Misr University for Science & Technology

Information Technology (CS)

Name: Ahmed Amgad Ahmed El-Kady

ID: 89483

(CS 331)

Algorithm project

(Assignment problem using Hungarian algorithm)

1. Assignment problem description

Assign **n** jobs to be executed be **n** people, each person is assigned to **exactly one** job and each job is assigned to exactly one person. The cost of assigning the **i**th person to the **j**th job is a known quantity, **C[i, j].** The following is a cost matrix for n = 4.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

2. Hungarian algorithm description

The Hungarian algorithm consists of the four steps below. The first two steps are executed once, while Steps 3 and 4 are repeated until an optimal assignment is found. The input of the algorithm is an n x n square matrix with only nonnegative elements.

Step 1: Subtract row minima:
For each row, find the lowest element and subtract it from each element in that row.

Step 2: Subtract column minima:
Similarly, for each column, find the lowest element and subtract it from each element in that column.

Step 3: Cover all zeros with a minimum number of lines:
Cover all zeros in the resulting matrix using a minimum number of horizontal and vertical lines. If n lines are required, an optimal assignment exists among the zeros. The algorithm stops. If less than n lines are required, continue with Step 4.

Step 4: Create additional zeros:
Find the smallest element (call it k) that is not covered by a line in Step 3. Subtract k from all uncovered elements.

3. Pseudo code

```
adjust_matrix( m[4][4])
        for i<--0 to 3 do
          minrow <-- m[0][0]
          for j<--0 to 3 do     //to get min row
            if minrow > m[i][j]
            minrow <-- m[i][j]

          for j<--0 to 3 do    //substract the min row
                m[i][j] <--m[i][j] - minrow

    for j<--0 to 3 do
          mincol <-- m[0][0]
          for i<--0 to 3 do    //get min column
                if mincol > m[i][j]
             mincol <-- m[i][j]
          for i<--0 to 3 do     //subtract min column
                m[i][j] <--m[i][j] - mincol

        return m[4][4]   //Adjusted matrix


drawLines(matrix[4][4], OriginalMatrix[4][4])
        rowZeroline[4] <-- false   //if true it acts as a drawn line
        colZeroline[4] <-- false   //if true it acts as a drawn line
        solMatrix[4]  //store selected job to each person

        //draw vertical line on zeros (if there is only 1 zero in the row)
        for i<--0 to 3 do
          ZerosNo <-- 0  //number of zeros in one row or column
          colNo <-- 0   //column location at which vertical line could be drawn
          for j<--0 to 3 do
```

```
            if (matrix[i][j] <-- 0 && colZeroline[j]<-- false) {
               ZerosNo<--ZerosNo +1
               colNo <-- j
            if (ZerosNo <-- 1 && colZeroline[colNo] <--false)
                colZeroline[colNo] <-- true  //drawn line
                solMatrix[i] <-- colNo

   //draw horizontal line on zeros (if there is only 1 zero in the column)
   for j<--0 to 3 do
ZerosNo <-- 0  //number of zeros in one row or column
rowNo <-- 0   //row location at which horizontal line could be drawn
      for i<--0 to 3 do
            if (matrix[i][j] <-- 0 && colZeroline[j] <-- false)
                ZerosNo<--ZerosNo +1
                rowNo <-- i

            if (ZerosNo <-- 1 && rowZeroline[rowNo] <-- false) {
                        rowZeroline[rowNo] <-- true
                        solMatrix[rowNo] <-- j

   for i<--0 to 3 do
            for j<--0 to 3 do
                        if (matrix[i][j] <-- 0 && colZeroline[j] <-- false && rowZeroline[i] <-- false)
                            rowZeroline[i] <-- true   //drawn line
                            solMatrix[i] <-- j

   //check that no zeros left
   NoLines <-- 0
   for m<--0 to 3 do
            if (rowZeroline[m] <-- true)
                        NoLines<-- NoLines + 1
            if (colZeroline[m] <-- true)
                        NoLines<-- NoLines + 1
```

```
        if (NoLines < 4)
                for i<--0 to 3 do
                        minno <-- matrix[0][0]   // min no zero elemnt
                        for j<--0 to 3 do
                                if (rowZeroline[i] <-- false && colZeroline[j] <-- false)
                                        if (minno > matrix[i][j] && matrix[i][j] > 0)
                                                minno <-- matrix[i][j]
                        for j<--0 to 3 do   //substract the min number from unmarked numbers
                                if (rowZeroline[i] <-- false && colZeroline[j] <-- false)
                                        matrix[i][j] <-- matrix[i][j]-minno
                drawLines(matrix, OriginalMatrix)

        //display jobs assignment
        for i<--0 to 3 do
                print<<" Person " << i + 1 << " do job no. " << solMatrix[i] + 1

        sum <--0
        for k<--0 to 3 do
                sum <-- sum + OriginalMatrix[k][solMatrix[k]]
        return sum

main()
  matrix[4][4] <-- { 9,2,7,8,6,4,3,7,5,8,1,8,7,6,9,4 }
  OriginalMatrix[4][4] <-- { 9,2,7,8,6,4,3,7,5,8,1,8,7,6,9,4 }
        matrix[4][4] <-- adjust_matrix(matrix)
        print << "Adjusted matrix :\n--------------------------"
        for i<--0 to 3 do //display after subtract min column & min row
                print << "\n"
                for j<--0 to 3 do
                        print << matrix[i][j] << "\t"

        print << "\n--------------------------\n"
        print << "\n\nOptimal solution = " << drawLines(matrix, OriginalMatrix) << "\n"
}
```

4. Code

Source.cpp

Project8 (Global Scope)

```
#include<iostream>
using namespace std;

int adjust_matrix(int m[4][4]) {
    for (int i = 0; i <= 3; i++) {
        int minrow = m[0][0];
        for (int j = 0; j <= 3; j++) { //to get min row
            if (minrow > m[i][j])
                minrow = m[i][j];
        }
        for (int j = 0; j <= 3; j++) { //substract the min row
            m[i][j] -= minrow;
        }
    }

    for (int j = 0; j <= 3; j++) {
        int mincol = m[0][0];
        for (int i = 0; i <= 3; i++) { //get min column
            if (mincol > m[i][j])
                mincol = m[i][j];
        }
        for (int i = 0; i <= 3; i++) { //subtract min column
            m[i][j] -= mincol;
        }
    }
    return m[4][4]; //Adjusted matrix
}

int drawLines(int matrix[4][4], int OriginalMatrix[4][4]) {
    bool rowZeroline[4] = { false }; //if true it acts as a drawn line
```

130 % 0 3

```
int drawLines(int matrix[4][4], int OriginalMatrix[4][4]) {
    bool rowZeroline[4] = { false }; //if true it acts as a drawn line
    bool colZeroline[4] = { false }; //if true it acts as a drawn line
    int solMatrix[4]; //store selected job to each person

    //draw vertical line on zeros (if there is only 1 zero in the row)
    for (int i = 0; i <= 3; i++) {
        int ZerosNo = 0; //number of zeros in one row or column
        int colNo = 0; //column location at which vertical line could be drawn
        for (int j = 0; j <= 3; j++) {
            if (matrix[i][j] == 0 && colZeroline[j] == false) {
                ZerosNo++;
                colNo = j;
            }
            if (ZerosNo == 1 && colZeroline[colNo] == false) {
                colZeroline[colNo] = true; //drawn line
                solMatrix[i] = colNo; //store selected job to person number i
            }
        }
    }

    //draw horizontal line on zeros (if there is only 1 zero in the column)
    for (int j = 0; j <= 3; j++) {
        int ZerosNo = 0; //number of zeros in one row or column
        int rowNo = 0; //row location at which horizontal line could be drawn
        for (int i = 0; i <= 3; i++) {
            if (matrix[i][j] == 0 && colZeroline[j] == false) {
                ZerosNo++;
                rowNo = i;
            }
```

% 0 3

```
59          }
60          if (ZerosNo == 1 && rowZeroline[rowNo] == false) {
61              rowZeroline[rowNo] = true; //drawn line
62              solMatrix[rowNo] = j; //store selected job to person number i(rowNo)
63          }
64      }
65
66      //check that no zeros left
67      for (int i = 0; i <= 3; i++) {
68          for (int j = 0; j <= 3; j++) {
69              if (matrix[i][j] == 0 && colZeroline[j] == false && rowZeroline[i] == false) {
70                  rowZeroline[i] = true;
71                  solMatrix[i] = j; //store selected job to person number i
72              }
73          }
74      }
75
76      int NoLines = 0;
77      for (int m = 0; m <= 3; m++) {
78          if (rowZeroline[m] == true)
79              NoLines += 1;
80          if (colZeroline[m] == true)
81              NoLines += 1;
82      }
83      cout << "\nNumber of drawn lines : " << NoLines << endl;
84      if (NoLines < 4) {
85          for (int i = 0; i <= 3; i++) {
86              int minno = matrix[0][0]; // min no zero elemnt
87              for (int j = 0; j <= 3; j++) {
88                  if (rowZeroline[i] == false && colZeroline[j] == false) {
```



```
88                  if (rowZeroline[i] == false && colZeroline[j] == false) {
89                      if (minno > matrix[i][j] && matrix[i][j] > 0)
90                          minno = matrix[i][j];
91                  }
92              }
93              for (int j = 0; j <= 3; j++) { //substract the min number from unmarked numbers
94                  if (rowZeroline[i] == false && colZeroline[j] == false) {
95                      matrix[i][j] -= minno;
96                  }
97              }
98          }
99          drawLines(matrix, OriginalMatrix);
100     }
101
102     //display jobs assignment
103     for (int i = 0; i <= 3; i++) {
104         cout << "\nPerson " << i + 1 << " do job no. " << solMatrix[i] + 1;
105     }
106     int sum = 0; //optimal solution
107     for (int k = 0; k <= 3; k++) {
108         sum += OriginalMatrix[k][solMatrix[k]];
109     }
110     return sum;
111 }
112
113 int main() {
114     int matrix[4][4] = { 9,2,7,8,6,4,3,7,5,8,1,8,7,6,9,4 };
115     int OriginalMatrix[4][4] = { 9,2,7,8,6,4,3,7,5,8,1,8,7,6,9,4 };
116     /*
117     cout<<"enter the 4x4 matrix you want to solve : \n";
```

```
112
113 int main() {
114     int matrix[4][4] = { 9,2,7,8,6,4,3,7,5,8,1,8,7,6,9,4 };
115     int OriginalMatrix[4][4] = { 9,2,7,8,6,4,3,7,5,8,1,8,7,6,9,4 };
116     /*
117     cout<<"enter the 4x4 matrix you want to solve : \n";
118     for (int i = 0; i <= 3; i++) {
119         for (int j = 0; j <= 3; j++) {
120             cout << "Enter element for ("<<i<<","<<j<<"): ";
121             cin>> matrix[i][j];
122         }
123     }*/
124     matrix[4][4] = adjust_matrix(matrix);
125
126     cout << "Adjusted matrix :\n--------------------------";
127     for (int i = 0; i <= 3; i++) { //display after subtract min column & min row
128         cout << "\n";
129         for (int j = 0; j <= 3; j++) {
130             cout << matrix[i][j] << "\t";
131         }
132     }
133     cout << "\n--------------------------\n";
134     cout << "\n\nOptimal solution = " << drawLines(matrix, OriginalMatrix) << "\n";
135 }
136
```