

while true

flag = false;

for r = 1:sizeMatrix

if (starMatrix(r, columnIndex))

flag = true;

rowIndex = r;

Lchain(rowIndex, columnIndex) = true;

end

end

if flag

for c = 1:sizeMatrix

if (dashMatrix(rowIndex, c))

columnIndex = c;

Lchain(rowIndex, columnIndex) = true;

end

end

else

break;

end

end

fprintf('Строим непродолжительную L-цепочку, начиная от текущего 0'': идем по столбцу до 0\*, по строке до 0''\n');

for r = 1:sizeMatrix

for c = 1:sizeMatrix

if (Lchain(r,c))

if (starMatrix(r,c))

starMatrix(r,c) = false;

elseif (dashMatrix(r,c))

dashMatrix(r,c) = false;

starMatrix(r,c) = true;

end

end

end

end

result = C;

result.starMatrix = starMatrix;

result.dashMatrix = dashMatrix;

result.markedColumns = markedColumns;

result.markedRows = markedRows;

end

function result = removeIcons(C)

sizeMatrix = C.sizeMatrix;

starMatrix = C.starMatrix;

dashMatrix = C.dashMatrix;

markedRows = C.markedRows;

markedColumns = C.markedColumns;

numberOfStars = 0;

for r = 1:sizeMatrix

if markedRows(r)

markedRows(r) = false;

end

for c = 1:sizeMatrix

if markedColumns(c)

markedColumns(c) = false;

end

if dashMatrix(r,c)

dashMatrix(r,c) = false;end

if starMatrix(r,c)

numberOfStars = numberOfStars +1;

end

end

end

result = C;

result.starMatrix = starMatrix;

result.dashMatrix = dashMatrix;

result.markedRows = markedRows;

result.markedColumns = markedColumns;

result.numberOfStars = numberOfStars;

end

function result = optimizedMatrix(C)

matrix = C.matrix;

sizeMatrix = C.sizeMatrix;

starMatrix = C.starMatrix;

for r = 1:sizeMatrix

for c = 1:sizeMatrix

if starMatrix(r,c)

matrix(r,c) = 1;

starMatrix(r,c) = false;

else

matrix(r,c) = 0;

end

end

end

result = C;

result.matrix = matrix;

result.starMatrix = starMatrix;

end

**Результаты расчетов для задач из индивидуального варианта**

1. Задача минимизации**:**

Матрица стоимостей =

9 11 3 6 6

10 9 11 5 6

8 10 5 6 4

6 8 10 4 9

11 10 9 8 7

x(opt) =

0 0 1 0 0

0 0 0 1 0

0 0 0 0 1

1 0 0 0 0

0 1 0 0 0

f(opt) = 28

1. Задача максимизации**:**

Матрица стоимостей =

9 11 3 6 6

10 9 11 5 6

8 10 5 6 4

6 8 10 4 9

11 10 9 8 7

x(opt) =

0 1 0 0 0

0 0 1 0 0

0 0 0 1 0

0 0 0 0 1

1 0 0 0 0

f(opt) = 48

**Вывод:** в результате выполнения работы был изучен венгерский метод решения задачи о назначениях. Результатом работы является файл с программой, реализующий поставленные задачи.