МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

З дисципліни

«Дискретна математика»

Виконав:

Студент КН-113

Сайкевич В.А.

Перевірила:

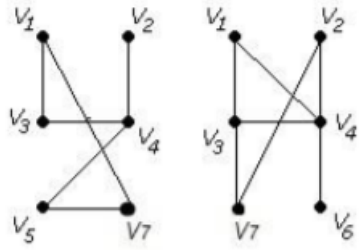
Мельникова Н.І.

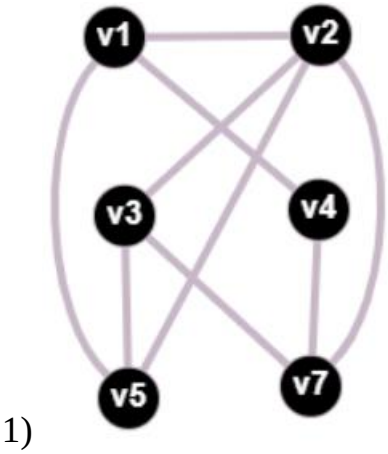
Львів – 2019

Варіант №17

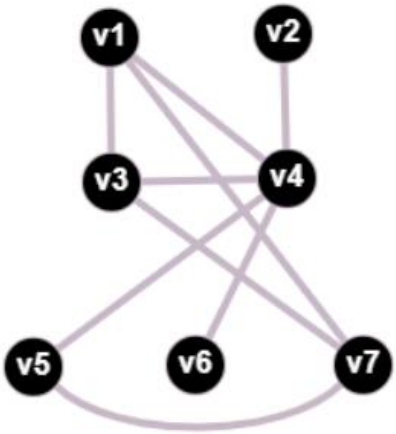
Завдання 1

Виконати наступні операції над графами: 1) знайти доповнення до першого графу; 2) об’єднання графів; 3) кільцеву сумму G1 та G2 (G1+G2); 4) розмножити вершину у другому графі; 5) виділити підграф А - що скадається з 3-х вершин в G1; 6) добуток графів.

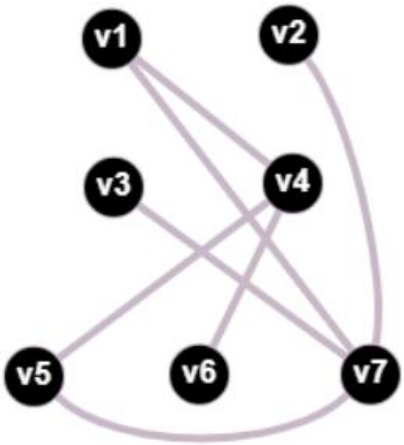




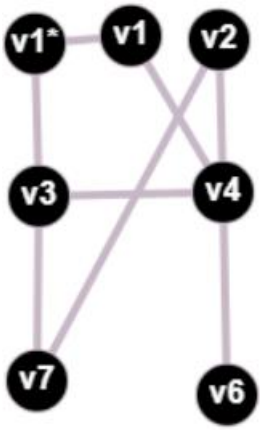
1)



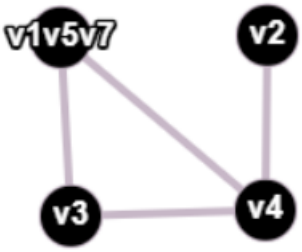
2)



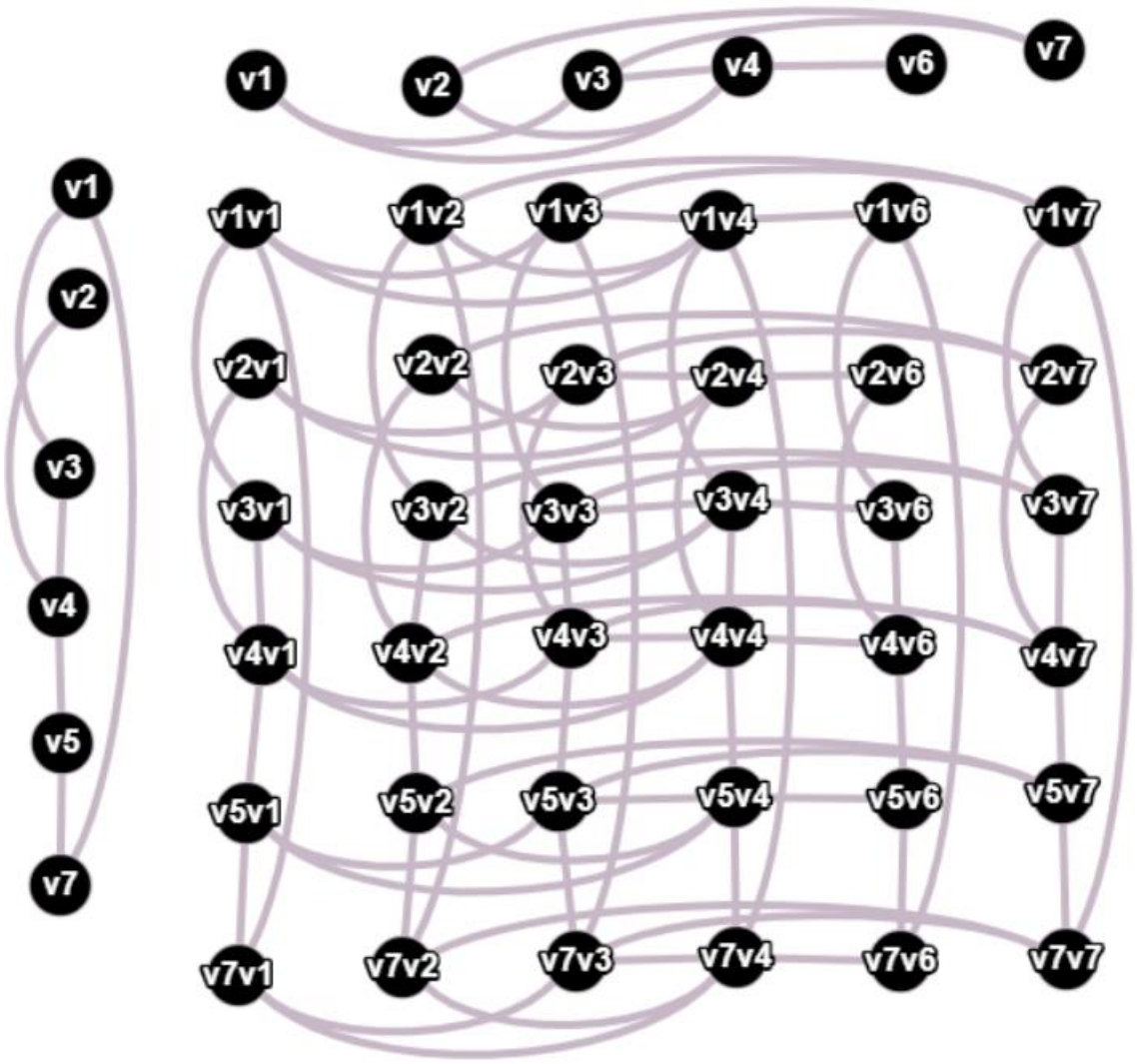
3)



4)



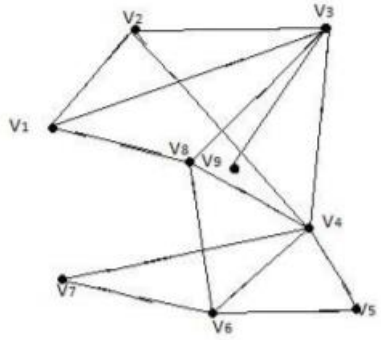
5)



6)

Завдання 2

Скласти таблицю суміжності для неорграфа.



	v1	v2	v3	v4	v5	v6	v7	v8	v9
v1	-	1	1	0	0	0	0	1	0
v2	1	-	1	1	0	0	0	0	0
v3	1	1	-	1	0	0	0	1	1
v4	0	1	1	-	1	1	1	1	0
v5	0	0	0	1	-	1	0	0	0
v6	0	0	0	1	1	-	1	1	0
v7	0	0	0	1	0	1	-	0	0
v8	1	0	1	1	0	1	0	-	0
v9	0	0	1	0	0	0	0	0	-

Завдання 3

Для графа з другого завдання знайти діаметр.

Діаметр графа=3.

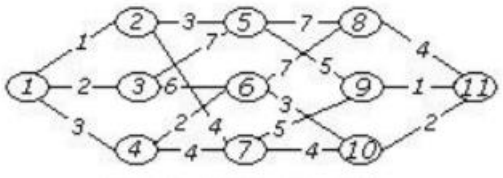
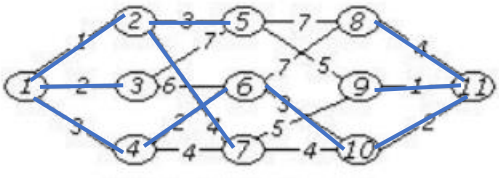
Завдання 4

Для графа з другого завдання виконати обхід дерева вглиб.

Елемент	Номер	Черга
1	1	1
8	2	18
6	3	186
5	4	1865
4	5	18654
7	6	186547
-		18654
2	7	186542
3	8	1865423
9	9	18654239
-		1865423
-		186542
-		18654
-		1865
-		186
-		18
-		1
-		-

Завдання 5

Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

Прима) з’єднуємо 1-2, 1-3, 1-4, 4-6, 2-5, 6-10, 10-11, 9-11, 2-7, 8-11.

Краскала) з’єднуємо 1-2, 9-11, 1-3, 4-6, 10-11, 1-4, 2-5, 6-10, 2-7, 8-11.

Завдання 6

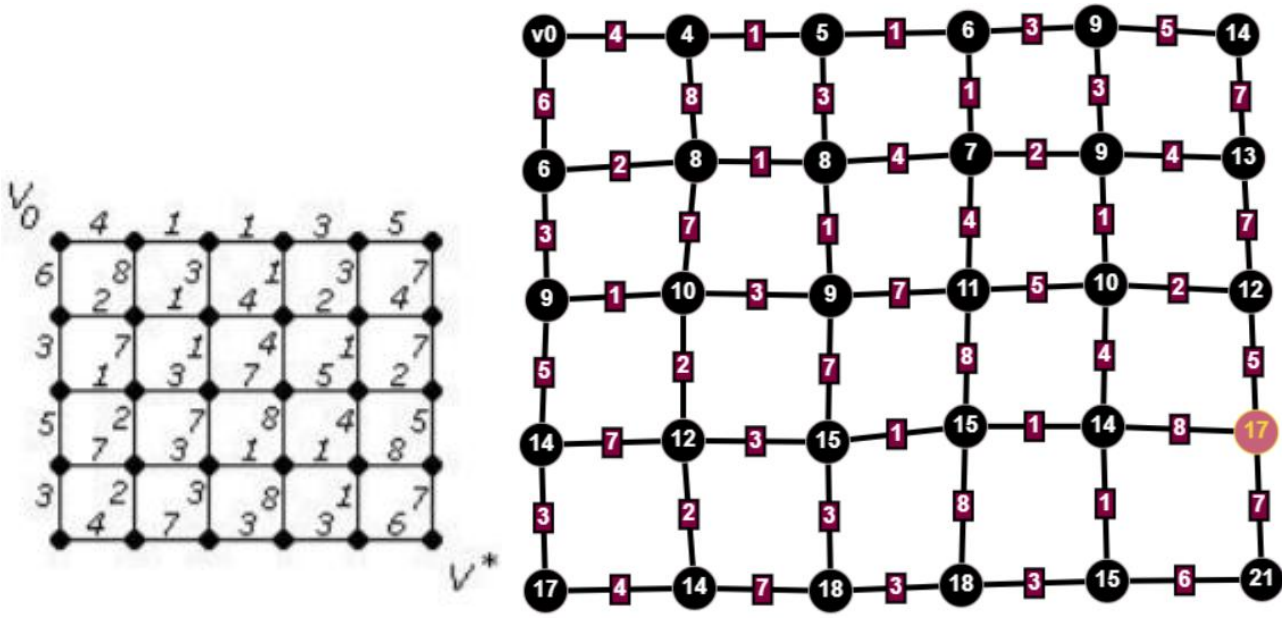
Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

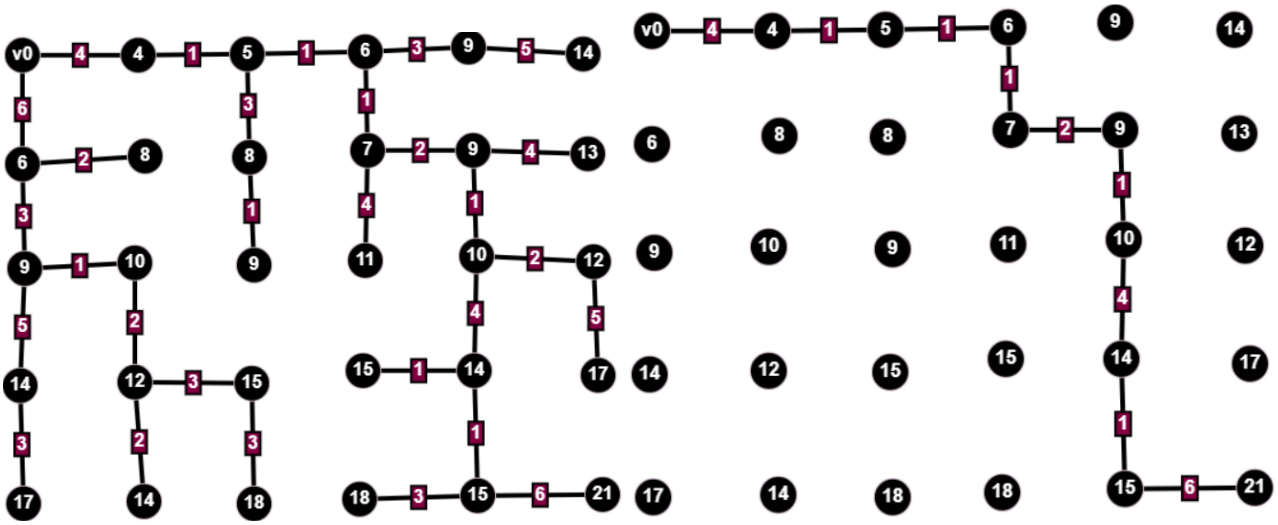
	1	2	3	4	5	6	7	8
1	∞	6	6	6	1	3	1	3
2	6	∞	5	5	1	6	1	5
3	6	5	∞	7	7	7	7	5
4	6	5	7	∞	6	5	1	2
5	1	1	7	6	∞	6	6	6
6	3	6	7	5	6	∞	1	2
7	1	1	7	1	6	1	∞	2
8	3	5	5	2	6	2	2	∞

Якщо йти з першої точки то буде 19

Завдання 7

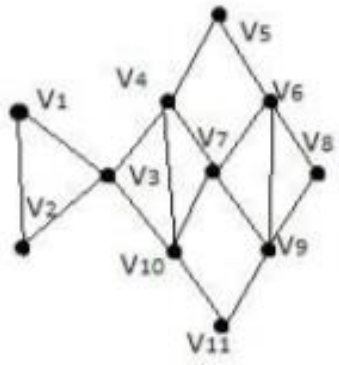
За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V*.





Завдання 8

Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.



А) 1,2,3,4,5,6,9,8,6,7,4,10,7,9,11,10,3,1.

Б) Виділяємо цикли 1,2,3,1; 3,4,5,6,8,9,11,10,3; 4,7,10,4; 6,7,9,6.

І виходить ейлеровий цикл 1,2,3,4,7,10,4,5,6,7,9,6,8,9,11,10,3,1.

Завдання 9

Спростити формулу (привести її до скороченої ДНФ).

$x\bar{y} \cup \bar{x}\bar{z} \cup yz = x\bar{y} \vee \bar{x}\bar{z} \vee yz = (x \wedge \bar{y}) \vee (\bar{x} \wedge \bar{z}) \vee (y \wedge z)$

$= (x \wedge \bar{y}) \vee ((\bar{x} \wedge \bar{z}) \vee y) \wedge ((\bar{x} \wedge \bar{z}) \vee z))$

$= (x \wedge \bar{y}) \vee ((\bar{x} \vee y) \wedge (\bar{z} \vee y)) \wedge ((\bar{x} \vee z) \wedge (\bar{z} \vee z))$

$= (x \wedge \bar{y}) \vee ((\bar{x} \vee y) \wedge (\bar{z} \vee y) \wedge (\bar{x} \vee z)) = (x \wedge \bar{y}) \vee \bar{x} \vee y$

$= ((x \vee \bar{x}) \wedge (\bar{y} \vee \bar{x})) \vee y = \bar{y} \vee \bar{x} \vee y = 1$

Код

1. Обхід графа вглиб

```cpp
#include <iostream>

using namespace std;

const int n = 9;
int i, j;
bool* visited = new bool[n];

int graph[n][n] =
{
{0,1,1,0,0,0,0,1,0},
{1,0,1,1,0,0,0,0,0},
{1,1,0,1,0,0,0,1,1},
{0,1,1,0,1,1,1,1,0},
{0,0,0,1,0,1,0,0,0},
{0,0,0,1,1,0,1,1,0},
{0,0,0,1,0,1,0,0,0},
{1,0,1,1,0,1,0,0,0},
{0,0,1,0,0,0,0,0,0}
};

void DFS(int st)
{
    int r;
    cout << st + 1 << " ";
    visited[st] = true;
    for (r = 0; r <= n; r++)
        if ((graph[st][r] != 0) && (!visited[r]))
            DFS(r);
}
```

```cpp
void main()
{
    int start;
    cout << "Adjecency matrix: " << endl;
    for (i = 0; i < n; i++)
    {
        visited[i] = false;
        for (j = 0; j < n; j++)
            cout << " " << graph[i][j];
        cout << endl;
    }
    cout << "Starting vertex "; cin >> start;
    bool* vis = new bool[n];
    cout << "Way: ";
    DFS(start - 1);
    delete[]visited;
    system("pause>>void");
}
```

Результат:

```
Adjecency matrix:
 0 1 1 0 0 0 0 1 0
 1 0 1 1 0 0 0 0 0
 1 1 0 1 0 0 0 1 1
 0 1 1 0 1 1 1 1 0
 0 0 0 1 0 1 0 0 0
 0 0 0 1 1 0 1 1 0
 0 0 0 1 0 1 0 0 0
 1 0 1 1 0 1 0 0 0
 0 0 1 0 0 0 0 0 0
Starting vertex 1
Way: 1 2 3 4 5 6 7 8 9
```

2. Прима знаходження найменшого остову

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>

using namespace std;

void output(int size, int** adjrcencyarr) {
    cout << "  ";
    for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; }
    for (int i = 0; i < size; i++) {
        cout << endl << setw(3) << i + 1;
        for (int j = 0; j < i; j++) {
            cout << setw(3) << adjrcencyarr[j][i];
        }
        cout << "  -";
        for (int j = i + 1; j < size; j++) {
            cout << setw(3) << adjrcencyarr[i][j];
        }
    }
}

void main()
{
    int size;
    cout << "Enter amount of vertices ";
    cin >> size;
    int** adjrcencyarr = new int* [size];
    bool* vertex = new bool[size];
    cout << "Enter adjacency matrix \n";
    cout << "  ";
```

```
    for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; adjrcencyarr[i] = new int[size]; vertex[i] = 0; }
    cout << endl;
    for (int i = 0; i < size; i++) {
        cout << setw(3) << i + 1;
        for (int j = 0; j < i; j++) { cout << setw(3) << adjrcencyarr[j][i]; }
        cout << "  -";
        for (int j = i + 1; j < size; j++) { cin >> adjrcencyarr[i][j]; }
    }
    cout << endl;
    vertex[0] = 1;
    int min, minI, minJ;
    for (int n = 0; n < size - 1; n++) {
        min = 100, minI = 0, minJ = 0;
        for (int i = 0; i < size; i++) {
            for (int j = i + 1; j < size; j++) {
                if (adjrcencyarr[i][j] < min&& adjrcencyarr[i][j]!=0 && (vertex[i] && !vertex[j])) {
                    min = adjrcencyarr[i][j];
                    minI = i, minJ = j;
                }
            }
        }
        vertex[minI] = 1, vertex[minJ] = 1;
        cout << n + 1 << ")conected " << minI+1 << "-" << minJ+1 << endl;
    }
}
```

Результати :

```
Enter amount of vertices 11
Enter adjacency matrix
    1  2  3  4  5  6  7  8  9 10 11
 1  -  1  2  4  0  0  0  0  0  0  0
 2  1  -  0  0  3  0  2  0  0  0  0
 3  2  0  -  0  7  6  0  0  0  0  0
 4  4  0  0  -  0  2  3  0  0  0  0
 5  0  3  7  0  -  0  0  7  5  0  0
 6  0  0  6  2  0  -  0  7  0  3  0
 7  0  2  0  3  0  0  -  0  5  4  0
 8  0  0  0  0  0  7  7  0  -  0  0  4
 9  0  0  0  0  0  5  0  5  0  -  0  1
 10  0  0  0  0  0  0  3  4  0  0  - 4
 11  0  0  0  0  0  0  0  0  4  1  4  -
1)conected 1-2
2)conected 1-3
3)conected 2-7
4)conected 2-5
5)conected 1-4
6)conected 4-6
7)conected 6-10
8)conected 10-11
9)conected 5-9
10)conected 5-8
```

3. Краскала знаходження найменшого остову

```
#include <iostream>
#include <iomanip>

 using namespace std;

int find(int i,int* vertex)
{
    while (vertex[i] != i)
        i = vertex[i];
    return i;
}

void output(int size, int** adjacencyarr) {
    cout << "  ";
    for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; }
    for (int i = 0; i < size; i++) {
        cout << endl << setw(3) << i + 1;
        for (int j = 0; j < i; j++) {
            cout << setw(3) << adjacencyarr[j][i];
        }
        cout << "  -";
        for (int j = i + 1; j < size; j++) {
            cout << setw(3) << adjacencyarr[i][j];
        }
    }
}

void main()
{
    int size;
```

```
    cout << "Enter amount of vertices ";
    cin >> size;
    int** adjacencyarr = new int* [size];
    int* vertex = new int[size];
    cout << "Enter adjacency matrix \n";
    cout << "  ";
    for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; adjacencyarr[i] = new int[size]; vertex[i] = i; }
    cout << endl;
    for (int i = 0; i < size; i++) {
        cout << setw(2) << i + 1;
        for (int j = 0; j < i; j++) { cout << setw(3) << adjacencyarr[j][i]; }
        cout << "  -";
        for (int j = i + 1; j < size; j++) { cin >> adjacencyarr[i][j]; }
    }
    cout << endl;
    int min, minI, minJ;
    for (int n = 0; n < size - 1; n++) {
        min = 100, minI = 0, minJ = 0;
        for (int i = 0; i < size; i++) {
            for (int j = i + 1; j < size; j++) {
                if (adjacencyarr[i][j] < min&& adjacencyarr[i][j]!=0 && (find(i, vertex) != find(j, vertex))) {
                    min = adjacencyarr[i][j];
                    minI = i, minJ = j;
                }
            }
        }
        vertex[find(minI, vertex)] = find(minJ, vertex);
        cout << n + 1 << ")conected " << minI+1 << "-" << minJ+1 << endl;
    }
}
```

Результати :

```
Enter amount of vertices 11
Enter adjacency matrix
    1  2  3  4  5  6  7  8  9 10 11
 1  -  1  2  4  0  0  0  0  0  0  0
 2  1  -  0  0  3  0  2  0  0  0  0
 3  2  0  -  0  7  6  0  0  0  0  0
 4  4  0  0  -  0  2  3  0  0  0  0
 5  0  3  7  0  -  0  0  7  5  0  0
 6  0  0  6  2  0  -  0  7  0  3  0
 7  0  2  0  3  0  0  -  0  5  4  0
 8  0  0  0  0  7  7  0  -  0  0  4
 9  0  0  0  0  5  0  5  0  -  0  1
10  0  0  0  0  0  0  3  4  0  0  -  4
11  0  0  0  0  0  0  0  0  4  1  4  -
1)conected 9-11
2)conected 1-2
3)conected 4-6
4)conected 2-7
5)conected 1-3
6)conected 6-10
7)conected 4-7
8)conected 2-5
9)conected 10-11
10)conected 8-11
```

4. Дейкстра знаходження найкоротшого ланцюга між парою вершин

```
#include <iostream>
#include <iomanip>

 using namespace std;

struct vertex {
    int x=INT_MAX, left=0, top=0, right=0, bot=0;
    bool used = 0;
};

void main()
{
    vertex arr[5][6];
    arr[0][0].x = 0;
    cout << "Enter lenghts \n";
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 5; j++) {
            cout << i + 1 << j + 1 << "-" << i + 1 << j + 2 << " "; cin >> arr[i][j].right; arr[i][j + 1].left = arr[i][j].right;
            cout << i + 1 << j + 1 << "-" << i + 2 << j + 1 << " "; cin >> arr[i][j].bot; arr[i + 1][j].top = arr[i][j].bot;
        }
        cout << i + 1 << "6" << "-" << i + 2 << "6" << " "; cin >> arr[i][5].bot; arr[i + 1][5].top = arr[i][5].bot;
    }
    for (int j = 0; j < 5; j++) {
        cout << "5" << j + 1 << "-" << "5" << j + 2 << " "; cin >> arr[4][j].right; arr[4][j + 1].left = arr[4][j].right;
    }
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 5; j++) cout << "#-" << setw(2) << arr[i][j].right << "-";
        cout << "#\n|    |    |    |    |    |\n";
        for (int j = 0; j < 6; j++) cout << setw(2) << arr[i][j].bot << "   ";
        cout << "\n|    |    |    |    |    |\n";
```

```
    }
    for (int j = 0; j < 5; j++) cout << "#-" << setw(2) << arr[4][j].right << "-";
    cout << "#\n\n";

    int min = INT_MAX, mini, minj;
    for (int n = 0; n < 30; n++, min = INT_MAX) {
        for (int i = 0; i < 5; i++) for (int j = 0; j < 6; j++) if (arr[i][j].x < min && !arr[i][j].used) { min = arr[i][j].x; mini = i; minj = j; }

        arr[mini][minj].used = 1;
        if (minj < 5) if (arr[mini][minj + 1].x>arr[mini][minj].x + arr[mini][minj].right) arr[mini][minj + 1].x = arr[mini][minj].x + arr[mini][minj].right;
        if (minj > 0) if (arr[mini][minj - 1].x > arr[mini][minj].x + arr[mini][minj].left) arr[mini][minj - 1].x = arr[mini][minj].x + arr[mini][minj].left;
        if (mini < 4) if (arr[mini + 1][minj].x > arr[mini][minj].x + arr[mini][minj].bot) arr[mini + 1][minj].x = arr[mini][minj].x + arr[mini][minj].bot;
        if (mini > 0) if (arr[mini - 1][minj].x > arr[mini][minj].x + arr[mini][minj].top) arr[mini - 1][minj].x = arr[mini][minj].x + arr[mini][minj].top;
    }
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 5; j++) cout << setw(2) << arr[i][j].x << "-" << setw(2) << arr[i][j].right << "-";
        cout << setw(2) << arr[i][5].x << "\n |     |     |     |     |     |\n";
        for (int j = 0; j < 6; j++) cout << setw(2) << arr[i][j].bot << "    ";
        cout << "\n |     |     |     |     |     |\n";
    }
    for (int j = 0; j < 5; j++) cout << setw(2) << arr[4][j].x << "-" << setw(2) << arr[4][j].right << "-";
    cout << setw(2) << arr[4][5].x << "\n";
}
```

Результати :

```
Enter lenghts
11-12 1
11-21 4
12-13 1
12-22 8
13-14 6
13-23 3
14-15 3
14-24 1
15-16 3
15-25 5
16-26 7
21-22 2
21-31 1
22-23 1
22-32 7
23-24 4
23-33 1
24-25 2
24-34 4
25-26 4
25-35 3
26-36 7
31-32 1
31-41 5
32-33 3
32-42 4
33-34 2
33-43 7
34-35 5
34-44 8
35-36 7
35-45 2
36-46 5
41-42 7
41-51 3
42-43 3
42-52 2
43-44 1
43-53 1
44-45 1
44-54 3
45-46 8
45-55 8
46-56 7
51-52 4
52-53 7
53-54 3
54-55 3
55-56 6
#- 1-#- 1-#- 6-#- 3-#- 3-#
 |     |     |     |     |     |
 4     8     3     1     5     7
 |     |     |     |     |     |
#- 2-#- 1-#- 4-#- 2-#- 4-#
 |     |     |     |     |     |
 1     7     1     4     3     7
 |     |     |     |     |     |
#- 1-#- 3-#- 2-#- 5-#- 7-#
 |     |     |     |     |     |
 5     4     7     8     2     5
 |     |     |     |     |     |
#- 7-#- 3-#- 1-#- 1-#- 8-#
 |     |     |     |     |     |
 3     2     1     3     8     7
 |     |     |     |     |     |
#- 4-#- 7-#- 3-#- 3-#- 6-#
```

```
 0- 1- 1- 1- 2- 6- 8- 3-11- 3-14
 |     |     |     |     |     |
 4     8     3     1     5     7
 |     |     |     |     |     |
 4- 2- 6- 1- 5- 4- 9- 2-11- 4-15
 |     |     |     |     |     |
 1     7     1     4     3     7
 |     |     |     |     |     |
 5- 1- 6- 3- 6- 2- 8- 5-13- 7-20
 |     |     |     |     |     |
 5     4     7     8     2     5
 |     |     |     |     |     |
10- 7-10- 3-13- 1-14- 1-15- 8-23
 |     |     |     |     |     |
 3     2     1     3     8     7
 |     |     |     |     |     |
13- 4-12- 7-14- 3-17- 3-20- 6-26
```

5. «Іди в найближчий» для розв’язання задачі комівояжера

```cpp
#include <iostream>
#include <stdio.h>
#define inf 1E9
#define NMAX 16
#define min(x, y) x < y ? x : y

using namespace std;

int n, i, j, k, m, temp, ans, d[NMAX][NMAX], t[1 << NMAX][NMAX];
int get(int nmb, int x)
{
    return (x & (1 << nmb)) != 0;
}
int main(void)
{
    cin>>n;
    for (i = 0; i < n; ++i)
        for (j = 0; j < n; ++j) cin>>d[i][j];
    t[1][0] = 0; m = 1 << n;
    for (i = 1; i < m; i += 2)
        for (j = (i == 1) ? 1 : 0; j < n; ++j)
        {
            t[i][j] = inf;
            if (j > 0 && get(j, i))
            {
                temp = i ^ (1 << j);
                for (k = 0; k < n; ++k)
                    if (get(k, i) && d[k][j] > 0) t[i][j] = min(t[i][j], t[temp][k] + d[k][j]);
            }
        }
    for (j = 1, ans = inf; j < n; ++j)
        if (d[j][0] > 0) ans = min(ans, t[m - 1][j] + d[j][0]);
    if (ans == inf) cout<<"-1"; else cout<<ans;
    return 0;
}
```

Результат:

```
8
0 6 6 6 1 3 1 3
6 0 5 5 1 6 1 5
6 5 0 7 7 7 7 5
6 5 7 0 6 5 1 2
1 1 7 6 0 6 6 6
3 6 7 5 6 0 1 2
1 1 7 1 6 1 0 2
3 5 5 2 6 2 2 0
19
```

6. Флері та елементарних циклів знаходження ейлерового ланцюга в ейлеровому графі.

```cpp
#include <iostream>
#include <vector>
#include <stack>
#include <algorithm>
#include <list>

using namespace std;

vector < list<int> > graph;
vector <int> deg;
stack<int> head, tail;
int main() {
    int n, a, x, y;
    cin >> n >> a;
    graph.resize(n + 1);
    deg.resize(n + 1);
    for (; a--;) {
        cin >> x >> y;
        graph[x].push_back(y);
        graph[y].push_back(x);
        ++deg[x];
        ++deg[y];
    }
    if (any_of(deg.begin() + 1, deg.end(), [](int i) {return i & 1; }))
        cout << "No euler cycle exists\n";
    else {
        head.push(1);
        while (!head.empty()) {
            while (deg[head.top()]) {
                int v = graph[head.top()].back();
                graph[head.top()].pop_back();
                graph[v].remove(head.top());
                --deg[head.top()];
                head.push(v);
                --deg[v];
            }
            while (!head.empty() && !deg[head.top()]) { tail.push(head.top()); head.pop(); }
        }
        while (!tail.empty()) { cout << tail.top() << ' '; tail.pop(); }
    }
}
```

Результат:

```
11 17
1 2
1 3
2 3
3 4
3 10
4 10
4 5
4 7
10 7
10 11
5 6
7 6
7 9
11 9
6 9
6 8
9 8
1 3 10 11 9 8 6 9 7 6 5 4 7 10 4 3 2 1
```