Московский государственный технический университет
имени Н. Э. Баумана

Факультет: Информатика и системы управления
Кафедра: Программное обеспечение ЭВМ и информационные технологии
Дисциплина: Методы вычислений

Лабораторная работа №1
Венгерский метод решения задачи о назначениях

Выполнил: Исполатов Филипп Олегович
Группа: ИУ7-12М
Вариант: 7

Москва, 2020г.

Содержание

1 Постановка задачи

1.1 Содержательная постановка задачи

Пусть есть n работ и n исполнителей. Стоимость выпонения i-й работы j-м исполнителем составляет $C_(ij) >= 0$. Требуется распределить все работы так, чтобы:

- каждый исполнитель выполнял одну работу;
- общая стоимость выполнения всех работ была минимальной/максимальной;

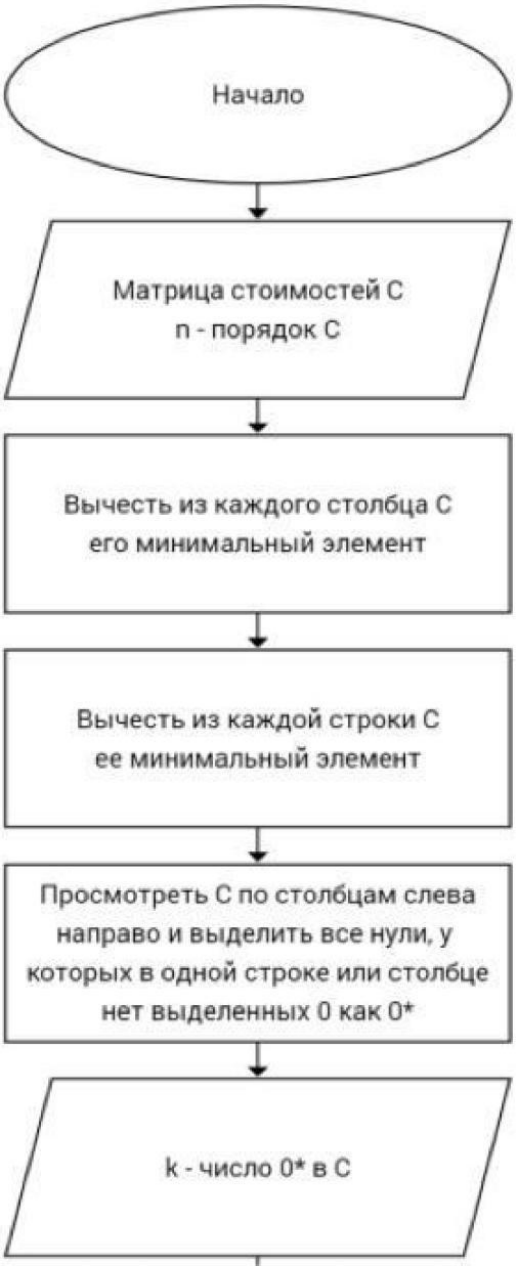
1.2 Математическая постановка задачи

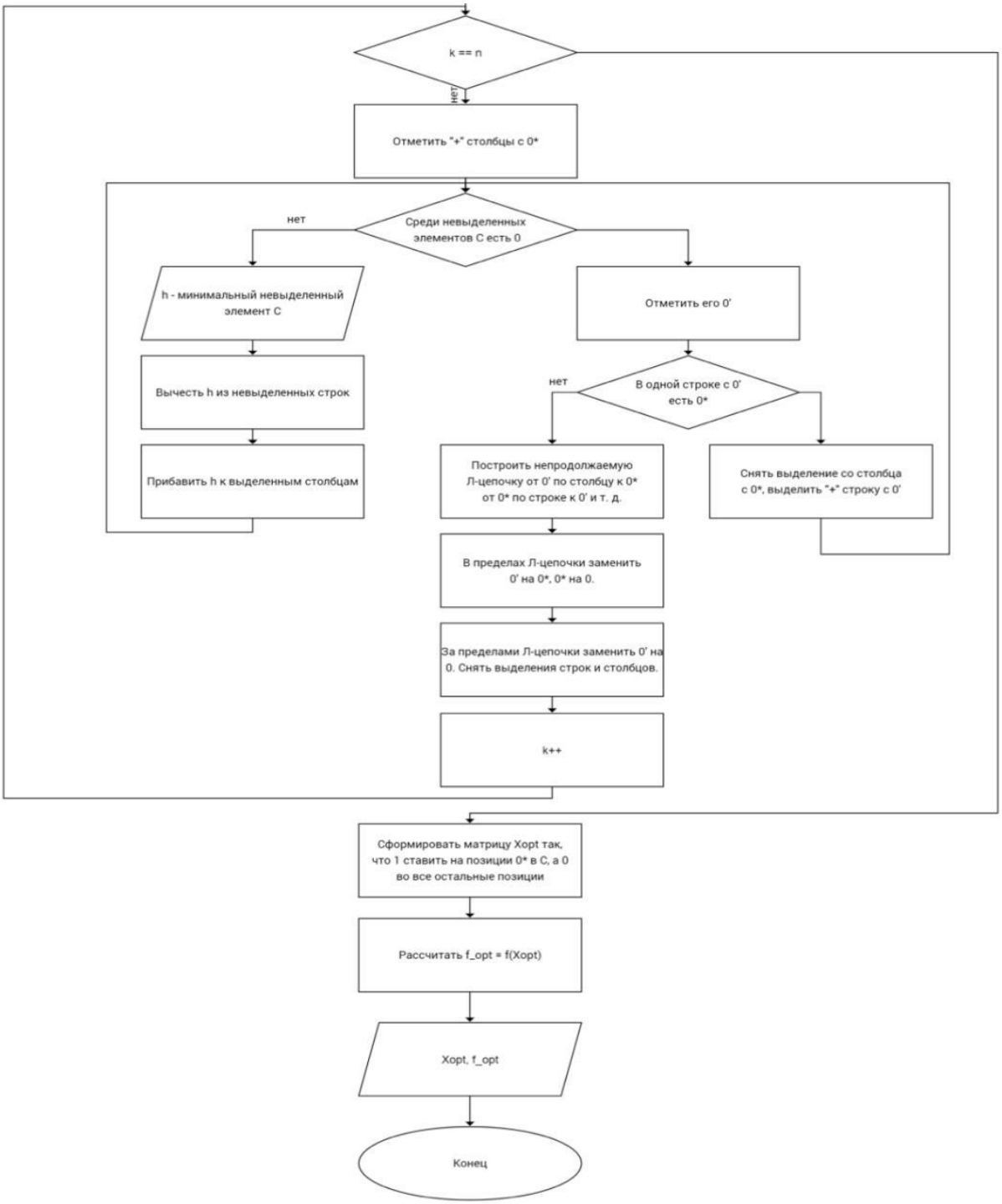
$$\begin{cases} f = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} c_{ij} \to min \\ \sum_{j=1}^{n} x_{ij} = 1, i = \overline{1,n} \\ \sum_{i=1}^{n} x_{ij} = 1, j = \overline{1,n} \\ x_{ij} \in 0,1 \end{cases}$$, где f - целевая функция, x - матрица назначений

2 Венгерский метод решения задачи о назначениях

Данный метод применяется для решения задачи о назначениях в форме задачи минимизации. Если матрица C интерпретируется как матрица прибыли, задача о назначениях принимает форму задачи максимизации. Для сведения данной задачи к задаче минимизации необходимо преобразовать матрицу C.

3 Схема алгоритма Венгерского метода




k == n
нет
Отметить "+" столбцы с 0*
Среди невыделенных элементов С есть 0
нет
h - минимальный невыделенный элемент С
Вычесть h из невыделенных строк
Прибавить h к выделенным столбцам
Отметить его 0'
В одной строке с 0' есть 0*
нет
Построить непродолжаемую Л-цепочку от 0' по столбцу к 0* от 0* по строке к 0' и т. д.
Снять выделение со столбца с 0*, выделить "+" строку с 0'
В пределах Л-цепочки заменить 0' на 0*, 0* на 0.
За пределами Л-цепочки заменить 0' на 0. Снять выделения строк и столбцов.
k++
Сформировать матрицу Xopt так, что 1 ставить на позиции 0* в С, а 0 во все остальные позиции
Рассчитать f_opt = f(Xopt)
Xopt, f_opt
Конец

4 Текст программы

```
1  global mode;
2  global solution;
3  mode = "DEBUG";
4  solution = "MAX";
5
6  [matr, count] = ReadDataFromFile('test.txt');
7  Method(matr, count);
8
9  function Method(matr, count)
10     src = matr;
11     global mode;
12     global solution;
13     if mode == "DEBUG"
14         printMatrix(matr, count);
15     end
16     if solution == "MAX"
17         matr = ToMax(matr, count);
18         if mode == "DEBUG"
19             fprintf("Inverse\n");
20             printMatrix(matr, count);
21         end
22     end
23     matr = MinusRow(matr, count);
24     if mode == "DEBUG"
25         printMatrix(matr, count);
26     end
27     matr = MinusCol(matr, count);
28     if mode == "DEBUG"
29         printMatrix(matr, count);
30     end
31     [matr, countStar] = MarkByStar(matr, count);
32     if mode == "DEBUG"
33         printMatrix(matr, count);
34     end
35
36     for iter = 1:count-countStar
37
38         [selectedColmns, selectedRows] = SelectColumns(matr, count);
39         [matr, row, col] = ResolveMatrix(matr, selectedColmns, selectedRows, cou
40         [imas, jmas, cl] = LChain(matr, count, row, col);
41         if mode == "DEBUG"
42             printMatrix(matr, count);
43         end
44         matr = Replace(matr, imas, jmas, cl);
45         if mode == "DEBUG"
46             printMatrix(matr, count);
47         end
48         res = Result(src, matr, count);
```

```
49      end
50  end
51
52  function [matr] = ToMax(matr, count)
53      max = matr{1, 1}.value;
54      for i = 1:count
55          for j = 1:count
56              if matr{i,j}.value > max
57                  max = matr{i,j}.value;
58              end
59          end
60      end
61      for i = 1:count
62          for j = 1:count
63                  matr{i,j}.value = matr{i,j}.value * -1;
64          end
65      end
66      for i = 1:count
67          for j = 1:count
68                  matr{i,j}.value = matr{i,j}.value + max;
69          end
70      end
71  end
72
73  function [matrix, count] = ReadDataFromFile(filename)
74       fileID = fopen(filename, 'r');
75
76       count = fscanf(fileID, '%d', 1);
77
78       matrix = {cell(count, count)};
79
80       for i = 1:count
81           for j = 1:count
82               value = fscanf(fileID, '%f', 1);
83               matrix{i,j} = struct('value', value, 'mark', '');
84           end
85       end
86       fclose(fileID);
87  end
88
89  function [matrix] = MinusRow(matrix, count)
90      x = matrix{1,:};
91      for i = 1:count
92          min = matrix{i, 1}.value;
93          for j = 1:count
94              if matrix{i, j}.value < min
95                  min = matrix{i, j}.value;
96              end
97          end
98          for j = 1:count
```

```
99                  matrix{i, j}.value = matrix{i, j}.value - min;
100             end
101         end
102     end
103
104     function [matrix] = MinusCol(matrix, count)
105         x = matrix{1,:};
106         for i = 1:count
107             min = matrix{1, i}.value;
108             for j = 1:count
109                 if matrix{j, i}.value < min
110                     min = matrix{j, i}.value;
111                 end
112             end
113             for j = 1:count
114                 matrix{j, i}.value = matrix{j, i}.value - min;
115             end
116         end
117     end
118
119     function printMatrix(matr, count)
120         for i = 1:count
121             for j = 1:count
122                 fprintf('%f(%s) ', matr{i, j}.value, matr{i, j}.mark);
123             end
124             fprintf('\n');
125         end
126         fprintf('\n');
127     end
128
129     function [isEqual] = isEqual(toCompare, value)
130         isEqual = false;
131         if toCompare == value
132             isEqual = true;
133         end
134         return;
135     end
136
137     function [isValid, p] = isHaveStarZeroInRow(matr, row, count)
138         p = -1;
139         isValid = false;
140         for i = 1:count
141             if isEqual(matr{row,i}.value, 0) && isEqual(matr{row,i}.mark, '*')
142                 isValid = true;
143                 p = i;
144                 return;
145             end
146         end
147     end
148
```

```
149 function [matrix, countStar] = MarkByStar(matrix, count)
150     countStar = 0;
151     for i = 1:count
152         for j = 1:count
153             if matrix{j, i}.value == 0
154                 [r, ~] = isHaveStarZeroInRow(matrix, j, count);
155                 if r == false
156                     matrix{j, i}.mark = '*';
157                     countStar = countStar + 1;
158                     break;
159                 end
160             end
161         end
162     end
163 end
164
165 function [selectedColumns, selectedRows] = SelectColumns(matr, count)
166     selectedColumns = zeros(1, count);
167     selectedRows = zeros(1, count);
168   for i = 1:count
169       for j = 1:count
170           if matr{j, i}.mark == '*'
171               selectedColumns(i) = 1;
172           end
173       end
174   end
175 end
176
177 % ?
178 function [matrix, row, col] = ResolveMatrix(matrix, slctdCols, slctdRows, count
179     global mode;
180     while 1
181         haveZeros = false;
182         if mode == "DEBUG"
183             printMatrix(matrix, count);
184         end
185         flag = true;
186         for i = 1:count
187             if slctdCols(i) == 0 && flag == true
188                 for j = 1:count
189                     if slctdRows(j) == 0 && flag == true
190                         if isEqual(matrix{j, i}.value, 0) && ~isEqual(matrix{j,
191                             haveZeros = true;
192                             matrix{j, i}.mark = '.';
193                             [r, p] = isHaveStarZeroInRow(matrix, j, count);
194                             if r
195                                 slctdRows(j) = 1;
196                                 slctdCols(p) = 0;
197                                 flag = false;
198                                 break;
```

```
199                                 end
200                                 if ~r
201                                     row = j;
202                                     col = i;
203                                     return;
204                                 end
205                             end
206                         end
207                     end
208                 end
209             end
210             if haveZeros ~= true
211                 min = 99999999999;
212                 for i = 1:count
213                     if slctdCols(i) == 0
214                         for j = 1:count
215                             if slctdRows(j) == 0
216                                 if matrix{j, i}.value < min
217                                     min = matrix{j, i}.value;
218                                 end
219                             end
220                         end
221
222                     end
223                 end
224                 for i = 1:length(slctdCols)
225                     if slctdCols(i) == 0
226                         for j = 1:count
227                             matrix{j,i}.value = matrix{j,i}.value - min;
228                         end
229
230                     end
231                 end
232                 for i = 1:length(slctdRows)
233                     if slctdRows(i) == 1
234                         for j = 1:count
235                             matrix{i,j}.value = matrix{i,j}.value + min;
236                         end
237
238                     end
239                 end
240             end
241         end
242 end
243
244 % min / max
245
246 function [row, col] = FindNextChain(matr, count, mark, ipos, jpos)
247     row = -1;
248     col = -1;
```

```
249     if mark == '.'
250         for i = 1:count
251             if matr{i,jpos}.mark == '*'
252                 row = i;
253                 col = jpos;
254                 return;
255             end
256         end
257     end
258     if mark == '*'
259         for j = 1:count
260             if matr{ipos,j}.mark == '.'
261                 row = ipos;
262                 col = j;
263                 return;
264             end
265         end
266     end
267 end
268
269 function [imas, jmas, i] = LChain(matrix, count, row, col)
270     imas = zeros(1, count);
271     jmas = zeros(1, count);
272     i = 1;
273     while row ~= -1 && col ~= -1
274         imas(i) = row;
275         jmas(i) = col;
276         [row, col] = FindNextChain(matrix, count, matrix{row, col}.mark, row, c
277         i = i + 1;
278     end
279 end
280
281 function [matrix] = Replace(matrix, imas, jmas, cl)
282     for i = 1:cl-1
283         if matrix{imas(i),jmas(i)}.mark == '*'
284             matrix{imas(i),jmas(i)}.mark = '';
285         end
286         if matrix{imas(i),jmas(i)}.mark == '.'
287             matrix{imas(i),jmas(i)}.mark = '*';
288         end
289     end
290 end
291
292 function res = Result(src, matr, count)
293     res = 0;
294     for i = 1:count
295         for j = 1:count
296             if matr{i, j}.mark == '*'
297                 res = res + src{i, j}.value;
298             end
```

```
299             end
300         end
301         fprintf('%f', res);
302 end
```

5 Результаты расчётов

Проведем решение двух вариантов задач о назначениях.

$$C = \begin{bmatrix} 11 & 4 & 11 & 6 & 11 \\ 7 & 5 & 6 & 7 & 12 \\ 9 & 7 & 8 & 10 & 10 \\ 9 & 11 & 6 & 10 & 9 \\ 7 & 10 & 4 & 8 & 8 \end{bmatrix}$$

5.1 Задача минимизации

Результат работы алгоритма - 33

5.2 Задача максимизации

Результат работы алгоритма - 52