**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 курсу «Методы вычислений»**
**Вариант 18**

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

**Студент** Сушина А.Д.

**Группа** ИУ7-12м

**Оценка (баллы) _____**

**Преподаватель** Власов П.А.

Москва.
2021 г

Цель работы: изучение венгерского метода решения задачи о назначениях.

# 1 Постановка задачи

**Содержательная постановка задачи**:

*В распоряжении имеется n работ и n исполнителей. Стоимость выполнения i-ой работы j-ым исполнителем составляет* $c_{ij} >= 0$. *Требуется распределить работы между исполнителями так, чтобы:*

1. *каждый из них выполнял ровно одну работу;*
2. *общая стоимость выполнения всех работ была минимальной.*

Обозначим $C=(c_{ij})_{i,j=\overline{1,n}}$ , где C назовем матрицей стоимостей.

Введем управляемые переменные

$$x_{i,j}=\begin{cases}1, \textit{если i работу выполняет j исполнитель}\\ 0, \textit{иначе}\end{cases}$$

Обозначим $X=(x_{ij})_{i,j=\overline{1,n}}$ , где X назовем матрицей назначений.

Тогда

$$f=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij} \ ,$$

где f – стоимость выполнения работ, отвечающая матрице X.

Таким образом, получаем систему

$$\begin{cases} f=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{iJ}\to min \\ x_{ij}\in\{0,1\}\, i,j=\overline{1,n}\end{cases}$$

Также следует участь, что один работник может выполнять только одну работу. Следовательно каждый столбец и каждая строка матрицы X должны содержать ровно одну единицу. Добавляем эти условия и приходим к **математической постановке задачи**:

$$\begin{cases} f=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{iJ}\to min \\ x_{ij}\in\{0,1\}\, i,j=\overline{1,n} \\ \sum_{i=1}^{n}x_{ij}=1, j=\overline{1,n} \\ \sum_{j=1}^{n}x_{ij}=1, i=\overline{1,n}\end{cases}$$

**Индивидуальный вариант матрицы стоимостей**

$$\begin{bmatrix} 6 & 7 & 3 & 9 & 9 \\ 8 & 9 & 7 & 6 & 9 \\ 6 & 11 & 5 & 4 & 6 \\ 10 & 10 & 10 & 3 & 5 \\ 8 & 8 & 5 & 10 & 9 \end{bmatrix}$$

# 2 Венгерский метод

Приведем кратное описание Венгерского метода решения задачи о назначениях.

Сначала необходимо выполнить следующие действия:

1. В каждом столбце матрицы стоимостей выбрать наименьший элемент и вычесть его из всех элементов этого столбца.
2. В каждой строке полученной матрицы выбрать наименьший элемент и вычесть его из каждого элемента этой строки.

В результате получим матрицу, каждая строка и каждый столбец которой содержат по крайней мере 1 нуль.

Теперь можно построить начальную систему независимых нулей. Для этого нужно просмотреть столбцы текущей матрицы стоимостей сверху вниз, начиная с первого столбца; первый найденный в столбце 0, в одной строке с которым нет «0*», отметить «*». Если полученная система независимых нулей содержит n элементов, значит найдено оптимальное решение. В противном случае систему необходимо улучшить.

Первый способ улучшения системы независимых нулей: «Исключить некоторые нули из системы независимых нулей так, чтобы вместо них можно было взять большее число нулей». Для этого нужно выполнить следующие шаги:

1. Отметить «+» столбцы, в которых стоят «0*». (Эти столбцы и все их элементы будем называть выделенными)
2. Найти 0 среди невыделенных элементов и отметить его «0'». (Он является кандидатом на включение в систему независимых нулей)
3. Если в одной строке с «0'» стоит «0*», то перенести выделение со столбца с этим «0*» на строку с текущим «0'». Затем повторить поиск.
4. Если в одно строке с «0'» не стоит «0*», то построить L-цепочку следующего вида:
   $тек\ 0' \rightarrow по\ столбцу \rightarrow 0* \rightarrow по\ строке \rightarrow 0' \rightarrow ... \rightarrow 0'$
   В пределах цепочки заменить «0*» на 0, а «0'» на «0*».

Первый способ можно использовать, если среди невыделенных элементов есть нули. Однако, если таких нулей нет, необходимо использовать второй способ.

Второй способ улучшения системы независимых нулей состоит из следующих шагов:

1. Выбрать наименьший элемент h среди всем невыделенных элементов (h>0).
2. Вычесть h из невыделенных столбцов.
3. Для «исправления» отрицательных элементов добавить h к выделенным строкам.

В результате преобразований получим эквивалентную матрицу, среди невыделенных элементов которой есть нуль. После этого можно продолжить использовать первый способ.

Иногда встречаются задачи, в которых $c_{ij}$ интерпретируются как прибыль, полученная при назначении j-го работника на i-ю работу. В этом случае задача о назначениях является задачей максимизации. В этом случае необходимо провести эквивалентные преобразования: в матрице C выбрать максимальный элемент и добавить его к каждому элементу матрицы -C. Таким образом получим задачу о назначениях (минимизации) и соответственно ее можно решать тем же способом.

(Тут должна быть схема алгоритма)

# 3 Текст программы

На листинге 1 приведен текст программы.

Листинг 1.

```
function lab1met()

debug=1;
findMax = 1;
startMatrix = [6 7 3 9 9;
   8 9 7 6 9;
   6 11 5 4 6;
   10 10 10 3 5;
   8 8 5 10 9];

fprintf('Исходная матрица\n');
printMatrix(startMatrix);

M = startMatrix;

% Если нужно найти максимум, проводим дополнительные преобразования
if findMax == 1
   maxM = max(max(M));
   for i = 1 : 1: length(M(:, 1))
      M(:, i) = M(:, i)  * (-1) + maxM;
   end

   if debug == 1
      fprintf('Матрица, после преобразований для поиска максимума\n');
      printMatrix(M);
   end
end


C = M;
MinCols = min(C);

for i = 1 : 1 : length(MinCols)
   C(:, i) = C(:, i) - MinCols(i);
end
```

```
if debug == 1
   fprintf('Вычитаем наименьшие элементы по строкам\n');
   printMatrix(C);
end

MinRows = min(C, [], 2);
for i = 1 : 1 : length(MinRows)
   C(i, :) = C(i, :) - MinRows(i);
end

if debug == 1
   fprintf('Вычитаем наименьшие элементы по столбцам\n');
   printMatrix(C);
end

D = C;

cols = length(MinCols);
rows = length(MinRows);
stars = zeros(rows, cols);

for i = 1: 1: cols
   for j = 1 : 1 : rows
      if D(j, i) == 0
         count = 0;
         for k = 1: 1: cols
            count = count + stars(j, k);
         end
         for k = 1: 1: rows
            count = count + stars(k, i);
         end
         if count == 0
            stars(j, i) = 1;
         end
      end
   end
end

if debug == 1
   fprintf('Начальная система независимых нулей\n');
   printSNN(D, stars)
end

k = sum(stars, 'all');
if debug == 1
   fprintf('k = %d\n', k);
end

while k < cols
   if debug == 1
      fprintf('---итерация алгоритма---\n');
   end

   selection = zeros(rows, cols);
   strihMatrix = zeros(rows, cols);
   selectedColumns = sum(stars);
```

```
for i = 1: 1: cols
   if selectedColumns(i) == 1
      selection(:, i) = selection(:, i) + 1;
   end
end


selectedRows = zeros(rows);

loop = 1;
strih = [-1, -1];
while loop == 1

   strih = findStrih(D, selection);

   if strih(1) == -1
      h = -1;
      for i = 1: 1: cols
         for j = 1 : 1 : rows
            if selection(j, i) == 0
               if D(j, i) < h || h == -1
                  h = D(j,i);
               end
            end
         end
      end


      for i = 1: 1: cols
         if selectedColumns(i) == 0
            D(:, i) = D(:, i) - h;
         end
      end
      for i = 1: 1: rows
         if selectedRows(i) == 1
            D(i, :) = D(i, :) + h;
         end
      end

      strih = findStrih(D, selection);
   end

   strihMatrix(strih(1), strih(2)) = 1;

   j = strih(1);
   star = [-1 -1];
   for i = 1: 1: cols
     if stars(j, i ) == 1
        star(1) = j;
        star(2) = i;
     end
   end

   if star(1) == -1
      loop = 0;
   else
      selection(:, star(2)) = selection(:, star(2)) - 1;
```

```
            selectedColumns(star(2)) = 0;
            selection(star(1), :) = selection(star(1), :) + 1;
            selectedRows(star(1)) = 1;
        end
    end

    i = strih(1);
    j = strih(2);
    strihFlag = 1;
    while i > 0 && j > 0 && i <= rows && j <= cols
        strihMatrix(i, j) = 0;
        stars(i, j) = 1;
        k = 1;
        while k <=rows  && (stars(k, j) ~= 1 || k == i)
            k = k+1;
        end
        if (k <= rows)
            l = 1;
            while l <= cols && (strihMatrix(k, l) ~= 1 || l == j)
                l = l+1;
            end

            if l <= cols
                stars(k,j) = 0;
            end
            j = l;
        end
        i = k;
    end


    k = sum(stars, 'all');
    if debug == 1
        fprintf('Текущая система независимых нулей\n');
        printSNN(D, stars);
        fprintf('k = %d\n', k);
    end

end

    fprintf('Конечная система независимых нулей\n');
    printSNN(D, stars);


    fprintf('X = \n');
    printMatrix(stars);

    result = 0;
    for i = 1:1:cols
        for j = 1:1:rows
            if stars(j, i) == 1
                result = result + startMatrix(j, i);
            end
        end
    end

    fprintf("Результат = %d\n", result)
```

```
end

function [strih] = findStrih(D, selection)
   strih = [-1 -1];
   cols = length(D(1, :));
   rows = length(D(:, 1));
   for i = 1: 1: cols
      for j = 1 : 1 : rows
        if selection(j, i) == 0 && D(j, i) == 0
            strih(1) = j;
            strih(2) = i;
            return;
        end
      end
   end
end

function [] = printSNN(M, stars)
   cols = length(M(1, :));
   rows = length(M(:, 1));

   for i = 1: 1: rows
      for j =1: 1: cols
         if stars(i, j) == 1
            fprintf("*\t");
         else
            fprintf(".\t");
         end
      end
      fprintf('\n');
      for j =1: 1: cols
         fprintf("%d\t", M(i, j))
      end
      fprintf('\n');
   end

end


function [] = printMatrix(M)
   fprintf([repmat('%d\t', 1, size(M, 2)) '\n'], M')
end
```

# 4 Результаты расчетов

## Задача максимизации

Исходная матрица

| | | | | |
|---|---|---|---|---|
| 6 | 7 | 3 | 9 | 9 |
| 8 | 9 | 7 | 6 | 9 |
| 6 | 11 | 5 | 4 | 6 |
| 10 | 10 | 10 | 3 | 5 |
| 8 | 8 | 5 | 10 | 9 |

Матрица, после преобразований для поиска максимума

| | | | | |
|---|---|---|---|---|
| 5 | 4 | 8 | 2 | 2 |
| 3 | 2 | 4 | 5 | 2 |
| 5 | 0 | 6 | 7 | 5 |
| 1 | 1 | 1 | 8 | 6 |
| 3 | 3 | 6 | 1 | 2 |

Вычитаем наименьшие элементы по столбцам

| | | | | |
|---|---|---|---|---|
| 4 | 4 | 7 | 1 | 0 |
| 2 | 2 | 3 | 4 | 0 |
| 4 | 0 | 5 | 6 | 3 |
| 0 | 1 | 0 | 7 | 4 |
| 2 | 3 | 5 | 0 | 0 |

Вычитаем наименьшие элементы по строкам

| | | | | |
|---|---|---|---|---|
| 4 | 4 | 7 | 1 | 0 |
| 2 | 2 | 3 | 4 | 0 |
| 4 | 0 | 5 | 6 | 3 |
| 0 | 1 | 0 | 7 | 4 |
| 2 | 3 | 5 | 0 | 0 |

Начальная система независимых нулей

| | | | | |
|---|---|---|---|---|
| . | . | . | . | * |
| 4 | 4 | 7 | 1 | 0 |
| . | . | . | . | . |
| 2 | 2 | 3 | 4 | 0 |
| . | * | . | . | . |
| 4 | 0 | 5 | 6 | 3 |
| * | . | . | . | . |
| 0 | 1 | 0 | 7 | 4 |
| . | . | . | * | . |
| 2 | 3 | 5 | 0 | 0 |

k = 4

---итерация алгоритма---

Текущая система независимых нулей

| | | | | |
|---|---|---|---|---|
| . | . | . | . | * |
| 2 | 4 | 5 | 1 | 0 |
| * | . | . | . | . |
| 0 | 2 | 1 | 4 | 0 |
| . | * | . | . | . |
| 2 | 0 | 3 | 6 | 3 |
| . | . | * | . | . |
| 0 | 3 | 0 | 9 | 6 |
| . | . | . | * | . |
| 0 | 3 | 3 | 0 | 0 |

k = 5

Конечная система независимых нулей

| | | | | |
|---|---|---|---|---|
| . | . | . | . | * |
| 2 | 4 | 5 | 1 | 0 |
| * | . | . | . | . |

| | | | | |
|---|---|---|---|---|
| 0 | 2 | 1 | 4 | 0 |
| . | * | . | . | . |
| 2 | 0 | 3 | 6 | 3 |
| . | . | * | . | . |
| 0 | 3 | 0 | 9 | 6 |
| . | . | . | * | . |
| 0 | 3 | 3 | 0 | 0 |

X =

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |

Результат = 48

## Задача минимизации

Исходная матрица

| | | | | |
|---|---|---|---|---|
| 6 | 7 | 3 | 9 | 9 |
| 8 | 9 | 7 | 6 | 9 |
| 6 | 11 | 5 | 4 | 6 |
| 10 | 10 | 10 | 3 | 5 |
| 8 | 8 | 5 | 10 | 9 |

Вычитаем наименьшие элементы по столбцам

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 6 | 4 |
| 2 | 2 | 4 | 3 | 4 |
| 0 | 4 | 2 | 1 | 1 |
| 4 | 3 | 7 | 0 | 0 |
| 2 | 1 | 2 | 7 | 4 |

Вычитаем наименьшие элементы по строкам

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 6 | 4 |
| 0 | 0 | 2 | 1 | 2 |
| 0 | 4 | 2 | 1 | 1 |
| 4 | 3 | 7 | 0 | 0 |
| 1 | 0 | 1 | 6 | 3 |

Начальная система независимых нулей

| | | | | |
|---|---|---|---|---|
| * | . | . | . | . |
| 0 | 0 | 0 | 6 | 4 |
| . | * | . | . | . |
| 0 | 0 | 2 | 1 | 2 |
| . | . | . | . | . |
| 0 | 4 | 2 | 1 | 1 |
| . | . | . | * | . |
| 4 | 3 | 7 | 0 | 0 |
| . | . | . | . | . |
| 1 | 0 | 1 | 6 | 3 |

k = 3

---итерация алгоритма---
Текущая система независимых нулей

| | | | | |
|---|---|---|---|---|
| . | . | * | . | . |
| 0 | 0 | 0 | 6 | 4 |
| . | * | . | . | . |
| 0 | 0 | 2 | 1 | 2 |
| * | . | . | . | . |
| 0 | 4 | 2 | 1 | 1 |
| . | . | . | * | . |
| 4 | 3 | 7 | 0 | 0 |
| . | . | . | . | . |
| 1 | 0 | 1 | 6 | 3 |

k = 4

---итерация алгоритма---
Текущая система независимых нулей

| | | | | |
|---|---|---|---|---|
| . | . | * | . | . |
| 0 | 0 | 0 | 5 | 3 |
| . | . | . | * | . |
| 0 | 0 | 2 | 0 | 1 |
| * | . | . | . | . |
| 0 | 4 | 2 | 0 | 0 |
| . | . | . | . | * |
| 5 | 4 | 8 | 0 | 0 |
| . | * | . | . | . |
| 1 | 0 | 1 | 5 | 2 |

k = 5

Конечная система независимых нулей

| | | | | |
|---|---|---|---|---|
| . | . | * | . | . |
| 0 | 0 | 0 | 5 | 3 |
| . | . | . | * | . |
| 0 | 0 | 2 | 0 | 1 |
| * | . | . | . | . |
| 0 | 4 | 2 | 0 | 0 |
| . | . | . | . | * |
| 5 | 4 | 8 | 0 | 0 |
| . | * | . | . | . |
| 1 | 0 | 1 | 5 | 2 |

X =

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 |

Результат = 28