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»

ОТЧЕТ

по Лабораторной работе №1

по курсу «Методы вычислений»

на тему: «Венгерский метод решения задачи о назначениях»

Вариант № 17

Студент группы ИУ7И-12М _____ (Подпись, дата) Динь Вьет Ань (Фамилия И.О.)

Преподаватель _____ (Подпись, дата) Власов П.А. (Фамилия И.О.)

2024 г.

Содержание

1 Теоретическая часть

Цель работы: изучение венгерского метода решения задачи о назначениях.

Задание:

1) Реализовать венгерский метод решения задачи о назначениях в виде программы на ЭВМ.

2) Провести решение задачи с матрицей стоимостей, заданной в индивидуальном варианте, рассмотрев два случая:

   — задача о назначениях является задачей минимизации,

   — задача о назначениях является задачей максимизации.

1.1 Содержательная и математическая постановки задачи о назаначениях

Содержательная постановка: имеется n работ и n исполнителей; стоимость выполнения i-ой работы j-ым исполнителем составляет $c_{ij} \geq 0$ единиц. Требуется распределить все работы между исполнителями так, чтобы:

— каждый исполнитель выполнял 1 работу;

— каждую работу выполнял только 1 исполнитель;

— общая стоимость выполнения всех работ была min.

Введем управляемые переменные:

$$x_{ij} = \begin{cases} 1, \text{ если } i\text{-ую работу выполняет } j\text{-ый работник}, \\ 0, \text{ иначе}; \end{cases}$$
$$i, j = \overline{1; n}. \tag{1.1}$$

Из переменных x_{ij}, $i, j = \overline{1; n}$, составим

$$X = (x_{ij})_{i,j=\overline{1;n}}, \tag{1.2}$$

которую назовем матрицей назначений.

Стоимости выполнения работ также записываем в матрицу

$$C = (c_{ij})_{i,j=\overline{1;n}}, \tag{1.3}$$

называемой матрицей стоимостей.

Тогда:

1) Стоимость выполнения работ:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}. \tag{1.4}$$

2) Условие того, что i-ую работу выполнит один работник:

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1;n}. \tag{1.5}$$

3) Условие того, что j-ый работник выполнит одну работу:

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = \overline{1;n}. \tag{1.6}$$

Таким образом приходим к **математической постановке**:

$$\begin{cases} f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \to min, \\ \sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1;n}, \\ \sum_{i=1}^{n} x_{ij} = 1,\ j = \overline{1;n}, \\ x_{ij} \in \{0, 1\},\ i, j = \overline{1;n}. \end{cases} \tag{1.7}$$

1.2 Исходные данные конкретного варианта

Вариант **17**:

$$C = \begin{bmatrix} 8 & 10 & 5 & 6 & 4 \\ 11 & 10 & 9 & 8 & 7 \\ 6 & 8 & 10 & 4 & 9 \\ 10 & 9 & 11 & 5 & 6 \\ 9 & 11 & 3 & 6 & 6 \end{bmatrix} \tag{1.8}$$

1.3 Краткое описание венгерского метода

Схема венгерского метода решения задачи о назначениях представлена на рисунках 1.1–1.2.

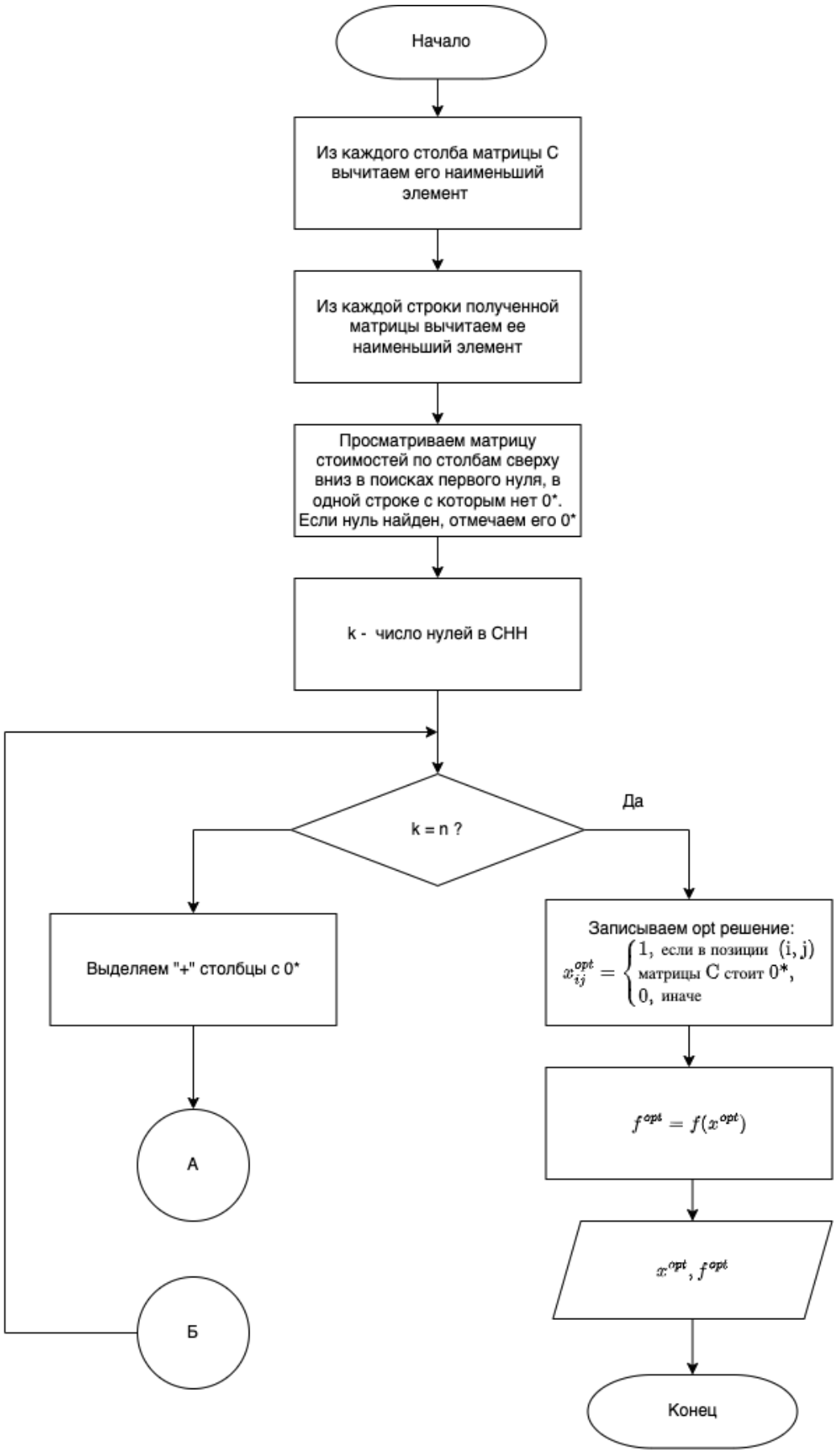

Рисунок 1.1 – Схема венгерского метода решения задачи о назначениях (часть 1)

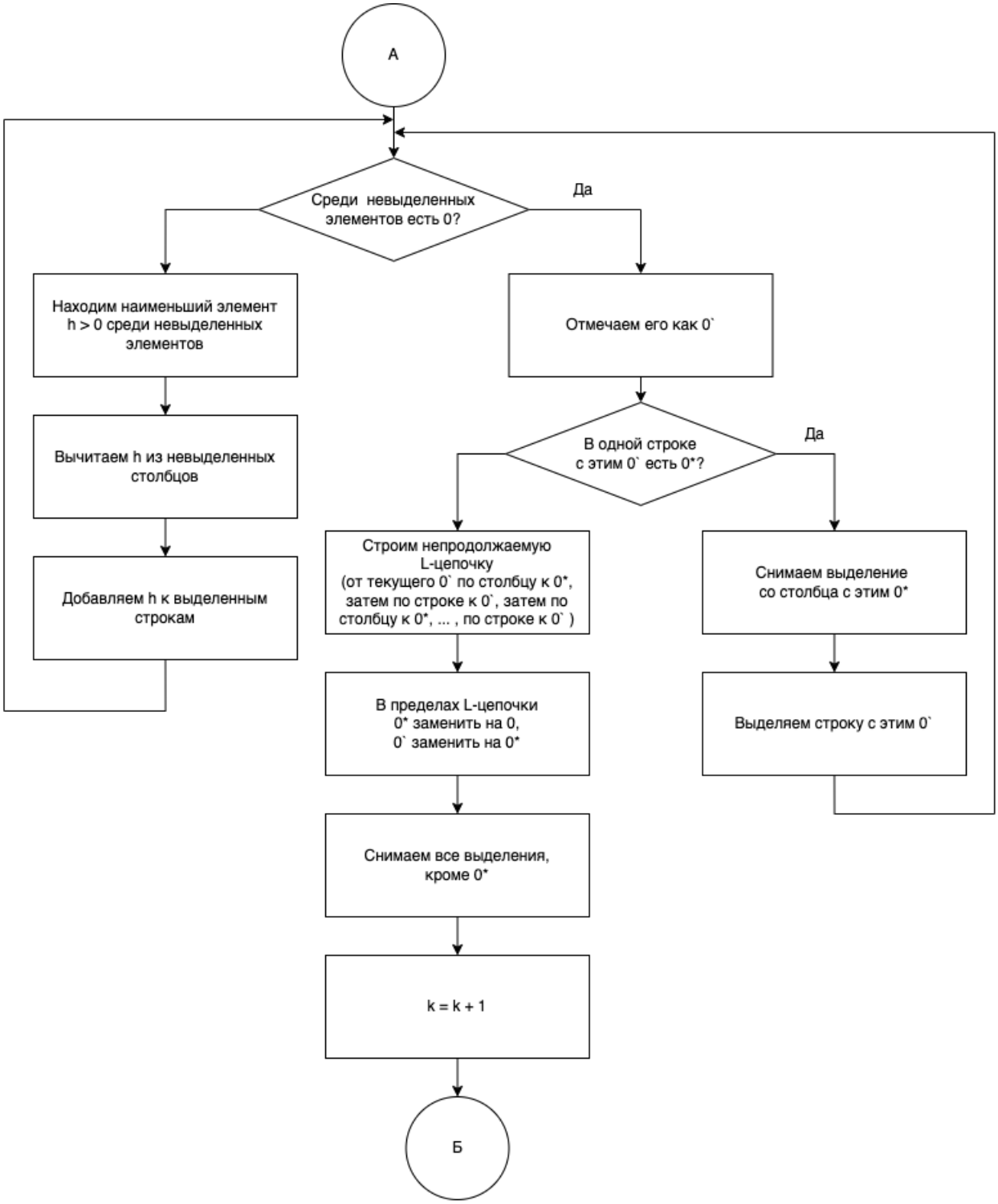

Рисунок 1.2 – Схема венгерского метода решения задачи о назначениях (часть 2)

2 Практическая часть

2.1 Текст программы

В листинге 2.1 представлен код программы.

Листинг 2.1 – Код программы

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

  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 = [
            8 10 5 6 4;
            11 10 9 8 7;
            6 8 10 4 9;
            10 9 11 5 6;
            9 11 3 6 6
        ];

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

  % Проверка матрицы
  [height, width] = size(C);
  if height ~= width || height == 0
    disp('Неправильный размер матрицы!');
    return;
  end

  n = height;
```

```
33
34  Ct = C;
35
36  if maximize
37    debug_disp('0. Сведём задачу максимизации к минимизации:');
38    debug_disp('умножим элементы матрицы на -1 и прибавим
        максимальный по модулю элемент.');
39
40    Ct = -Ct + max(Ct, [], 'all');
41
42    debug_disp('C~ =');
43    debug_disp(Ct);
44  end
45
46  debug_disp('[I] Подготовительный этап');
47  debug_disp('1. Из каждого столбца матрицы вычтем его
      наименьший элемент');
48
49  minInColumns = min(Ct);
50  Ct = Ct - minInColumns;
51
52  debug_disp('Наименьшие элементы в столбцах матрицы
      стоимостей:');
53  debug_disp(minInColumns);
54  debug_disp('C~ =');
55  debug_disp(Ct);
56
57  debug_disp('2. Из каждой строки матрицы вычтем её наименьший
      элемент');
58
59  minInRows = min(Ct, [], 2);
60  Ct = Ct - minInRows;
61
62  debug_disp('Наименьшие элементы в строках матрицы
      стоимостей:');
63  debug_disp(minInRows);
64  debug_disp('C~ =');
65  debug_disp(Ct);
66
67  debug_disp('3. Строим начальную СНН:');
68  debug_disp('Просмотрим столбцы текущей матрицы стоимостей (в
```

```
    порядке возрастания номера столбца) сверху вниз.');
69  debug_disp('Первый в столбце нуль, в одной строке с которым
    нет 0*, отмечаем 0*.');
70
71  stars = initStars(Ct, n);
72  strokes = false(n);
73  colsBusy = false([1 n]);
74  rowsBusy = false([n 1]);
75
76  debug_disp('C~ =');
77  debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);
78
79  debug_disp('4. k := |СНН|');
80
81  k = sum(stars, 'all');
82  debug_fprintf('k = %d\n\n', k);
83
84  debug_disp('[II] Основной этап');
85
86  iteration = 1;
87  while k ~= n
88    debug_fprintf('-- Итерация %d\n', iteration);
89    debug_disp('5. Столбцы с 0* отмечаем "+"');
90
91    colsBusy = fillColsBusy(colsBusy, stars, n);
92
93    debug_disp('C~ =');
94    debug_disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy);
95
96    runInnerWhile = true;
97    while runInnerWhile
98      runInnerWhile = false;
99      runOuterWhile = false;
100     h = Inf;
101     for col = setdiff(1:n, find(colsBusy)) % col = 1:n
          except indices in colsBusy
102       for row = setdiff(1:n, find(rowsBusy)) % row = 1:n
            except indices in rowsBusy
103         if Ct(row, col) == 0
104           debug_disp("6. Среди невыделенных есть 0, отмечаем
                его 0':");
```

```
105
106             strokes(row, col) = true;
107
108             debug_disp('C~ =');
109             debug_disp_matrix(Ct, stars, strokes, colsBusy,
                    rowsBusy);
110
111             idx = find(stars(row, :), 1);
112             if ~isempty(idx)
113               debug_disp("7. В одной строке с текущим 0' есть
                      0*, поэтому");
114               debug_disp("снимаем выделение со столбца с этим
                      0*, выделяем строку с этим 0'");
115
116               colsBusy(idx) = false;
117               rowsBusy(row) = true;
118
119               debug_disp('C~ =');
120               debug_disp_matrix(Ct, stars, strokes, colsBusy,
                      rowsBusy);
121
122               runInnerWhile = true;
123               break;
124             end
125
126             debug_disp("8. В одной строке с текущим 0' нет 0*,
                    поэтому");
127             debug_disp("строим непродолжаемую L-цепочку: от
                    текущего 0' по столбцу в 0* по строке ... по
                    строке в 0'");
128
129             Lchain = initLchain(stars, strokes, row, col);
130
131             debug_disp('L-цепочка [row col]:');
132             debug_disp(Lchain);
133
134             debug_disp("9. В пределах L-цепочки меняем 0* на
                    0, а 0' на 0*");
135
136             [stars, strokes] = processLchain(stars, strokes,
                    Lchain);
```

```
137
138             debug_disp('C~ =');
139             debug_disp_matrix(Ct, stars, strokes, colsBusy,
                    rowsBusy);
140
141             debug_disp("10. Снимаем все выделения, k :=
                    |CHH|");
142
143             colsBusy(:) = false;
144             rowsBusy(:) = false;
145             strokes(:) = false;
146
147             debug_disp('C~ =');
148             debug_disp_matrix(Ct, stars, strokes, colsBusy,
                    rowsBusy);
149
150             k = sum(stars, 'all');
151             debug_fprintf('k = %d\n', k);
152
153             runOuterWhile = true;
154             break;
155           elseif Ct(row, col) < h
156             h = Ct(row, col);
157           end
158         end
159
160         if runInnerWhile || runOuterWhile
161           break;
162         end
163       end
164
165       if ~runInnerWhile && ~runOuterWhile
166         debug_disp('11. Среди невыделенных элементов нет 0,
                поэтому');
167         debug_disp('найдём h  минимальный элемент среди
                невыделенных.');
168         debug_fprintf('h = %d\n', h);
169
170         debug_disp('Вычтем h из невыделенных столбцов.');
171         Ct(:, ~colsBusy) = Ct(:, ~colsBusy) - h;
172         debug_disp('C~ =');
```

```
173         debug_disp_matrix(Ct, stars, strokes, colsBusy,
                rowsBusy);
174
175         debug_disp('Добавим h к выделенным строкам.');
176         Ct(rowsBusy, :) = Ct(rowsBusy, :) + h;
177         debug_disp('C~ =');
178         debug_disp_matrix(Ct, stars, strokes, colsBusy,
                rowsBusy);
179
180         runInnerWhile = true;
181       end
182     end
183
184     iteration = iteration + 1;
185   end
186
187   debug_disp('12. k = n, запишем оптимальное решение');
188   disp('Оптимальное решение: X* =');
189   disp(stars);
190
191   f = sum(C .* stars, 'all');
192   fprintf('Стоимость: f* = %d\n', f);
193 end
194
195 function stars = initStars(Ct, n)
196   stars = zeros(n);
197   rowsBusy = false([n 1]);
198   for col = 1:n
199     for row = 1:n
200       if Ct(row, col) == 0 && ~rowsBusy(row)
201         stars(row, col) = 1;
202         rowsBusy(row) = 1;
203         break;
204       end
205     end
206   end
207 end
208
209 function colsBusy = fillColsBusy(colsBusy, stars, n)
210   for col = 1:n
211     colsBusy(col) = ~isempty(find(stars(:, col), 1));
```

```
212   end
213 end
214
215 function Lchain = initLchain(stars, strokes, row_init,
      col_init)
216   row = row_init;
217   col = col_init;
218   Lchain = [row col];
219   row = find(stars(:, col), 1);
220   while ~isempty(row)
221     Lchain = [Lchain; row col];
222     col = find(strokes(row, :), 1);
223     Lchain = [Lchain; row col];
224     row = find(stars(:, col), 1);
225   end
226 end
227
228 function [stars, strokes] = processLchain(stars, strokes,
      Lchain)
229   Lrows = size(Lchain, 1);
230
231   for i = 1:2:Lrows
232     x = Lchain(i, 1);
233     y = Lchain(i, 2);
234     strokes(x, y) = false;
235     stars(x, y) = true;
236   end
237
238   for i = 2:2:Lrows-1
239     x = Lchain(i, 1);
240     y = Lchain(i, 2);
241     stars(x, y) = false;
242   end
243 end
244
245 function debug_generic(debug, func, varargin)
246   if debug
247     func(varargin{:});
248   end
249 end
250
```

```
251 function disp_matrix(Ct, stars, strokes, colsBusy, rowsBusy)
252   addition_symbols = [" ", "*", "'"];
253   busy_symbols = [" ", "+"];
254   [h, w] = size(Ct);
255   for i = 1:h
256     fprintf(' ');
257     for j = 1:w
258       fprintf('%5d', Ct(i, j));
259       fprintf('%c', addition_symbols(1 + stars(i, j) + 2 *
              strokes(i, j)));
260     end
261     fprintf(' %c\n', busy_symbols(1 + rowsBusy(i)));
262   end
263
264   for j = 1:w
265     fprintf('%6c', busy_symbols(1 + colsBusy(j)));
266   end
267   fprintf('\n');
268 end
```

2.2 Результаты расчетов для задач из индивидуального варианта

В листинге 2.2 представлены расчеты для задачи минимизации.

Листинг 2.2 – Задача минимизации

```
[Минимизация стоимости]
Матрица стоимостей:
     8    10     5     6     4
    11    10     9     8     7
     6     8    10     4     9
    10     9    11     5     6
     9    11     3     6     6

[I] Подготовительный этап
1. Из каждого столбца матрицы вычтем его наименьший элемент
Наименьшие элементы в столбцах матрицы стоимостей:
     6     8     3     4     4

C~ =
     2     2     2     2     0
     5     2     6     4     3
     0     0     7     0     5
     4     1     8     1     2
     3     3     0     2     2

2. Из каждой строки матрицы вычтем её наименьший элемент
Наименьшие элементы в строках матрицы стоимостей:
     0
     2
     0
     1
     0

C~
     2     2     2     2     0
     3     0     4     2     1
     0     0     7     0     5
     3     0     7     0     1
     3     3     0     2     2

3. Строим начальную СНН:
```

```
Просмотрим столбцы текущей матрицы стоимостей (в порядке
   возрастания номера столбца) сверху вниз.
Первый в столбце нуль, в одной строке с которым нет 0*,
   отмечаем 0*.
C~ =
     2     2     2     2     0*
     3     0*    4     2     1
     0*    0     7     0     5
     3     0     7     0*    1
     3     3     0*    2     2

4. k := |CHH|
k = 5

[II] Основной этап
12. k = n, запишем оптимальное решение
Оптимальное решение: X* =
     0     0     0     0     1
     0     1     0     0     0
     1     0     0     0     0
     0     0     0     1     0
     0     0     1     0     0

Стоимость: f* = 28
```

В листинге 2.3 представлены расчеты для задачи максимизации.

Листинг 2.3 – Задача максимизации

```
[Максимизация стоимости]
Матрица стоимостей:
     8    10     5     6     4
    11    10     9     8     7
     6     8    10     4     9
    10     9    11     5     6
     9    11     3     6     6

0. Сведём задачу максимизации к минимизации:
умножим элементы матрицы на -1 и прибавим максимальный по
   модулю элемент.
C~ =
     3     1     6     5     7
     0     1     2     3     4
```

```
     5     3     1     7     2
     1     2     0     6     5
     2     0     8     5     5

[I] Подготовительный этап
1. Из каждого столбца матрицы вычтем его наименьший элемент
Наименьшие элементы в столбцах матрицы стоимостей:
     0     0     0     3     2

C~ =
     3     1     6     2     5
     0     1     2     0     2
     5     3     1     4     0
     1     2     0     3     3
     2     0     8     2     3

2. Из каждой строки матрицы вычтем её наименьший элемент
Наименьшие элементы в строках матрицы стоимостей:
     1
     0
     0
     0
     0

C~ =
     2     0     5     1     4
     0     1     2     0     2
     5     3     1     4     0
     1     2     0     3     3
     2     0     8     2     3

3. Строим начальную СНН:
Просмотрим столбцы текущей матрицы стоимостей (в порядке
   возрастания номера столбца) сверху вниз.
Первый в столбце нуль, в одной строке с которым нет 0*,
   отмечаем 0*.
C~ =
     2     0*    5     1     4
     0*    1     2     0     2
     5     3     1     4     0*
     1     2     0*    3     3
```

```
     2     0     8     2     3

4. k := |СНН|
k = 4

[II] Основной этап
-- Итерация 1
5. Столбцы с 0* отмечаем "+"
C~ =
     2     0*    5     1     4
     0*    1     2     0     2
     5     3     1     4     0*
     1     2     0*    3     3
     2     0     8     2     3
     +     +     +           +
6. Среди невыделенных есть 0, отмечаем его 0':
C~ =
     2     0*    5     1     4
     0*    1     2     0'    2
     5     3     1     4     0*
     1     2     0*    3     3
     2     0     8     2     3
     +     +     +           +
7. В одной строке с текущим 0' есть 0*, поэтому
снимаем выделение со столбца с этим 0*, выделяем строку с этим
   0'
C~ =
     2     0*    5     1     4
     0*    1     2     0'    2  +
     5     3     1     4     0*
     1     2     0*    3     3
     2     0     8     2     3
           +     +           +
11. Среди невыделенных элементов нет 0, поэтому
найдём h  минимальный элемент среди невыделенных.
h = 1
Вычтем h из невыделенных столбцов.
C~ =
     1     0*    5     0     4
    -1*    1     2    -1'    2  +
     4     3     1     3     0*
```

```
     0      2      0*     2      3
     1      0      8      1      3
            +      +             +
Добавим h к выделенным строкам.
C~ =
     1      0*     5      0      4
     0*     2      3      0'     3  +
     4      3      1      3      0*
     0      2      0*     2      3
     1      0      8      1      3
            +      +             +
6. Среди невыделенных есть 0, отмечаем его 0':
C~ =
     1      0*     5      0      4
     0*     2      3      0'     3  +
     4      3      1      3      0*
     0'     2      0*     2      3
     1      0      8      1      3
            +      +             +
7. В одной строке с текущим 0' есть 0*, поэтому
снимаем выделение со столбца с этим 0*, выделяем строку с этим
   0'
C~ =
     1      0*     5      0      4
     0*     2      3      0'     3  +
     4      3      1      3      0*
     0'     2      0*     2      3  +
     1      0      8      1      3
            +                    +
6. Среди невыделенных есть 0, отмечаем его 0':
C~ =
     1      0*     5      0'     4
     0*     2      3      0'     3  +
     4      3      1      3      0*
     0'     2      0*     2      3  +
     1      0      8      1      3
            +                    +
7. В одной строке с текущим 0' есть 0*, поэтому
снимаем выделение со столбца с этим 0*, выделяем строку с этим
   0'
C~ =
```

```
     1      0*     5      0'     4   +
     0*     2      3      0'     3   +
     4      3      1      3      0*
     0'     2      0*     2      3   +
     1      0      8      1      3
                                  +
6. Среди невыделенных есть 0, отмечаем его 0':
C~ =
     1      0*     5      0'     4   +
     0*     2      3      0'     3   +
     4      3      1      3      0*
     0'     2      0*     2      3   +
     1      0'     8      1      3
                                  +
8. В одной строке с текущим 0' нет 0*, поэтому
строим непродолжаемую L-цепочку: от текущего 0' по столбцу в
   0* по строке ... по строке в 0'
L-цепочка [row col]:
     5      2
     1      2
     1      4

9. В пределах L-цепочки меняем 0* на 0, а 0' на 0*
C~ =
     1      0      5      0*     4   +
     0*     2      3      0'     3   +
     4      3      1      3      0*
     0'     2      0*     2      3   +
     1      0*     8      1      3
                                  +
10. Снимаем все выделения, k := |СНН|
C~ =
     1      0      5      0*     4
     0*     2      3      0      3
     4      3      1      3      0*
     0      2      0*     2      3
     1      0*     8      1      3

k = 5
12. k = n, запишем оптимальное решение
Оптимальное решение: X* =
```

```
     0      0      0      1      0
     1      0      0      0      0
     0      0      0      0      1
     0      0      1      0      0
     0      1      0      0      0

Стоимость: f* = 48
```