МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконав:
студент групи КН-114
Долінський А.Г.
Викладач:
Мельникова Н.І.

Львів – 2019р.

Варіант №7

Завдання № 1

Виконати наступні операції над графами:

1) знайти доповнення до першого графу;

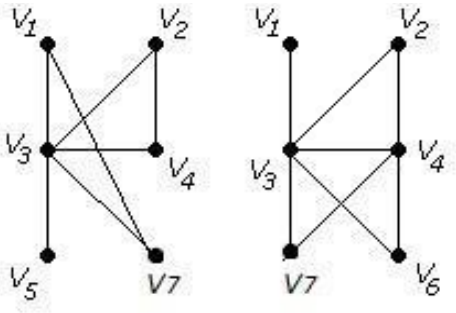
2) об’єднання графів;

3) кільцеву сумму G1 та G2 (G1+G2);

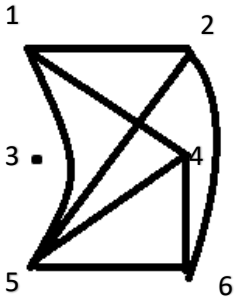
4) розмножити вершину у другому графі;

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

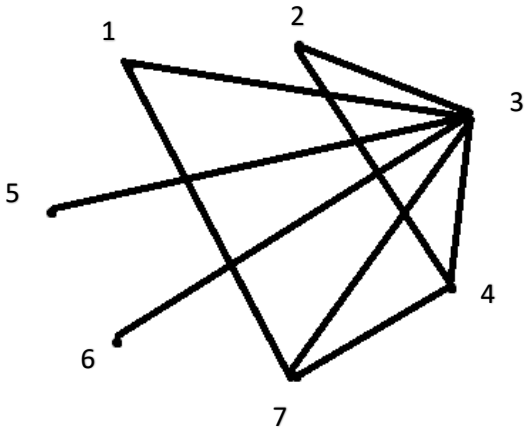
6) добуток графів;



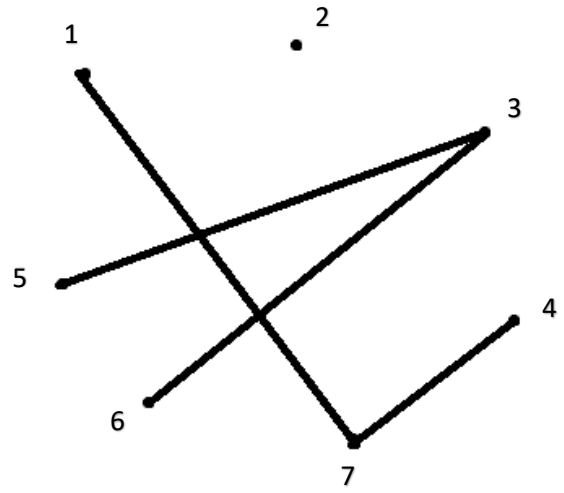
1)



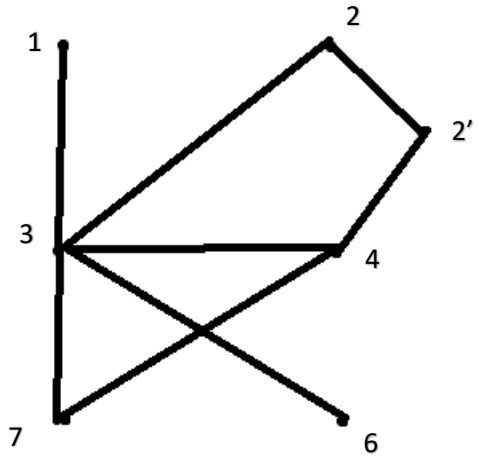
2)



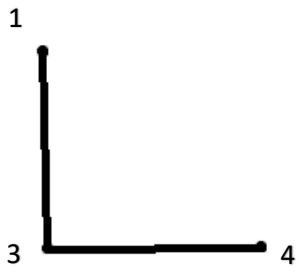
3)



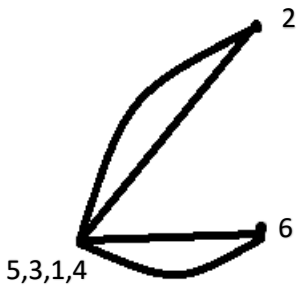
4)



5)



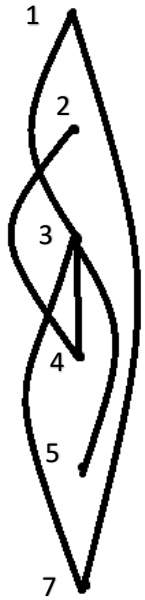
Підграф A



Стягнення A в G1

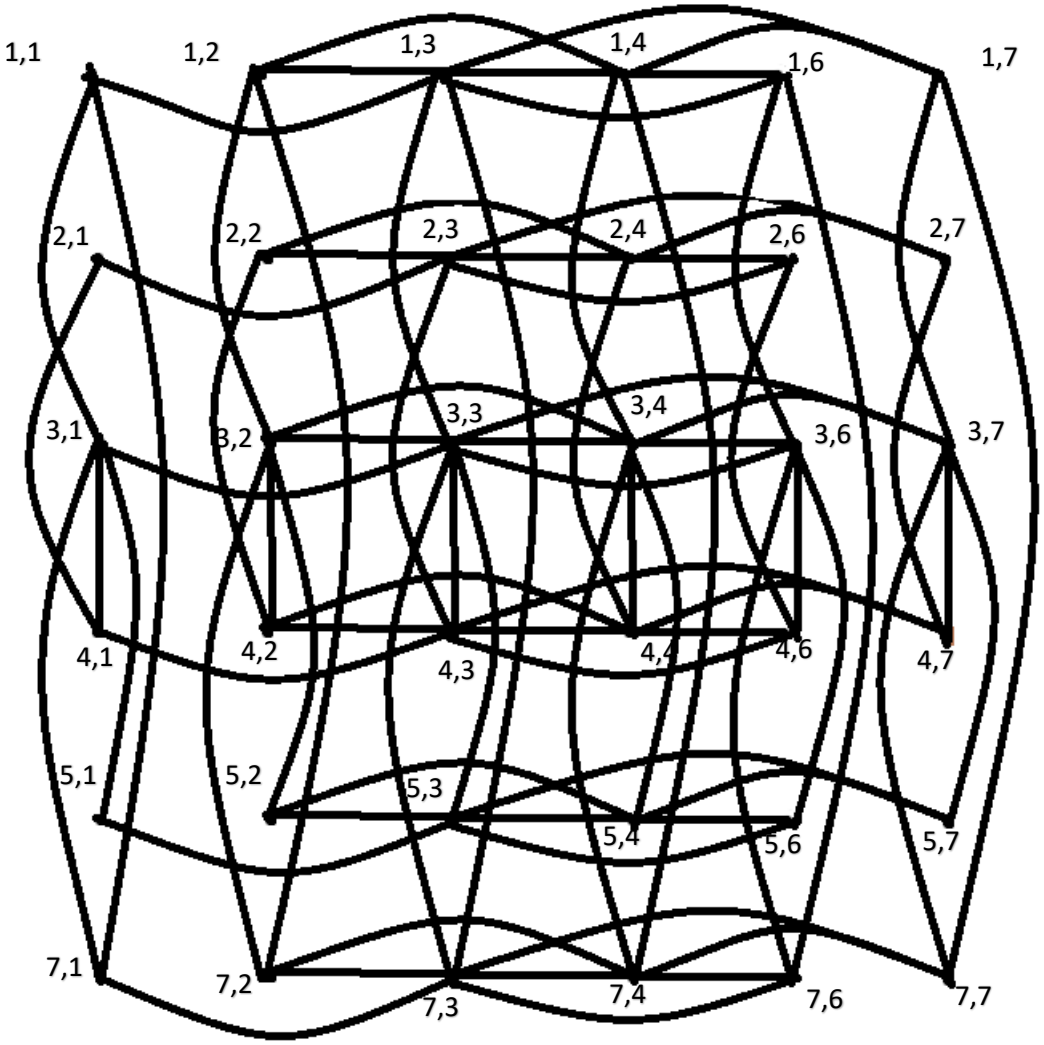
6)

G1


1
2
3
4
5
7

G2

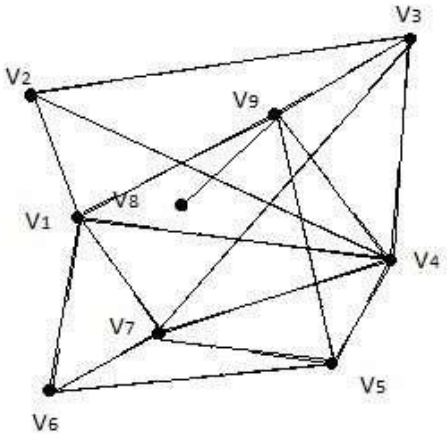

1
2
3
4
6
7


1,1
1,2
1,3
1,4
1,6
1,7
2,1
2,2
2,3
2,4
2,6
2,7
3,1
3,2
3,3
3,4
3,6
3,7
4,1
4,2
4,3
4,4
4,6
4,7
5,1
5,2
5,3
5,4
5,6
5,7
7,1
7,2
7,3
7,4
7,6
7,7

G1 x G2

Завдання № 2

Скласти таблицю суміжності для орграфа.



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

Завдання № 3

Для графа з другого завдання знайти діаметр.

	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	2	1	2	1	1	2	1
V2	1	0	1	1	2	2	2	3	2
V3	2	1	0	1	2	2	1	2	1
V4	1	1	1	0	1	2	1	2	1
V5	2	2	2	1	0	1	1	2	1
V6	1	2	2	2	1	0	1	3	2
V7	1	2	1	1	1	1	0	3	2
V8	2	3	2	2	2	3	3	0	1
V9	1	2	1	1	1	2	2	1	0

Діаметр = 3.

Завдання № 4

Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).

Вершина	Номер	Stack
V1	0	V1
V2	1	V1 V2
V3	2	V1 V2 V3
V4	3	V1 V2 V3 V4
V5	4	V1 V2 V3 V4 V5
V6	5	V1 V2 V3 V4 V5 V6
V7	6	V1 V2 V3 V4 V5 V6 V7
-	-	V1 V2 V3 V4 V5 V6
-	-	V1 V2 V3 V4 V5
-	-	V1 V2 V3 V4
V9	7	V1 V2 V3 V4 V9
V8	8	V1 V2 V3 V4 V9 V8
-	-	V1 V2 V3 V4 V9
-	-	V1 V2 V3 V4
-	-	V1 V2 V3
-	-	V1 V2
-	-	V1
-	-	-

```cpp
#include <iostream>
using namespace std;
int n;
int i, j;
bool* visited = new bool[n];


void DFS(int st, int** graph);

int main()
{
    cout << "Enter a number of vertexes: ";
    cin >> n;
    int** graph = new int* [n];
    for (int i = 0; i < n; i++) {
        graph[i] = new int[n];
    }
    int start;
    cout << "Matrix" << endl;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            cin >> graph[i][j];
        }
    }
    for (i = 0; i < n; i++)
```

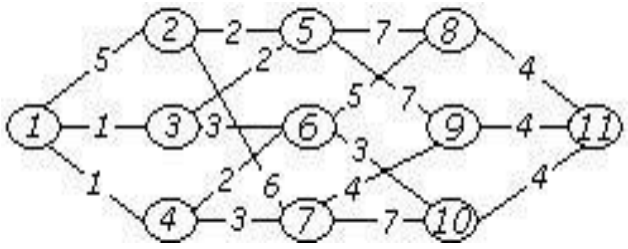
```
    {
        visited[i] = false;
    }
    cout << "Enter a vertex to start with: ";
    cin >> start;
    bool* vis = new bool[n];
    cout << "Search is done: ";
    DFS(start - 1, graph);
    delete[]visited;
    for (int i = 0; i < n; i++) {
        delete[] graph[i];
    }
    delete[] graph;
    system("pause");
    return 0;
}
void DFS(int st, int** graph)
{
    int r;
    cout << st + 1 << " ";
    visited[st] = true;
    for (r = 0; r <= n; r++)
        if ((graph[st][r] != 0) && (!visited[r]))
            DFS(r, graph);
}
```

```
Microsoft Visual Studio Debug Console
Enter a number of vertexes: 9
Matrix
0 1 0 1 0 1 1 0 1
1 0 1 1 0 0 0 0 0
0 1 0 1 0 0 1 0 1
1 1 1 0 1 0 1 0 1
0 0 0 1 0 1 1 0 1
1 0 0 0 1 0 1 0 0
1 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 1
0 0 1 1 1 0 0 1 0
Enter a vertex to start with: 3
Search is done: 3 2 1 4 5 6 7 9 8
```

```
Microsoft Visual Studio Debug Console
Enter a number of vertexes: 9
Matrix
0 1 0 1 0 1 1 0 1
1 0 1 1 0 0 0 0 0
0 1 0 1 0 0 1 0 1
1 1 1 0 1 0 1 0 1
0 0 0 1 0 1 1 0 1
1 0 0 0 1 0 1 0 0
1 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 1
0 0 1 1 1 0 0 1 0
Enter a vertex to start with: 1
Search is done: 1 2 3 4 5 6 7 9 8
```

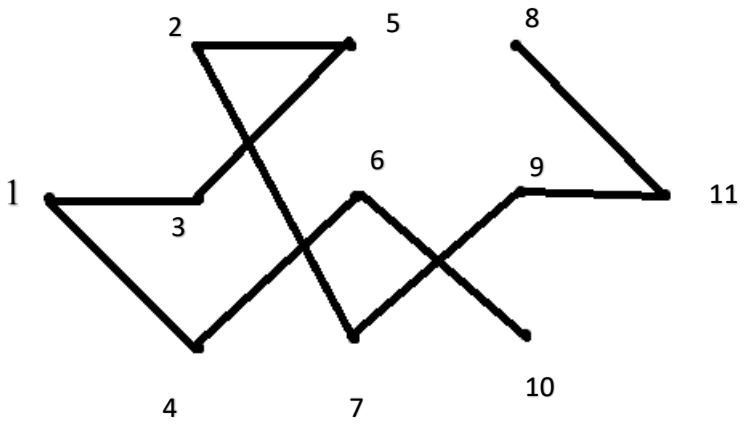
Завдання № 5

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



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

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



Метод Прима.

```
#include <iostream>

using namespace std;
struct edge {
    int firstPoint;
    int secondPoint;
    int weight;
};
void sortEdges(edge* a, int n);
bool isInclude(int* arr, int n, int k);
void findTheTree(edge* a, int* points, edge *tree, int &n, int &k, int &i, int &j);
bool isMinimum(int w, edge* a, int k, int* points, int n);
int main() {
    int n;
    int k;
    cout << "Enter a number of points: ";
    cin >> n;
    cout << "Enter a number of edges: ";
    cin >> k;
    edge* edges = new edge[k];
    cout << "Enter edges (first point | second point | weight) " << endl;
    for (int i = 0; i < k; i++) {
        cin >> edges[i].firstPoint >> edges[i].secondPoint >> edges[i].weight;

    }
    sortEdges(edges, k);
    for(int i = 0; i < k; i ++){
        cout << edges[i].firstPoint << " " << edges[i].secondPoint << ", " << edges[i].weight << endl;
    }
    cout << endl;
```

```
    int* points = new int[n];
    points[0] = edges[0].firstPoint;
    points[1] = edges[0].secondPoint;
    edge* tree = new edge[n-1];
    tree[0].firstPoint = points[0];
    tree[0].secondPoint = points[1];
    int i = 2;
    int j = 1;



    findTheTree(edges, points, tree, n, k, i, j);
    cout << i << " " << j << endl;
    cout << "V(G) = { ";
    for (int i = 0; i < n; i++) {
        cout << points[i] << ", ";
    }
    cout << "}" << endl;
    cout << "E(G) = { ";
    for (int i = 0; i < n-1; i++) {
        cout <<"("<< tree[i].firstPoint << "," << tree[i].secondPoint << ")"<<" ";
    }
    cout << " }" << endl;


    return 0;
}
```

```
void findTheTree(edge* a, int* points, edge *tree, int &n, int &k, int &i, int &j) {
    if(i == n){
        return;
    }
    else if (j == k) {
        j = 1;
    }

        if(isInclude(points, n, a[j].firstPoint) && isInclude(points, n, a[j].secondPoint)){
            j++;
            findTheTree(a, points, tree, n, k, i, j);
        }
        else if (!isInclude(points, n, a[j].firstPoint) && isInclude(points, n, a[j].secondPoint) && isMinimum(a[j].weight, a, k, points, n)) {
            tree[i - 1].firstPoint = a[j].secondPoint;
            tree[i - 1].secondPoint = a[j].firstPoint;
            points[i] = a[j].firstPoint;
            tree[i - 1].weight = a[j].weight;

            j++;
            i++;
            findTheTree(a, points, tree, n, k, i, j);

        }
        else if (isInclude(points, n, a[j].firstPoint) && !isInclude(points, n, a[j].secondPoint) && isMinimum(a[j].weight, a, k, points, n)) {
            tree[i - 1].firstPoint = a[j].firstPoint;
            tree[i - 1].secondPoint = a[j].secondPoint;
            points[i] = a[j].secondPoint;

            tree[i - 1].weight = a[j].weight;

            j++;
            i++;
            findTheTree(a, points, tree, n, k, i, j);
        }
        else {
            j++;
            findTheTree(a, points, tree, n, k, i, j);
        }




}
void sortEdges(edge* a, int n) {
    edge temp;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (a[j].weight > a[j+1].weight) {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}
```

```
bool isInclude(int* arr, int n, int k) {
    for (int i = 0; i < n; i++) {
        if (k == arr[i]) {
            return true;
        }

    }
    return false;
}
```

```
bool isMinimum(int w, edge* a,int k, int* points, int n) {
    for (int j = 1; j < k; j++) {
        if (((!isInclude(points, n, a[j].firstPoint) && isInclude(points, n, a[j].secondPoint)) || (isInclude(points, n, a[j].firstPoint) && !isInclude(points, n, a[j].secondPoint))) && a[j].weight < w) {
            return false;
        }
    }
    return true;
}
```

```
C:\Users\Admin\Documents\CodeBlocksProjects\12345\bin\Debug\12345.exe
Enter a number of points: 11
Enter a number of edges: 18
Enter edges (first point | second point | weight)
1 2 5
1 3 1
1 4 1
2 5 2
2 7 6
3 5 2
3 6 3
4 6 2
2 7 3
5 8 7
5 9 7
6 8 5
6 10 3
7 9 4
7 10 7
8 11 4
9 11 4
10 11 4

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

Process returned 0 (0x0)   execution time : 14.562 s
Press any key to continue.
```

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

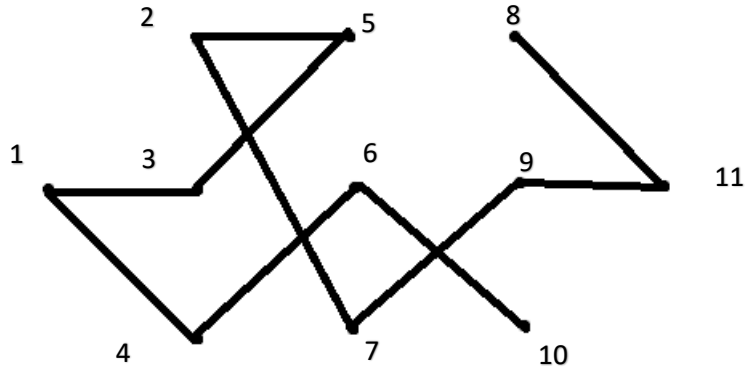
(4,7) – утв. цикл;

(1,2), (6,8) – утв. цикл;

(7,2) – утв. цикл;

(10, 11) – утв. цикл.

(7,10), (5,9), (5,8) – утв. цикл;



Метод Краскала.

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

using namespace std;
vector<pair<int, pair<int, int> > > graph;
vector<pair<int, int> > the_tree;
int main() {
    int m, n, fp, sp, w, cost = 0;

    cout << "Enter a number of points:" << endl;

    cin >> n;
    cout << "Enter a number of edges:" << endl;
    cin >> m;

    vector<int> tree_temp(n);

    cout << "Enter edges (first point | second point | weight)" << endl;

    for (int i = 0; i < m; ++i) {

        cin >> fp >> sp >> w;

        fp--; sp--;

        graph.push_back({ w,{fp,sp} });
    }
    sort(graph.begin(), graph.end());

    for (int i = 0; i < n; ++i)
```

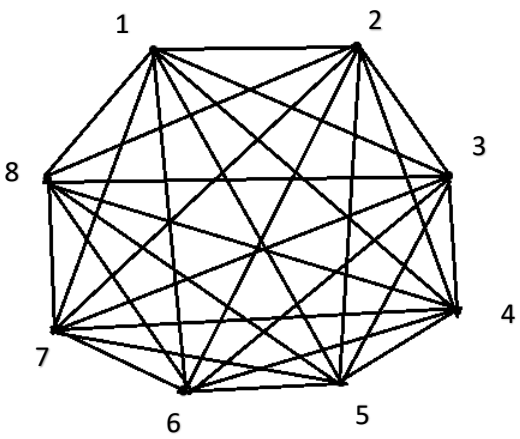
```cpp
        tree_temp[i] = i;
    for (int i = 0; i < m; ++i)
    {
        int x = graph[i].second.first, y = graph[i].second.second, l = graph[i].first;

        if (tree_temp[x] != tree_temp[y])
        {
            cost += l;
            the_tree.emplace_back(x + 1, y + 1);

            int prev_id = tree_temp[x], new_id = tree_temp[y];

            for (int j = 0; j < n; ++j)

                if (tree_temp[j] == prev_id)

                    tree_temp[j] = new_id;
        }
    }
    cout << "THE TREE:" << endl;
    cout << "E(G) = { ";
    for (auto i : the_tree) {

        cout <<" ("<< i.first << ',' << i.second << ") ";

    }
    cout << " }" << endl;
    cout << "Weight: " << cost;
}
```

```
Microsoft Visual Studio Debug Console
Enter a number of points:
11
Enter a number of edges:
18
Enter edges (first point | second point | weight)
1 2 5
1 3 1
1 4 1
2 5 2
2 7 6
3 5 2
3 6 3
4 6 2
2 7 3
5 8 7
5 9 7
6 8 5
6 10 3
7 9 4
7 10 7
8 11 4
9 11 4
10 11 4
THE TREE:
E(G) = {  (1,3)  (1,4)  (2,5)  (3,5)  (4,6)  (2,7)  (6,10)  (7,9)  (8,11)  (9,11)  }
Weight: 26
```

Завдання № 6

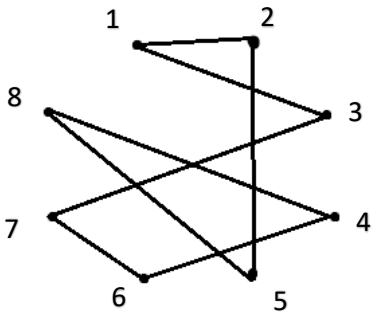
Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

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



1. 1-5-4-8-6-7-2-3-1 = 25
   1-5-7-6-4-8-7-3-1 = 24
   1-5-8-4-6-7-2-3-1 = 26
2. 2-4-5-7-6-8-3-1-2 = 25
   2-4-8-5-7-6-3-1-2 = 24
   2-5-4-8-6-7-3-1-2 = 24
   2-5-8-4-6-7-3-1-2 = 23
   2-5-7-6-4-8-3-1-2 = 24
3. 3-4-5-7-6-8-1-2-3 = 28
   3-7-5-8-4-6-2-1-3 = 24
   3-4-8-5-7-6-2-1-3 = 24
4. 4-5-8-6-7-2-1-3-4 = 26
   4-8-5-7-6-2-1-3-4 = 24
5. 5-8-4-6-7-2-1-3-5 = 25
   5-4-8-6-7-2-1-3-5 = 26
   5-7-6-4-8-3-1-2-5 = 24
6. 6-4-5-8-7-2-1-3-6 = 28
   6-7-5-8-4-2-1-3-6 = 24
7. 7-5-4-8-6-2-1-3-7 = 25
8. 8-4-5-7-6-2-1-3-8 = 25
   8-5-4-6-7-2-1-3-8 = 25

Найкоротший шлях: 2-5-8-4-6-7-3-1-2 = 23.




```cpp
#include <iostream>
#include <utility>
#include <vector>
#include <algorithm>
using namespace std;
int graph[100][100] = { 0 };
int controlLine[100] = { 0 };
pair<int, int> controlpair[100];
vector<int> indexes;

void Index(int size, int ins);
int main() {
    cout << "Enter a number of vertexes: ";
    int n;
    cin >> n;
    cout << "Matrix" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> graph[i][j];
            controlLine[j] += graph[i][j];
        }
    }

    for (int i = 0; i < n; i++) {
        controlpair[i] = make_pair(controlLine[i], i);
    }
```

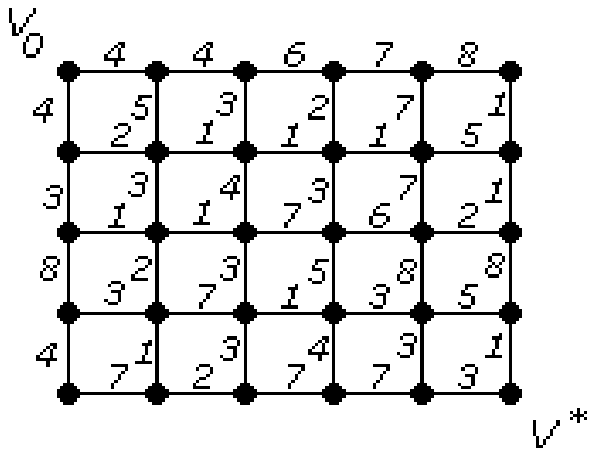
```
    sort(controlpair, controlpair + n);
    reverse(controlpair, controlpair + n);

    for (int i = 0; i < 3; i++)
        indexes.push_back(controlpair[i].second);


    for (int i = 3; i < n; i++) {
        Index(i, controlpair[i].second);
    }
    cout << "THE BEST WAY" << endl;
    for (int i = 0; i < n; i++) {
        cout << indexes[i] + 1 << " -> ";
    }


}
void Index(int size, int ins) {
    int d = INT_MAX;
    int nd;
    int current_position = -1;
```

```
    for (int i = 0; i < size - 1; i++) {
        nd = graph[indexes[i]][ins] + graph[ins][indexes[i + 1]] - graph[indexes[i]][indexes[i + 1]];
        if (nd < d) {
            d = nd;
            current_position = i + 1;
        }
    }
    nd = graph[indexes[size - 1]][ins] + graph[ins][indexes[0]] - graph[indexes[size - 1]][indexes[0]];
    if (nd < d) {
        current_position = size;
    }
    indexes.insert(indexes.begin() + current_position, ins);

}
```

```
Microsoft Visual Studio Debug Console
Enter a number of vertexes: 8
Matrix
0 5 5 5 4 6 5 5
5 0 7 3 3 5 4 5
5 7 0 4 5 6 4 5
5 3 4 0 1 2 5 1
4 3 5 1 0 5 1 1
6 5 6 2 5 0 2 3
5 4 4 5 1 2 0 5
5 5 5 1 1 3 5 0
THE BEST WAY
3 -> 1 -> 2 -> 5 -> 4 -> 8 -> 6 -> 7 ->
```

Завдання № 7

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



V0 = 0

V1 = 4

V2 = 4

V3 = 8

V4 = ~~9~~ 6

V5 = 7

V6 = 7

V7 = ~~9~~ 8

V8 = ~~15~~ 13

V9 = 8

V10 = ~~11~~ 9

V11 = ~~14~~ 10

V12 = 10

V13 = 9

V14 = 11

V15 = 12

V16 = 14

V17 = 16

V18 = 16

V19 = 11

V20 = 16 ~~13~~

V21 = 13

V22 = 18 ~~17~~

V23 = 16

V24 = 17

V25 = 15

V26 = 15

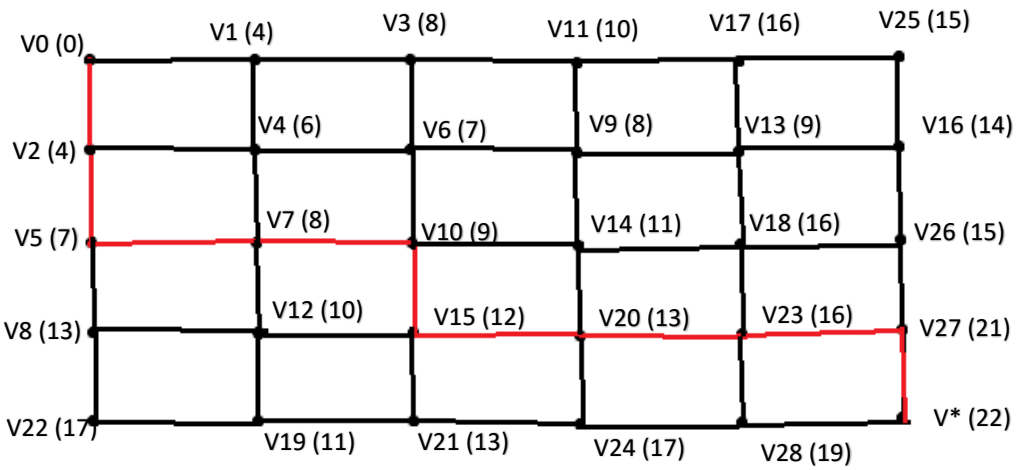
V27 = 23 ~~21~~

V28 = 19

V^* = 22

V0 -> V2 -> V5 -> V7 -> V10 -> V15 -> V20 -> V23 -> V27 -> V^*

Відстань - 22



```cpp
#include <iostream>
using namespace std;
int numberOfPoints;
int graph[99][99] = { 0 };
int way[99];
int shortestDistance[99];
bool isVisited[99];
void printAWay(int m);
int calculateADistance();
void findTheShortestWay(int graph[99][99]);
int main() {
    cout << "Enter a number of points: ";
    cin >> numberOfPoints;
    int s, e;
    cout << "Enter your graph`s size (rows / columns)";
    cin >> s >> e;
    for (int i = 0; i < numberOfPoints; i++) {
        for (int j = i + 1; j < numberOfPoints; j++) {
            if (j == i + 1 || j == i + s) {
                cout << i + 1 << " - " << j + 1 << ": ";
                cin >> graph[i][j];
            }
            else {
                graph[i][j] = 0;
            }
        }
    }
    findTheShortestWay(graph);
    cout << "The shortest way is: " << shortestDistance[numberOfPoints - 1] << endl;
    cout << "THE WAY" << endl;
    cout << "1 -> ";
    printAWay(29);
    cout << "END" << endl;
    system("pause");

    return 0;
}

int calculateADistance() {

    int min = 9999;
    int minDistance;
    for (int c = 0; c < numberOfPoints; c++) {
        if (isVisited[c] == false && shortestDistance[c] <= min) {
            min = shortestDistance[c];
            minDistance = c;
        }
    }
    return minDistance;
}

void findTheShortestWay(int graph[99][99]) {
    way[0] = -1;
    for (int i = 0; i < numberOfPoints; i++) {

        shortestDistance[i] = 9999;
        isVisited[i] = false;
    }
    shortestDistance[0] = 0;
    for (int i = 0; i < numberOfPoints - 1; i++) {
        int v = calculateADistance();
        isVisited[v] = true;
        for (int j = 0; j < numberOfPoints; j++) {
            if (isVisited[j] == false && (graph[v][j] && shortestDistance[v] + graph[v][j] < shortestDistance[j])) {
                way[j] = v;
                shortestDistance[j] = shortestDistance[v] + graph[v][j];
            }
        }
    }

}
```

```
void printAWay(int m) {
    if (way[m] == -1) {
        return;
    }
    printAWay(way[m]);
    cout << m + 1 << " -> ";
}
```

C:\Users\Admin\source\repos\Lab 5 (math)\Debug\Lab 5 (math).exe

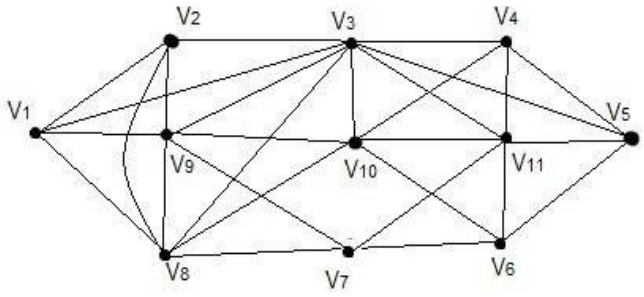
```
Enter a number of points: 30
Enter your graph`s size (rows / columns)6 5
1 - 2: 4
1 - 7: 4
2 - 3: 4
2 - 8: 5
3 - 4: 6
3 - 9: 3
4 - 5: 7
4 - 10: 2
5 - 6: 8
5 - 11: 7
6 - 7: 0
6 - 12: 1
7 - 8: 2
7 - 13: 3
8 - 9: 1
8 - 14: 3
9 - 10: 1
9 - 15: 4
10 - 11: 1
10 - 16: 3
11 - 12: 5
11 - 17: 7
12 - 13: 0
12 - 18: 1
13 - 14: 1
13 - 19: 8
14 - 15: 1
14 - 20: 2
15 - 16: 7
15 - 21: 3
16 - 17: 6
16 - 22: 5
17 - 18: 2
17 - 23: 8
18 - 19: 0
18 - 24: 8
19 - 20: 3
19 - 25: 4
20 - 21: 7
20 - 26: 1
21 - 22: 1
21 - 27: 3
22 - 23: 3
22 - 28: 4
23 - 24: 5
23 - 29: 3
24 - 25: 0
24 - 30: 1
25 - 26: 7
26 - 27: 2
27 - 28: 7
28 - 29: 7
29 - 30: 3
```

```
The shortest way is: 22
THE WAY
1 -> 7 -> 13 -> 14 -> 15 -> 21 -> 22 -> 23 -> 29 -> 30 -> END
Press any key to continue . . .
```

Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері; б) елементарних циклів.



а)

Ейлерів цикл: 1 – 9 – 7 – 11 – 6 – 10 – 8 – 9 – 3 – 10 – 4 – 11 – 3 – 8 – 2 – 9 – 10 – 11 – 5 – 3 – 1 – 2 – 3 – 4 - 5 – 6 – 7 - 8 - 1

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

int findStartVertex(int** tempGraph, int m);
int edgeCount(int** tempGraph, int m);
void FleuryAlgorithm(int start, int** tempGraph, int m);
int main() {
    // Введення матриці суміжності графа.
    int m;
    cout << "Enter a number of vertexes: ";
    cin >> m;
    int** graph = new int* [m];
    for (int i = 0; i < m; i++) {
        graph[i] = new int[m];
    }
    int** tempGraph = new int* [m];
    for (int i = 0; i < m; i++) {
        tempGraph[i] = new int[m];
    }
    cout << "Matrix" << endl;
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < m; j++) {
            cin >> graph[i][j];
        }
    }
    for (int i = 0; i < m; i++)
        for (int j = 0; j < m; j++)
```

```
            tempGraph[i][j] = graph[i][j];
    cout << "Euler Loop: ";
    FleuryAlgorithm(findStartVertex(tempGraph, m), tempGraph, m);
    cout << "END" << endl;
    system("pause");
    return 0;
}

int findStartVertex(int** tempGraph, int m) {
    for (int i = 0; i < m; i++) {
        int deg = 0;
        for (int j = 0; j < m; j++) {  //Шукаємо вершину, з якої починаємо, вона має мати парний степінь.
            if (tempGraph[i][j])
                deg++;
        }
        if (deg % 2 != 0)
            return i;
    }
    return 0; //Якщо всі вершини мають парний степінь, то починаємо з 0 (1).
}

bool isBridge(int u, int v, int** tempGraph, int m) { // Перевірка чи ребро є "мостом".
    int deg = 0;
    for (int i = 0; i < m; i++)
        if (tempGraph[v][i])
            deg++;
    if (deg > 1) {
```

```
        return false;
    }
    return true;
}

int edgeCount(int** tempGraph, int m) {
    int count = 0;
    for (int i = 0; i < m; i++)
        for (int j = i; j < m; j++)
            if (tempGraph[i][j])
                count++;
    return count; // Підрахунок ребер у графі.
}
void FleuryAlgorithm(int start, int** tempGraph, int m) { // Безпосередньо алгоритм.
    int edge = edgeCount(tempGraph, m);
    for (int i = 0; i < m; i++) {
        if (tempGraph[start][i]) {
            if (edge <= 1 || !isBridge(start, i, tempGraph, m)) {
                cout << "(" << start + 1 << "," << i + 1 << ")  -> ";
                tempGraph[start][i] = tempGraph[i][start] = 0; //Забираємо ребро із тимчасового графа.
                edge--;
                FleuryAlgorithm(i, tempGraph, m);
            }
        }
    }
}
```

```
C:\Users\Admin\source\repos\Fleury\Debug\Fleury.exe
Enter a number of vertexes: 11
Matrix
0 1 1 0 0 0 0 1 1 0 0
1 0 1 0 0 0 0 1 1 0 0
1 1 0 1 1 0 0 1 1 1 1
0 0 1 0 1 0 0 0 0 1 1
0 0 1 1 0 1 0 0 0 0 1
0 0 0 0 1 0 1 0 0 1 1
0 0 0 0 0 1 0 1 1 0 1
1 0 1 0 0 0 1 0 1 1 0
1 1 1 0 0 0 1 1 0 1 0
0 0 1 1 0 1 0 1 1 0 1
0 0 1 1 1 1 1 0 0 1 0
Euler Loop: (8,1)  -> (1,2)  -> (2,3)  -> (3,1)  -> (1,9)  -> (9,2)  -> (2,8)  -> (8,3)  -> (3,4)  -> (4,5)  -> (5,3)  -
> (3,9)  -> (9,7)  -> (7,6)  -> (6,5)  -> (5,11)  -> (11,3)  -> (3,10)  -> (10,4)  -> (4,11)  -> (11,6)  -> (6,10)  -> (
10,8)  -> (8,7)  -> (7,11)  -> (11,10)  -> (10,9)  -> (9,8)  -> END
Press any key to continue . . .
```

Завдання №9

Спростити формули (звести їх до скороченої ДНФ)

7. $(x \cdot \overline{x \cdot \bar{x}} \to y \cdot \bar{y} \to z)$

$x * \overline{x * \bar{x}} \to y * \bar{y} \to z = x * x * \bar{x} \to y * \bar{y} + z = x * x * x + y * \bar{y} + z =$

$= x + (y * \bar{y}) + z = x + z$