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»

ОТЧЁТ

По лабораторной работе № 1

По дисциплине: «Методы вычислений»

На тему: *«Венгерский метод решения задачи о назначениях»*

Вариант 6

Студент ИУ7-13М _____ Керимов А. Ш.

(Группа) (Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Власов П. А.

(Подпись, дата) (Фамилия И. О.)

Москва 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: изучение венгерского метода решения задачи о назначениях.

Содержание работы

а) реализовать венгерский метод решения задачи о назначениях в виде программы на ЭВМ[1)];

б) провести решение задачи с матрицей стоимостей, заданной в индивидуальном варианте, рассмотрев два случая:

1) задача о назначениях является задачей минимизации,
2) задача о назначениях является задачей максимизации.

[1)] В программе необходимо предусмотреть два режима работы: «итоговый», когда программа печатает только матрицу назначений, и «отладочный», когда на каждой итерации на экран выводится текущая матрица эквивалентной задачи с отмеченной (например, цветом или шрифтом) системой независимых нулей.

1 Аналитический раздел

1.1 Постановка задачи

1.1.1 Содержательная постановка

В распоряжении работодателя имеется n работ и n исполнителей. Стоимость выполнения i-й работы j-м исполнителем составляет $c_{ij} \geqslant 0$ единиц.

— Требуется распределить все работы по исполнителям так, чтобы каждый исполнитель выполнял ровно 1 работу.
— Общая стоимость всех работ должна быть минимальной.

1.1.2 Математическая постановка

Обозначим за матрицу стоимостей

$$C = (c_{ij}), \quad i, j = \overline{1, n}. \tag{1.1}$$

Введём так называемые управляемые переменные:

$$x_{ij} = \begin{cases} 1, \text{ если } i\text{-ю работу выполняет } j\text{-й работник;} \\ 0, \text{ иначе.} \end{cases} \quad i, j = \overline{1, n} \tag{1.2}$$

Обозначим за матрицу назначений

$$X = (x_{ij}), \quad i, j = \overline{1, n}. \tag{1.3}$$

Математическая постановка задачи о назначениях:

$$\begin{cases} f(x) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} c_{ij}x_{ij} & \to \min, \\ \sum\limits_{i=1}^{n} x_{ij} = 1, & j = \overline{1,n}, \\ \sum\limits_{j=1}^{n} x_{ij} = 1, & i = \overline{1,n}, \\ x_{ij} \in \{0,1\}, & i,j = \overline{1,n}. \end{cases} \tag{1.4}$$

1.2 Исходные данные варианта 6

$$C = \begin{bmatrix} 10 & 8 & 6 & 4 & 9 \\ 11 & 9 & 10 & 5 & 6 \\ 5 & 10 & 8 & 6 & 4 \\ 3 & 11 & 9 & 6 & 6 \\ 8 & 10 & 11 & 8 & 7 \end{bmatrix} \tag{1.5}$$

2 Конструкторский раздел

2.1 Краткое описание венгерского метода

Алгоритм 1 Венгерский метод решения задачи о назначениях

1: **Начало**
2: Из каждого столбца матрицы назначения вычитаем его min элемент
3: Из каждой строки матрицы назначения вычитаем её min элемент
4: Строим начальную СНН: просм. ст-цы тек. м-цы ст-тей (в порядке возр-я номера ст-ца) сверху вниз. Первый в ст-це нуль, в одной стр. с кот. нет 0*, отмечаем 0*
5: $k \coloneqq |\text{СНН}|$
6: **Если** $k = n$ **тогда**
7: Записываем оптимальное решение:

$$x^*_{ij} \coloneqq \begin{cases} 1, \text{ если в позиции (i, j) м-цы ст-тей стоит 0*,} \\ 0, \text{ иначе} \end{cases} \tag{2.1}$$

8: $f^* \coloneqq f(X^*)$
9: Вывод X^*, f^*
10: **Конец**
11: **Иначе**
12: Столбцы с 0* отмечаем "+"
13: **Если** Среди невыделенных элементов есть 0 **тогда**
14: Отмечаем его 0'
15: **Если** В одной строке с текущим 0' есть 0* **тогда**
16: Снимаем выделение со столбца с этим 0*, выделяем "+"стр. с тек. 0'
17: **Перейти к** 13 шагу алгоритма
18: **Иначе**
19: Строим непродолж. L-цеп.: от тек. 0' по ст-цу в 0* по стр. . . . по стр. в 0'
20: В пределах L-цепочки: $0^* \longmapsto 0; \quad 0' \longmapsto 0^*$
21: Снимаем все выделения, $k \coloneqq |\text{СНН}|$
22: **Перейти к** 6 шагу алгоритма
23: **Конец условия**
24: **Иначе**
25: Ищем h — min элемент среди невыделенных
26: Вычитаем h из невыд. столбцов, добавляем h к выд. стркокам.
27: **Перейти к** 13 шагу алгоритма
28: **Конец условия**
29: **Конец условия**

Шаги алгоритма с 2 по 4 называются подготовительным этапом, с 5 по 29 — основным этапом.

Задача о назначениях для максимизации стоимости сводится к существующему алгоритму минимизации стоимости заменой целевой функции на

$$f_2(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} (M - c_{ij}) x_{ij} \to \min, \tag{2.2}$$

где $M = \max\limits_{i,j=\overline{1,n}} \{c_{ij}\}$.

3 Технологический раздел

3.1 Листинг программы

Листинг 3.1 — `lab01.m`

```
function lab01
  % Режим работы
  debug = true;
  maximize = false;

  debug_disp = @(varargin) debug_generic(debug, @disp, varargin{:});
  debug_fprintf = @(varargin) debug_generic(debug, @fprintf, varargin{:});
  debug_disp_matrix = @(varargin) debug_generic(debug, @disp_matrix, varargin{:});

  modes = [Минимизация, Максимизация];
  fprintf('[%s стоимости]\n', modes(1 + maximize));

  % Матрица стоимостей
  C = [10   8   6   4   9;
       11   9  10   5   6;
        5  10   8   6   4;
        3  11   9   6   6;
        8  10  11   8   7];

  disp('Матрица стоимостей:');
  disp(C);

  % Проверка квадратности матрицы стоимостей
  [height, width] = size(C);
  if height ~= width
    disp('Матрица не квадратная!');
    return;
  end

  % Проверка размерности матрицы стоимостей
  n = height;
  if n == 0
    disp('Матрица нулевой размерности!');
    return;
  end

  Ct = C;

  if maximize
    debug_disp('0. Сведём задачу максимизации к минимизации:');
    debug_disp('умножим элементы матрицы на -1 и прибавим максимальный по модулю элемент.');

    Ct = -Ct + max(Ct, [], 'all');

    debug_disp(' =');
    debug_disp(Ct);
  end


  debug_disp('[I] Подготовительный этап');
  debug_disp('1. Из каждого столбца матрицы вычтем его наименьший элемент');

  minInColumns = min(Ct);
```

```
54  Ct = Ct - minInColumns;
55
56  debug_disp('Наименьшие элементы в столбцах матрицы стоимостей:');
57  debug_disp(minInColumns);
58  debug_disp(' =');
59  debug_disp(Ct);
60
61
62  debug_disp('2. Из каждой строки матрицы вычтем её наименьший элемент');
63
64  minInRows = min(Ct, [], 2);
65  Ct = Ct - minInRows;
66
67  debug_disp('Наименьшие элементы в строках матрицы стоимостей:');
68  debug_disp(minInRows);
69  debug_disp(' =');
70  debug_disp(Ct);
71
72
73  debug_disp('3. Строим начальную СНН:');
74  debug_disp('Просмотрим столбцы текущей матрицы стоимостей (в порядке возрастания номера
        столбца) сверху вниз.');
75  debug_disp('Первый в строке нуль, в одной строке с которым нет 0*, отмечаем 0*.');
76
77  stars = initStars(Ct, n);
78  strokes = false(n);
79  colsBusy = false([1 n]);
80  rowsBusy = false([n 1]);
81
82  debug_disp(' =');
83  debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);
84
85
86  debug_disp('4. k := |СНН|');
87
88  k = sum(stars, 'all');
89
90  debug_fprintf('k = %d\n\n', k);
91
92
93  debug_disp('[II] Основной этап');
94
95  found_h = false;
96  iteration = 1;
97  while k ~= n
98    if ~found_h
99      debug_fprintf('-- Итерация %d\n', iteration);
100   end
101   found_h = false;
102
103   debug_disp('5. Столбцы с 0* отмечаем +');
104
105   colsBusy = fillColsBusy(colsBusy, stars, n);
106   rowsBusy(:) = false;
107
108   debug_disp(' =');
109   debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);
110
111
112   runInnerWhile = true;
113   while runInnerWhile
```

```
    runInnerWhile = false;
    runOuterWhile = false;
    h = Inf;
    for col = setdiff(1:n, find(colsBusy)) % col = 1:n except indices in colsBusy
      for row = setdiff(1:n, find(rowsBusy)) % row = 1:n except indices in rowsBusy
        if Ct(row, col) == 0
          debug_disp(6. Среди невыделенных есть 0, отмечаем его 0':);

          strokes(row, col) = true;

          debug_disp(' =');
          debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);


          idx = find(stars(row, :), 1);
          if ~isempty(idx)
            debug_disp(7. В одной строке с текущим 0' есть 0*, поэтому);
            debug_disp(снимаем выделение со столбца с этим 0*, выделяем строку с этим 0');

            colsBusy(idx) = false;
            rowsBusy(row) = true;

            debug_disp(' =');
            debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);

            runInnerWhile = true;
            break;
          end


          debug_disp(8. В одной строке с текущим 0' нет 0*, поэтому);
          debug_disp(строим непродолжаемую L-цепочку: от текущего 0' по столбцу в 0* по строке
              ... по строке в 0');

          Lchain = initLchain(stars, strokes, row, col);

          debug_disp('L-цепочка [row col]:');
          debug_disp(Lchain);


          debug_disp(9. В пределах L-цепочки меняем 0* на 0, а 0' на 0*);

          [stars, strokes] = processLchain(stars, strokes, Lchain);

          debug_disp(' =');
          debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);


          debug_disp(10. Снимаем все выделения, k := |CHH|);

          colsBusy(:) = false;
          rowsBusy(:) = false;

          debug_disp(' =');
          debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);

          k = sum(stars, 'all');
          debug_fprintf('k = %d\n', k);

          runOuterWhile = true;
          break;
```

```
          elseif Ct(row, col) < h
            h = Ct(row, col);
          end
        end

        if runInnerWhile || runOuterWhile
          break;
        end
      end

      if ~runInnerWhile && ~runOuterWhile
        debug_disp('11. Среди невыделенных элементов нет 0, поэтому');
        debug_disp('найдём h  минимальный элемент среди невыделенных.');
        debug_fprintf('h = %d\n', h);

        debug_disp('Вычтем h из невыделенных столбцов.');
        Ct(:, ~colsBusy) = Ct(:, ~colsBusy) - h;
        debug_disp(' =');
        debug_disp(Ct);

        debug_disp('Добавим h к выделенным строкам.');
        Ct(rowsBusy, :) = Ct(rowsBusy, :) + h;
        debug_disp(' =');
        debug_disp(Ct);

        found_h = true;
        iteration = iteration - 1;
      end
    end

    iteration = iteration + 1;
  end


  debug_disp('12. k = n, запишем оптимальное решение');

  disp('Оптимальное решение: X* =');
  disp(stars);

  f = sum(C .* stars, 'all');
  fprintf('f* = %d\n', f);
end

function stars = initStars(Ct, n)
  stars = zeros(n);
  rowsBusy = false([n 1]);
  for col = 1:n
    for row = 1:n
      if Ct(row, col) == 0 && ~rowsBusy(row)
        stars(row, col) = 1;
        rowsBusy(row) = 1;
        break;
      end
    end
  end
end

function colsBusy = fillColsBusy(colsBusy, stars, n)
  for col = 1:n
    colsBusy(col) = ~isempty(find(stars(:, col), 1));
  end
```

```
end

function Lchain = initLchain(stars, strokes, row_init, col_init)
  row = row_init;
  col = col_init;
  Lchain = [row col];
  row = find(stars(:, col), 1);
  while ~isempty(row)
    Lchain = [Lchain; row col];
    col = find(strokes(row, :), 1);
    Lchain = [Lchain; row col];
    row = find(stars(:, col), 1);
  end
end

function [stars, strokes] = processLchain(stars, strokes, Lchain)
  Lrows = size(Lchain, 1);

  for i = 1:2:Lrows
    x = Lchain(i, 1);
    y = Lchain(i, 2);
    strokes(x, y) = false;
    stars(x, y) = true;
  end

  for i = 2:2:Lrows-1
    x = Lchain(i, 1);
    y = Lchain(i, 2);
    stars(x, y) = false;
  end
end

function debug_generic(debug, func, varargin)
  if debug
    func(varargin{:});
  end
end

function disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy)
  addition_symbols = [ , *, '];
  busy_symbols = [ , +];
  [h, w] = size(Ct);
  for i = 1:h
    fprintf(' ');
    for j = 1:w
      fprintf('%5d', Ct(i, j));
      fprintf('%c', addition_symbols(1 + stars(i, j) + 2 * strokes(i, j)));
    end
    fprintf(' %c\n', busy_symbols(1 + rowsBusy(i)));
  end

  for j = 1:w
    fprintf('%6c', busy_symbols(1 + colsBusy(j)));
  end
  fprintf('\n');
end
```

4 Исследовательский раздел

4.1 Результаты расчётов для задач из индивидуального варианта

4.1.1 Минимизация

```
Оптимальное решение: X* =
     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* = 28
```

4.1.2 Максимизация

```
Оптимальное решение: X* =
     0     0     0     0     1
     1     0     0     0     0
     0     0     0     1     0
     0     1     0     0     0
     0     0     1     0     0

f* = 48
```