МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-115

Кагуй Андрій

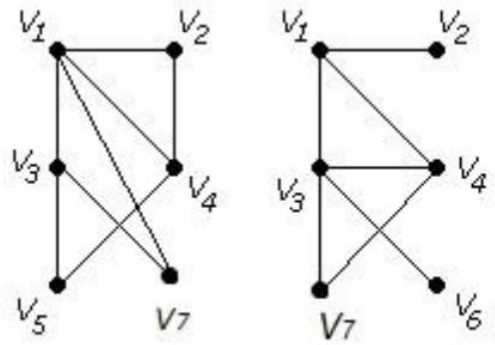
Викладач:

Мельникова Н.І.

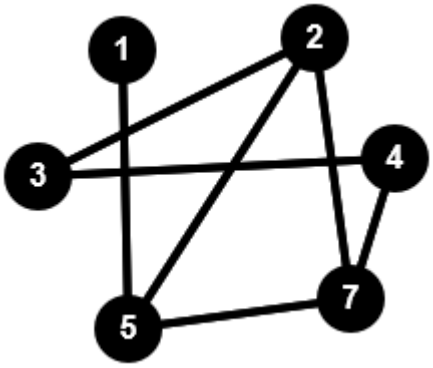
Львів-2019

Варіант 9

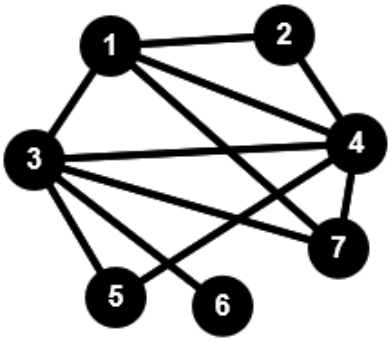
Завдання № 1
Виконати наступні операції над графами:
9)



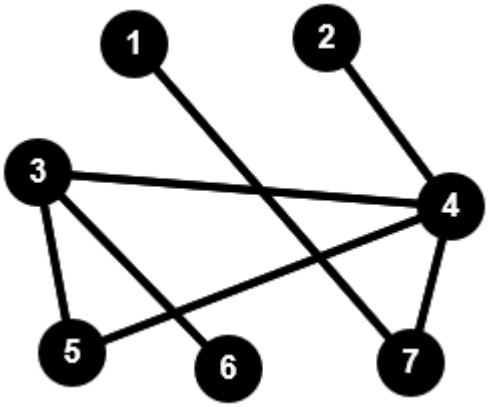
1) знайти доповнення до першого графу,



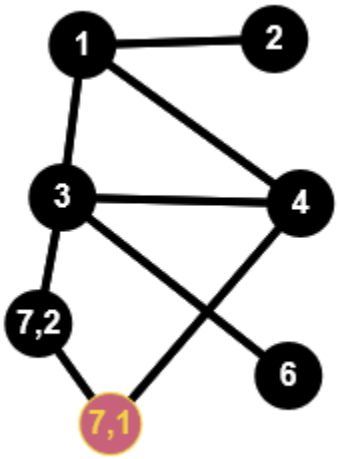
2) об’єднання графів,



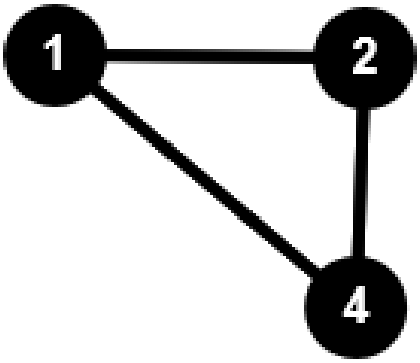
3) кільцеву суму G1 та G2 (G1+G2)



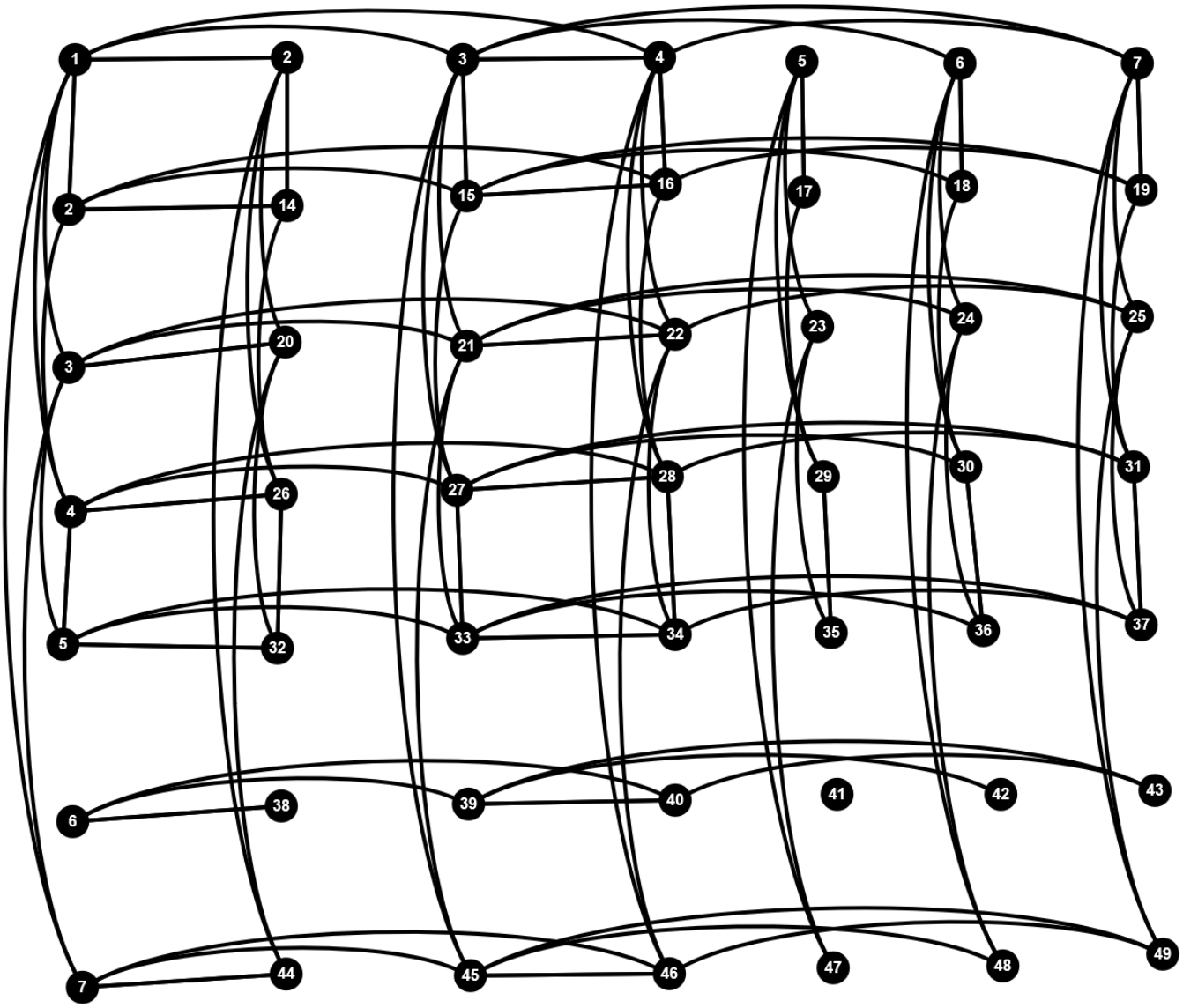
4) розмножити вершину у другому графі,



5) виділити підграф А - що скадається з 3-х вершин в G1

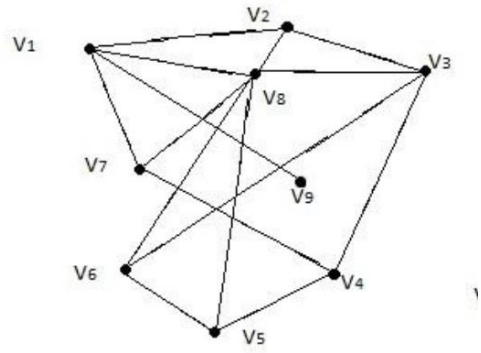


6) добуток графів


1
2
3
4
5
6
7
2
14
15
16
17
18
19
3
20
21
22
23
24
25
4
26
27
28
29
30
31
5
32
33
34
35
36
37
6
38
39
40
41
42
43
7
44
45
46
47
48
49

Завдання № 2
Скласти таблицю суміжності для орграфа.

9)



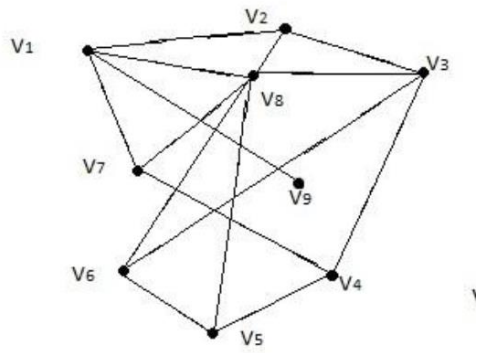
	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
V2	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**
V3	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
V4	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**
V5	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
V6	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**
V7	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
V8	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
V9	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Завдання № 3
Для графа з другого завдання знайти діаметр.

Діаметр = 3

Завдання № 4
Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).

9)



Обхід дерева вглиб:

Вершина	**Стек**
V1	**V1**
V2	**V1V2**
V3	**V1V2V3**
V4	**V1V2V3V4**
V5	**V1V2V3V4V5**
V6	**V1V2V3V4V5V6**
V8	**V1V2V3V4V5V6V8**
V7	**V1V2V3V4V5V6V8V7**
-	**V1V2V3V4V5V6V8**
-	**V1V2V3V4V5V6**
-	**V1V2V3V4V5**
-	**V1V2V3V4**
-	**V1V2V3**
-	**V1V2**
-	**V1**
V9	**V1V9**
-	**V1**
-	**Ø**

Програмна реалізація:

```cpp
#include <iostream>
using namespace std;
const int n = 9;
int i, j;
bool* visited = new bool[n];
int graph[n][n] =
{
{0,1,0,0,0,0,1,1,1},
{1,0,1,0,0,0,0,1,0},
{0,1,0,1,0,1,0,1,0},
{0,0,1,0,1,0,1,0,0},
{0,0,0,1,0,1,0,1,0},
{0,0,1,0,1,0,0,1,0},
{1,0,0,1,0,0,0,1,0},
{1,1,1,0,1,1,1,0,0},
{1,0,0,0,0,0,0,0,0}
};
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);
}
void main()
{
    int start;
    cout << "Matrix: " << endl;
    for (i = 0; i < n; i++)
    {
        visited[i] = false;
        for (j = 0; j < n; j++)
            cout << " " << graph[i][j];
        cout << endl;
    }
    cout << "Start edge >> "; cin >> start;
    bool* vis = new bool[n];
    cout << "Path: ";
    DFS(start - 1);
    delete[]visited;
}
```

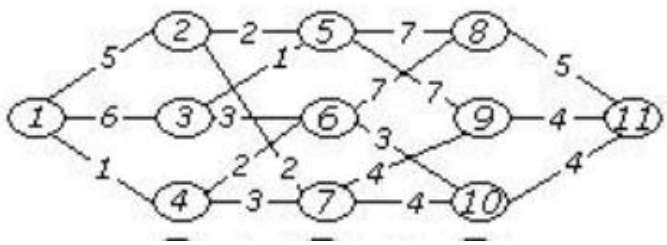
Апробація програми:

Microsoft Visual Studio Debug

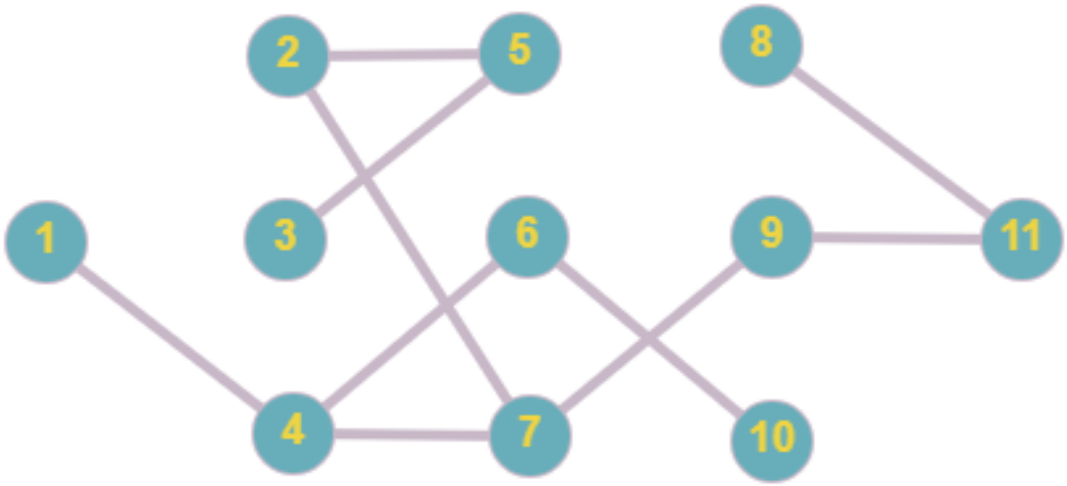
```
Matrix:
 0 1 0 0 0 0 1 1 1
 1 0 1 0 0 0 0 1 0
 0 1 0 1 0 1 0 1 0
 0 0 1 0 1 0 1 0 0
 0 0 0 1 0 1 0 1 0
 0 0 1 0 1 0 0 1 0
 1 0 0 1 0 0 0 1 0
 1 1 1 0 1 1 1 0 0
 1 0 0 0 0 0 0 0 0
Start edge >> 1
Path: 1 2 3 4 5 6 8 7 9
```

Завдання № 5
Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

9)



Прима:



V={1,4,6,10,7,2,5,3,9,11,8}

E={(1,4),(4,6), (6,10),((4,7),(7,2),(2,5),(5,3),(7,9),(9,11),(11,8)}

Програмна реалізація:

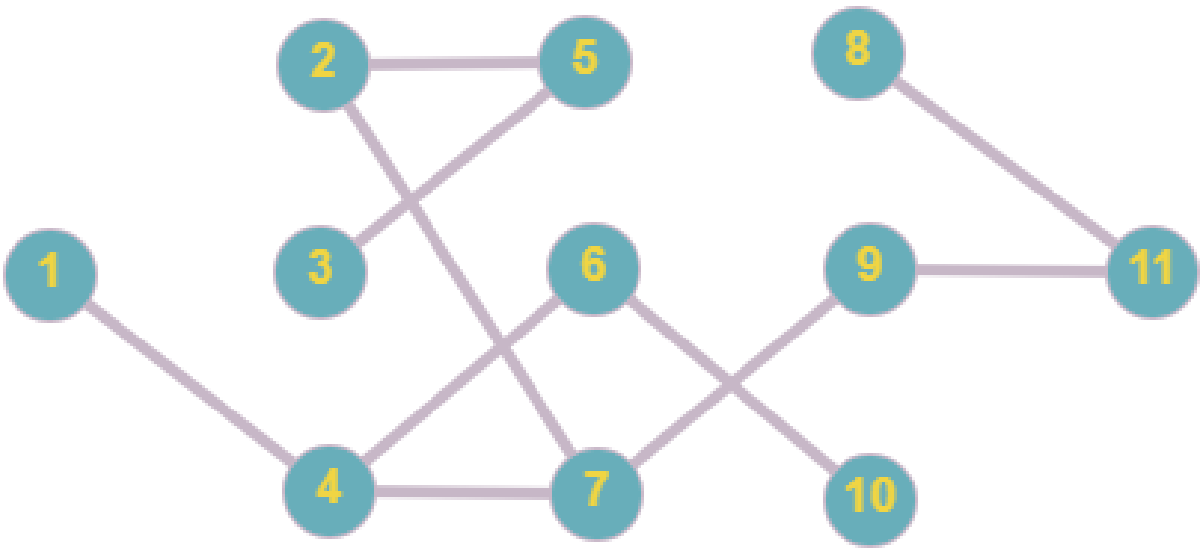
```cpp
using namespace std;
#include <stdio.h>
#include <iostream>

#define V 11

int graph[V][V] = {
    {0, 5, 6, 1, 0, 0, 0, 0, 0, 0, 0},
    {5, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0},
    {6, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0},
    {1, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0},
    {0, 2, 1, 0, 0, 0, 0, 7, 7, 0, 0},
    {0, 0, 3, 2, 0, 0, 0, 7, 0, 3, 0},
    {0, 2, 0, 3, 0, 0, 0, 0, 4, 4, 0},
    {0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 5},
    {0, 0, 0, 0, 7, 0, 4, 0, 0, 0, 4},
    {0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 4},
    {0, 0, 0, 0, 0, 0, 0, 5, 4, 4, 0}
};

int main() {

    int no_edge;
    int selected[V];
    memset(selected, false, sizeof(selected));

    no_edge = 0;

    selected[0] = true;

    int x;
    int y;

    cout << "Edge" << " : " << "Weight";
    cout << endl;
    while (no_edge < V - 1) {


        int min = INT_MAX;
        x = 0;
        y = 0;

        for (int i = 0; i < V; i++) {
            if (selected[i]) {
                for (int j = 0; j < V; j++) {
                    if (!selected[j] && graph[i][j]) {
                        if (min > graph[i][j]) {
                            min = graph[i][j];
                            x = i;
                            y = j;
                        }

                    }
                }
            }
        }
        cout << x + 1 << " - " << y + 1 << " : " << graph[x][y];
        cout << endl;
        selected[y] = true;
        no_edge++;
    }

    return 0;
}
```

Апробація програми :

Microsoft Visual S

```
Edge : Weight
1 - 4 : 1
4 - 6 : 2
4 - 7 : 3
7 - 2 : 2
2 - 5 : 2
5 - 3 : 1
6 - 10 : 3
7 - 9 : 4
9 - 11 : 4
11 - 8 : 5
```

Краскала:

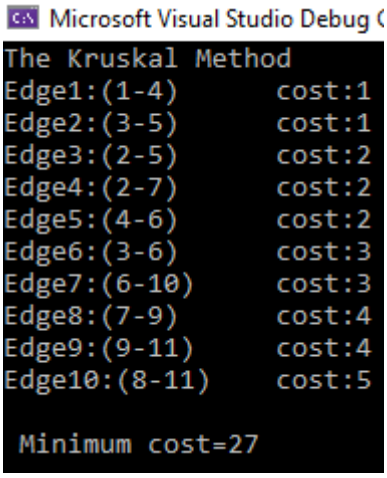


V={1,4,3,5,6,2,7,10,9,11,8}
E={(1,4),(3,5),(2,5),(2,7),(4,6),(3,6),(6,10),(7,9),(9,11),(11,8)}

Програмна реалізація:

```cpp
#include <iostream>
#define INT_MAX 2147483647

using namespace std;

#define V 11
int parent[V];

int find(int i) {
    while (parent[i] != i)
        i = parent[i];
    return i;
}
void union1(int i, int j) {
    int a = find(i);
    int b = find(j);
    parent[a] = b;
}
void Kruskal(int cost[][V]) {
    cout << "The Kruskal Method" << endl;
    int min_cost = 0;

    for (int i = 0; i < V; i++)
        parent[i] = i;

    int edge_count = 0;
    while (edge_count < V - 1) {
        int min = INT_MAX, a = -1, b = -1;
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                if (find(i) != find(j) && cost[i][j] < min) {
                    min = cost[i][j];
                    a = i;
                    b = j;
                }
            }
        }
        if (min == INT_MAX) {
            cout << "There is no minimum spanning tree." << endl;
            exit(0);
        }
        union1(a, b);
        cout << "Edge" << edge_count++ + 1 <<":" << "(" << a + 1 << '-' << b + 1 << ")" << " \t cost:" << min << endl;
        min_cost += min;
    }
    cout << "\n Minimum cost=" <<  min_cost << endl;
}
int main() {
    int cost[][V] = {
            {INT_MAX, 5,       6,       1,       INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX},//1
            {5,       INT_MAX, INT_MAX, INT_MAX, 2,       INT_MAX, 2,       INT_MAX, INT_MAX, INT_MAX, INT_MAX},//2
            {6,       INT_MAX, INT_MAX, INT_MAX, 1,       3,       INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX},//3
            {1,       INT_MAX, INT_MAX, INT_MAX, INT_MAX, 2,       3,       INT_MAX, INT_MAX, INT_MAX, INT_MAX},//4
            {INT_MAX, 2,       1,       INT_MAX, INT_MAX, INT_MAX, INT_MAX, 7,       7,       INT_MAX, INT_MAX},//5
            {INT_MAX, INT_MAX, 3,       2,       INT_MAX, INT_MAX, INT_MAX, 7,       INT_MAX, 3,       INT_MAX},//6
            {INT_MAX, 2,       INT_MAX, 3,       INT_MAX, INT_MAX, INT_MAX, INT_MAX, 4,       4,       INT_MAX},//7
            {INT_MAX, INT_MAX, INT_MAX, INT_MAX, 7,       7,       INT_MAX, INT_MAX, INT_MAX, INT_MAX, 5},//8
            {INT_MAX, INT_MAX, INT_MAX, INT_MAX, 7,       INT_MAX, 4,       INT_MAX, INT_MAX, INT_MAX, 4},//9
            {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 3,       4,       INT_MAX, INT_MAX, INT_MAX, 4},//10
            {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 5,       4,       4,       INT_MAX},//11
    };

    Kruskal(cost);
    return 0;
}
```

Апробація програми:



Завдання № 6

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

9)

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

Вершина	Маршрут	Вага
-	1	0
8	1-8	1
5	1-8-7	3
4	1-8-7-4	4
7	1-8-7-4-5	5
2	1-8-7-4-5-2	7
6	1-8-7-4-5-2-6	8
3	1-8-7-4-5-2-6-3	14

1	1-8-7-4-5-2-6-3-1	19

Програмна реалізація:

```cpp
#include<iostream>
#include<vector>

using namespace std;
#define V 8

int counter = 0, Miminal_way = INT_MAX;

bool check(vector<int> v, int Node) {
    for (auto i = v.begin(); i != v.end(); i++)
        if (*i == Node)
            return false;
    return true;
}

int minimum(vector<int>* v, int graph[][8], int i) {
    int min = INT_MAX;
    for (int j = 0; j < V; j++) {
        if (graph[i][j] < min && graph[i][j] != 0 && check((*v), j))min = graph[i][j];
    }
    return min;
}
void search(vector<int>* v, int graph[][8], int pos, vector<int>* res) {
    int min, i = pos;
    min = minimum(v, graph, i);
    for (int j = 0; j < V; j++)
        if (graph[i][j] == min && check((*v), j)) {
            (*v).push_back(j);
            search(v, graph, j, res);
        }
    if (v->size() == V) {
        (*v).push_back((*v)[0]);
        counter = 0;
        for (int l = 1; l <= V; l++)
            counter += graph[(*v)[l - 1]][(*v)[l]];
        if (Miminal_way == counter) {
            for (int op = 0; op <= V; op++)
                (*res).push_back((*v)[op]);
            (*res).push_back(counter);
        }
        else if (Miminal_way > counter) {
            (*res).clear();
            for (int op = 0; op <= V; op++)
                (*res).push_back((*v)[op]);
            (*res).push_back(counter);
            Miminal_way = counter;
        }
        v->pop_back();
    }
    v->pop_back();
}
int main()
{
    vector<int> v;
    vector<int> result;

    int graph[][V] = {
            {0, 5, 5, 3, 3, 4, 4, 1},
            {5, 0, 4, 3, 2, 1, 4, 6},
            {5, 4, 0, 4, 5, 6, 5, 5},
            {3, 3, 4, 0, 1, 5, 1, 7},
            {3, 2, 5, 1, 0, 5, 5, 2},
            {4, 1, 6, 5, 5, 0, 7, 3},
            {4, 4, 5, 1, 5, 7, 0, 2},
            {1, 6, 5, 7, 2, 3, 2, 0},
    };

    for (int i = 0; i < V; i++) {
        v.clear();
        v.push_back(i);
        search(&v, graph, i, &result);
    }
    cout << "The shortest way is:  " << endl;
    for (int i = 1; i <= 10; i++) {
        if (i != 0 && i % (V + 2) == 0)
            cout << "\n" << "The weight of path is: " <<result[i - 1] << endl;
        else
            cout << result[i - 1] + 1 << " ";
    }
}
```

Апробація програми:

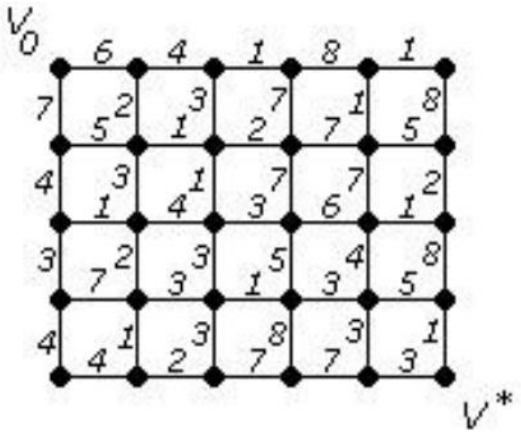
Microsoft Visual Studio Debug Console

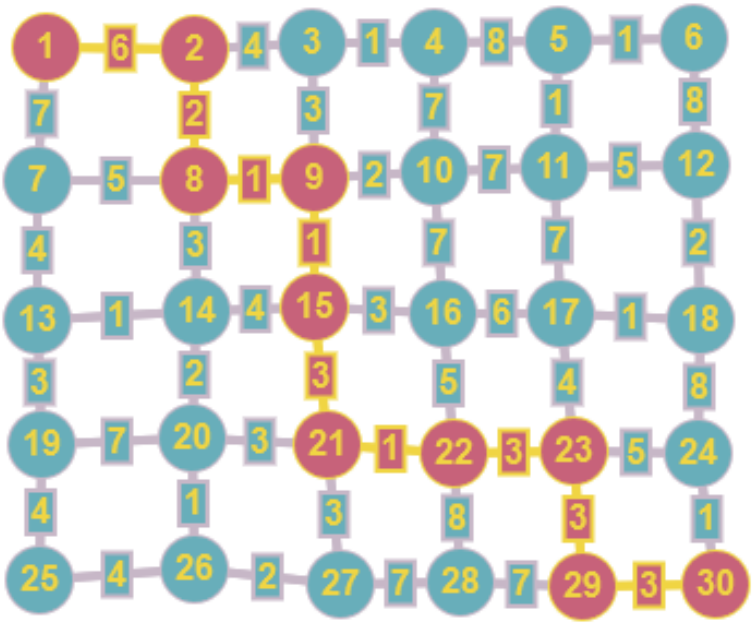
```
The shortest way is:
1 8 7 4 5 2 6 3 1
The weight of path is: 19
```

Завдання № 7

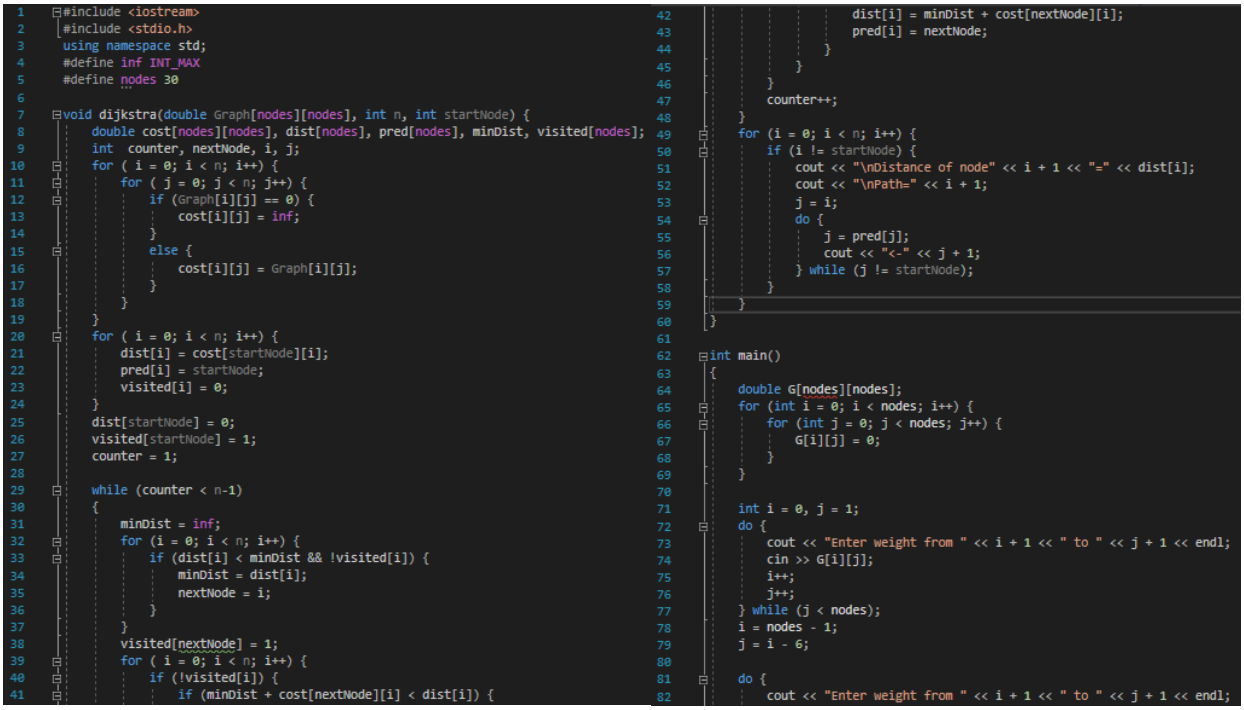
За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин *V*0 і *V** .

9)

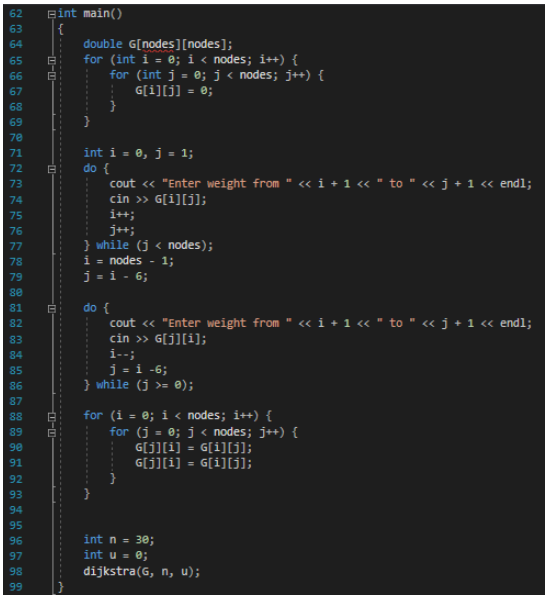




Програмна реалізація:

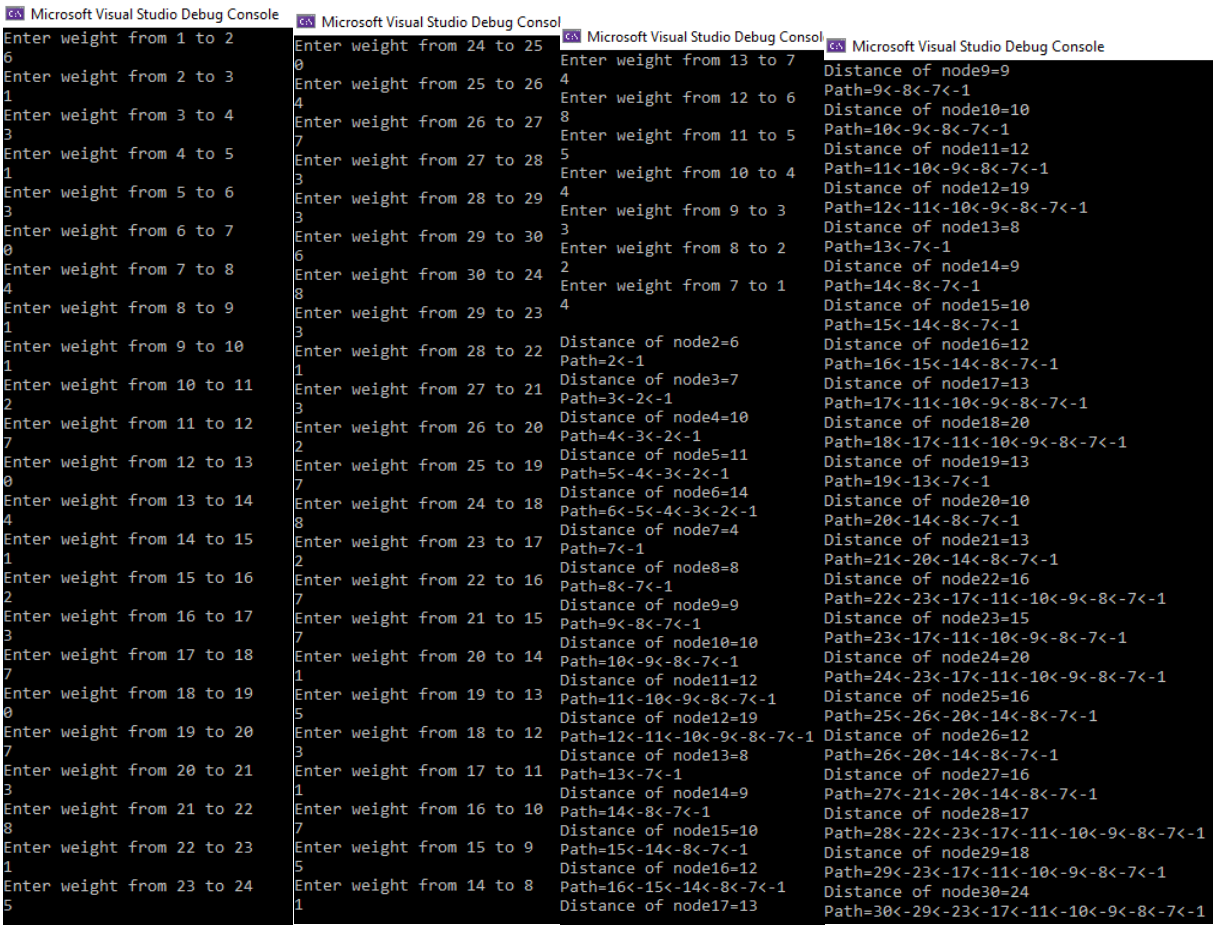


```cpp
#include <iostream>
#include <stdio.h>
 using namespace std;
 #define inf INT_MAX
 #define nodes 30

void dijkstra(double Graph[nodes][nodes], int n, int startNode) {
    double cost[nodes][nodes], dist[nodes], pred[nodes], minDist, visited[nodes];
    int  counter, nextNode, i, j;
    for ( i = 0; i < n; i++) {
        for ( j = 0; j < n; j++) {
            if (Graph[i][j] == 0) {
                cost[i][j] = inf;
            }
            else {
                cost[i][j] = Graph[i][j];
            }
        }
    }
    for ( i = 0; i < n; i++) {
        dist[i] = cost[startNode][i];
        pred[i] = startNode;
        visited[i] = 0;
    }
    dist[startNode] = 0;
    visited[startNode] = 1;
    counter = 1;

    while (counter < n-1)
    {
        minDist = inf;
        for (i = 0; i < n; i++) {
            if (dist[i] < minDist && !visited[i]) {
                minDist = dist[i];
                nextNode = i;
            }
        }
        visited[nextNode] = 1;
        for ( i = 0; i < n; i++) {
            if (!visited[i]) {
                if (minDist + cost[nextNode][i] < dist[i]) {
                    dist[i] = minDist + cost[nextNode][i];
                    pred[i] = nextNode;
                }
            }
        }
        counter++;
    }
    for (i = 0; i < n; i++) {
        if (i != startNode) {
            cout << "\nDistance of node" << i + 1 << "=" << dist[i];
            cout << "\nPath=" << i + 1;
            j = i;
            do {
                j = pred[j];
                cout << "<-" << j + 1;
            } while (j != startNode);
        }
    }
}

int main()
{
    double G[nodes][nodes];
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            G[i][j] = 0;
        }
    }

    int i = 0, j = 1;
    do {
        cout << "Enter weight from " << i + 1 << " to " << j + 1 << endl;
        cin >> G[i][j];
        i++;
        j++;
    } while (j < nodes);
    i = nodes - 1;
    j = i - 6;

    do {
        cout << "Enter weight from " << i + 1 << " to " << j + 1 << endl;
```



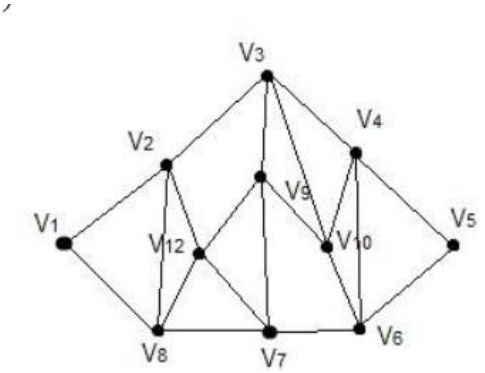
```cpp
int main()
{
    double G[nodes][nodes];
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            G[i][j] = 0;
        }
    }

    int i = 0, j = 1;
    do {
        cout << "Enter weight from " << i + 1 << " to " << j + 1 << endl;
        cin >> G[i][j];
        i++;
        j++;
    } while (j < nodes);
    i = nodes - 1;
    j = i - 6;

    do {
        cout << "Enter weight from " << i + 1 << " to " << j + 1 << endl;
        cin >> G[j][i];
        i--;
        j = i -6;
    } while (j >= 0);

    for (i = 0; i < nodes; i++) {
        for (j = 0; j < nodes; j++) {
            G[j][i] = G[i][j];
            G[j][i] = G[i][j];
        }
    }


    int n = 30;
    int u = 0;
    dijkstra(G, n, u);
}
```

Апробація програми:



Завдання № 8

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

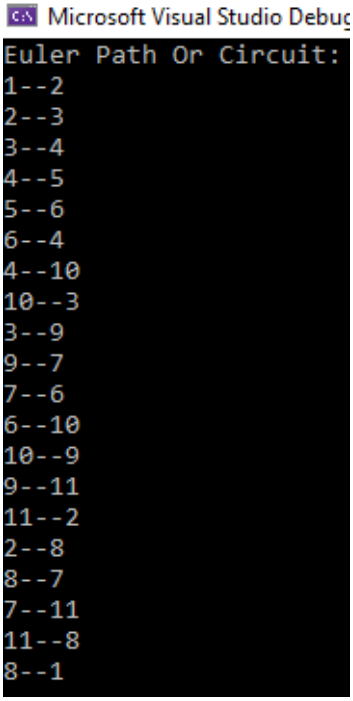


а)1-2-8-11-9-10-6-4-10-3-9-7-11-2-3-4-5-6-7-8-1

Програмна реалізація:

```cpp
#include<iostream>
#define NODE 11
using namespace std;
int graph[NODE][NODE] = {
        {0,1,0,0,0,0,0,1,0,0,0},
        {1,0,1,0,0,0,0,1,0,0,1},
        {0,1,0,1,0,0,0,0,1,1,0},
        {0,0,1,0,1,1,0,0,0,1,0},
        {0,0,0,1,0,1,0,1,0,0,0},
        {0,0,0,1,1,0,1,0,0,1,0},
        {0,0,0,0,0,1,0,1,1,0,1},
        {1,1,0,0,0,0,1,0,0,0,1},
        {0,0,1,0,0,0,1,0,0,1,1},
        {0,0,1,1,0,1,0,0,1,0,0},
        {0,1,0,0,0,0,0,1,1,1,0}
};
int tempGraph[NODE][NODE];
int findStartVert() {
    for (int i = 1; i < NODE; i++) {
        int deg = 0;
        for (int j = 0; j < NODE; j++) {
            if (tempGraph[i][j])
                deg++;
        }
        if (deg % 2 != 0)
            return i;
    }
    return 0;
}
bool isBridge(int u, int v) {
    int deg = 0;
    for (int i = 0; i < NODE; i++)
        if (tempGraph[v][i])
            deg++;
    if (deg > 1) {
        return false;
    }
    return true;
}
int edgeCount() {
    int count = 0;
    for (int i = 0; i < NODE; i++)
        for (int j = i; j < NODE; j++)
            if (tempGraph[i][j])
                count++;
    return count;
}
void fleuryAlgorithm(int start) {
    static int edge = edgeCount();
    for (int v = 0; v < NODE; v++) {
        if (tempGraph[start][v]) {
            if (edge <= 1 || !isBridge(start, v)) {
                cout << start + 1 << "--" << v + 1 << " " << endl;
                tempGraph[start][v] = tempGraph[v][start] = 0;
                edge--;
                fleuryAlgorithm(v);
            }
        }
    }
}
int main() {
    for (int i = 0; i < NODE; i++)
        for (int j = 0; j < NODE; j++)
            tempGraph[i][j] = graph[i][j];
    cout << "Euler Path Or Circuit: \n";
    fleuryAlgorithm(findStartVert());
}
```

Апробація програми:



б)

1) 1 – 2 – 8 – 1 +

2) 2 – 3 – 9 – 12 -2 +

3) 12 – 8 – 7 -12

4) 7 – 9 – 10 – 6 – 7

5) 3 – 10 - 4 - 3

5) 6 – 4 – 5 – 6

- **1 – 2 – 8 – 1**
- **1 – 2 – 3 – 9 – 12 - 2 – 8 – 1**
- **1 – 2 – 3 – 9 – 12 - 2 – 8- 12 – 7 – 9 – 10 – 6 – 7 – 8 – 1**
- **1 – 2 – 3 – 9 – 12 - 2 – 8- 12 – 7 – 9 – 10 – 6 – 4 – 10 – 3 – 4 – 5 – 6 – 7 – 8 – 1**

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

9. $(x \to y) \cdot (y \to z) \to (x \to z)$

(¬x∨y)(¬y∨z)→(x→z)

(¬x¬y∨¬xz∨yz)→(x→z)

(¬x¬y∨yz)→(¬x∨z)

¬(¬x¬y∨yz)∨¬xz

((x∨y)(¬y∨¬z))∨¬x∨z

x¬y∨x¬z∨y¬y∨y¬z∨¬x∨z

x¬y∨x¬z∨y¬z∨¬x∨z

y¬z∨x¬y∨¬x∨z

¬x∨¬y∨¬z∨z

¬x∨¬y∨1