МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахунково-графічні завдання

З дисципліни
“Дискретна математика”

Виконав:
Студент групи КН-115
Лукавий Мар’ян
Викладач:
Мельникова Н.І.

Варіант 14

Завдання № 1

Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

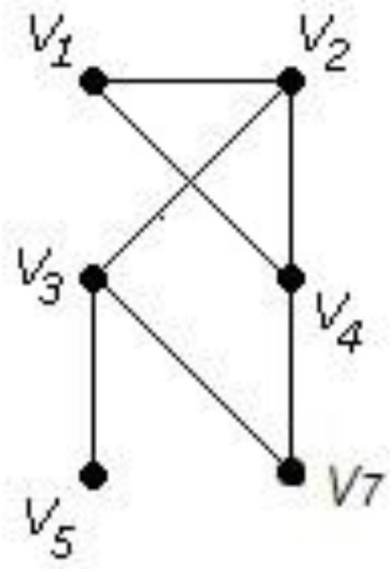
2) об'єднання графів,

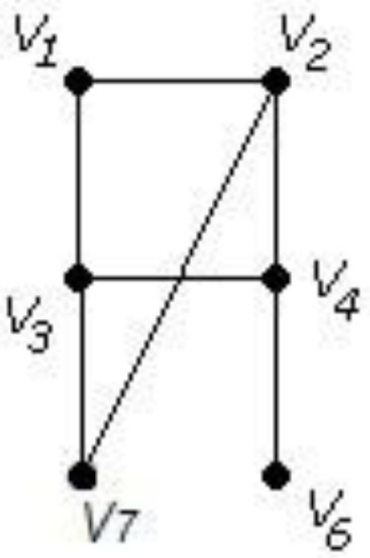
3) кільцеву сумму G1 та G2 (G1+G2),

4) розмножити вершину у другому графі,

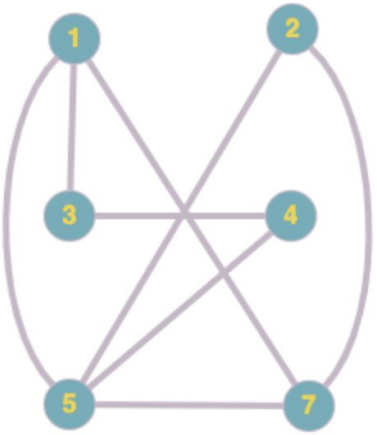
5) виділити підграф A - що скадається з 3-х вершин в G1

6) добуток графів.

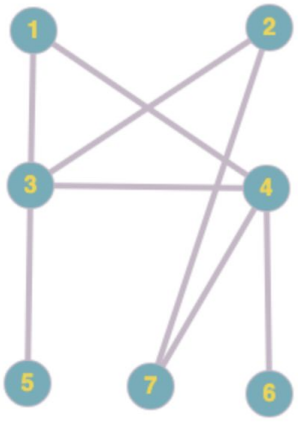




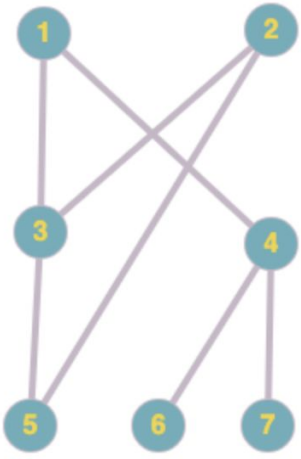
1) знайти доповнення до першого графу,



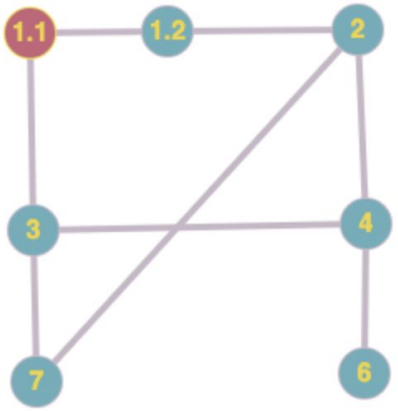
2) об'єднання графів,



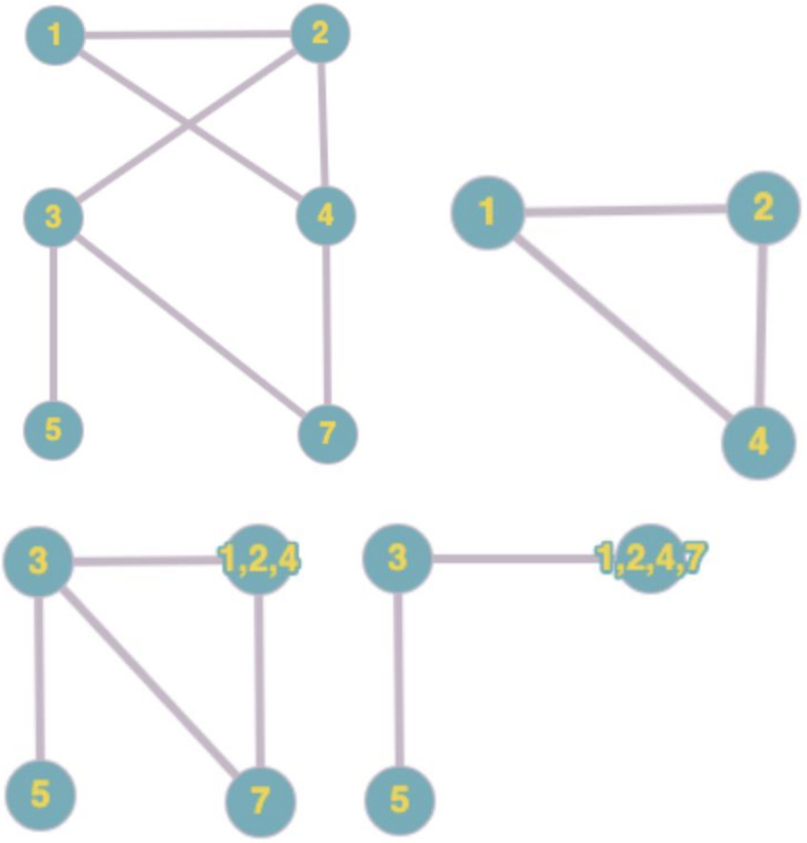
3) кільцеву сумму G1 та G2 (G1+G2),



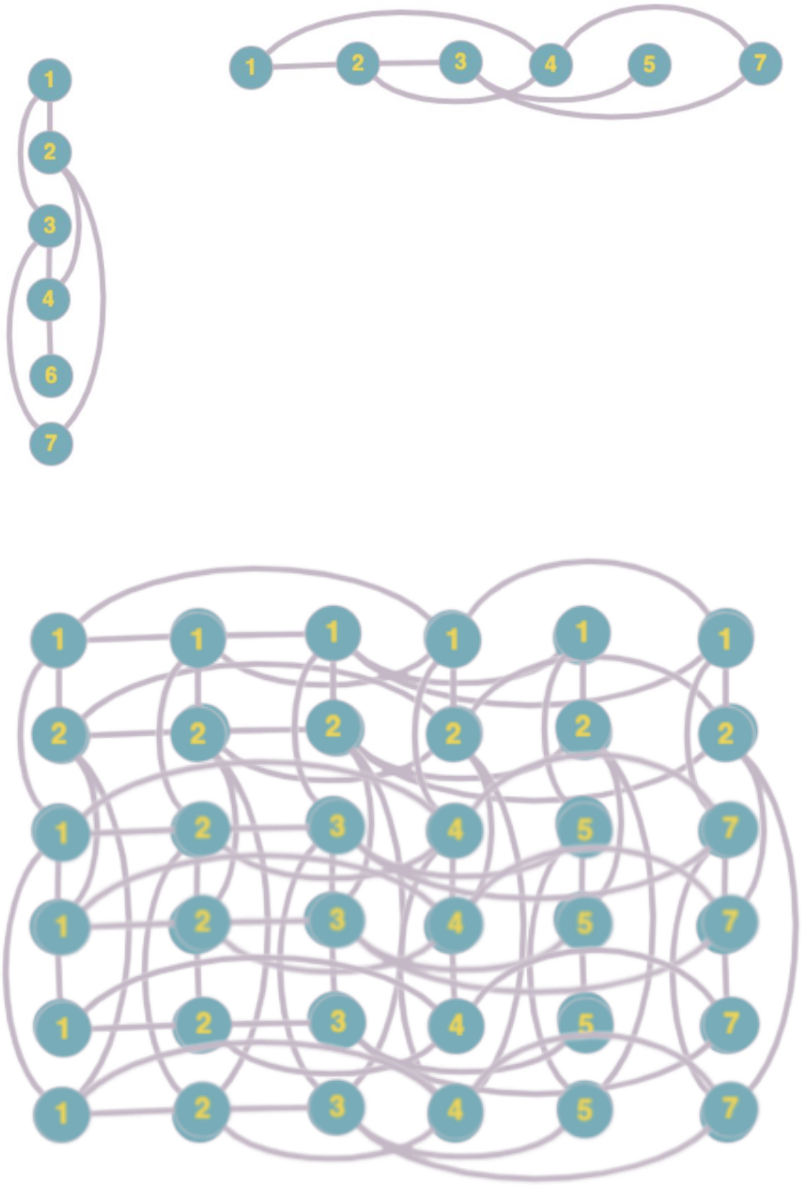
4) розмножити вершину у другому графі,



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

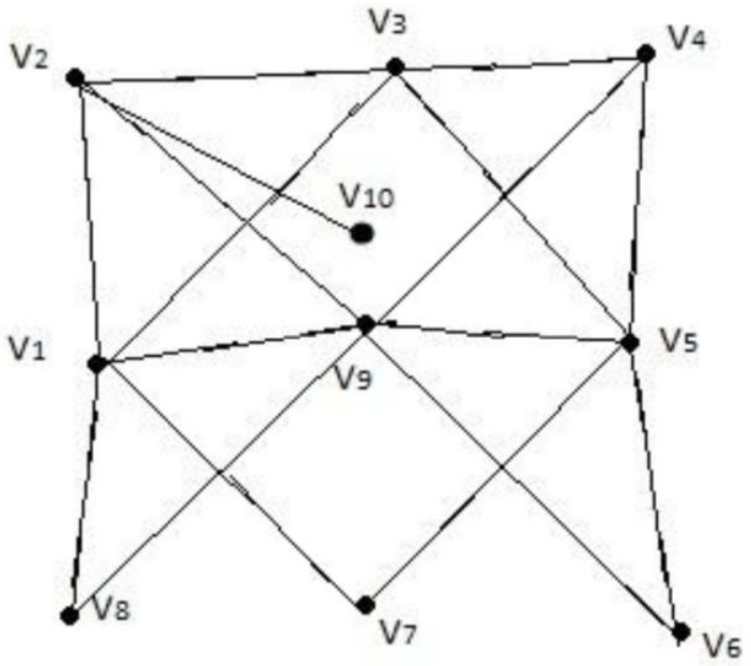


6) добуток графів.



Завдання № 2

Скласти таблицю суміжності для орграфа.



	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	0	1	1	1	0
2	1	0	1	0	0	0	0	0	1	1
3	1	1	0	1	1	0	0	0	0	0
4	0	0	1	0	1	0	0	0	1	0
5	0	0	1	1	0	1	1	0	1	0
6	0	0	0	0	1	0	0	0	1	0
7	1	0	0	0	1	0	0	0	0	0
8	1	0	0	0	0	0	0	0	1	0
9	1	1	0	1	1	1	0	1	0	0
10	0	1	0	0	0	0	0	0	0	0

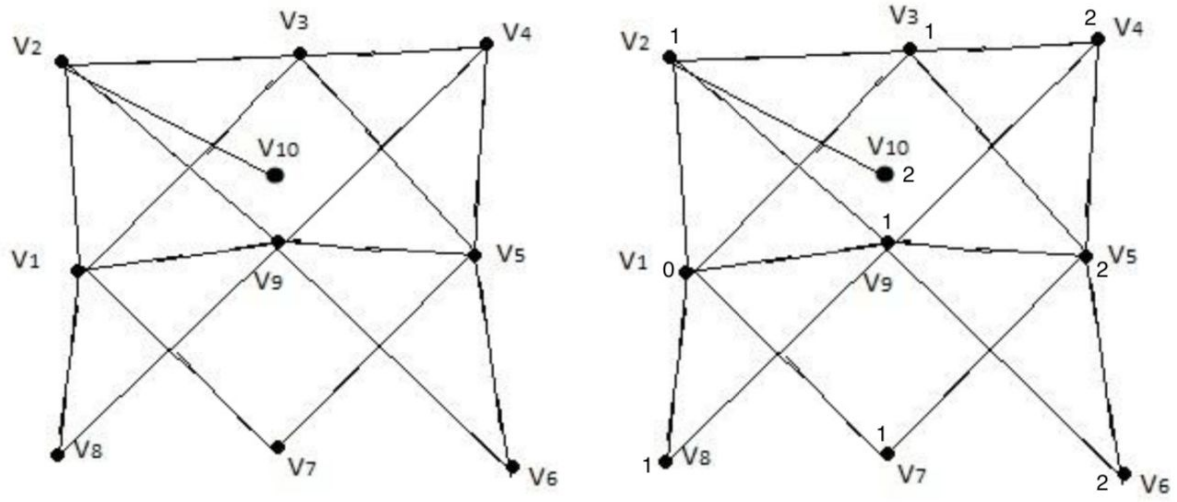
Завдання № 3

Для графа з другого завдання знайти діаметр.

V1=>V2=>V10

Завдання № 4

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



0: q = (V1)

1: V1, q = (V2, V3, V9,V7, V8)

2: V2, q = (V3, V9, V8, V7, V10)

3: V3, q = (V9, V8, V7, V10, V4, V5)

4: V9, q = (V8, V7, V10, V4, V5, V6)

5: V8, q = (V7, V10, V4, V5, V6)

6: V7, q = (V10, V4, V5, V6)

7: V10, q = (V4, V5, V6)

8: V4, q = (V5, V6)

9: V5, q = (V6)

10: V6, q = ( )

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

class Di
{
    int V;

    list<int> *adjList;
public:
    Di(int V);

    void addEdge(int v, int w);

    void BFS(int s);
};

Di::Di(int V)
{
    this->V = V;
    adjList = new list<int>[V];
}
 void Di::addEdge(int v, int w)
{
    adjList[v].push_back(w);
}
```

```
void Di::BFS(int s)
{
    bool *visited = new bool[V];
    for(int i = 0; i < V; i++)
        visited[i] = false;

    list<int> queue;

    visited[s] = true;
    queue.push_back(s);

    list<int>::iterator i;

    while(!queue.empty())
    {
        s = queue.front();
        cout << s << " ";
        queue.pop_front();

        for (i = adjList[s].begin(); i
        {
            if (!visited[*i])
            {
                visited[*i] = true;
                queue.push_back(*i);
            }
```

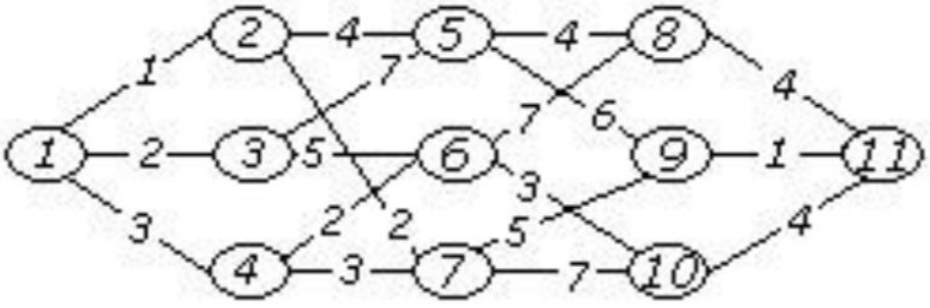
```
}
int main()
{
        Di g(11);
    g.addEdge(1, 2);
    g.addEdge(1, 8);
    g.addEdge(1, 9);
    g.addEdge(1, 3);
    g.addEdge(1, 7);

    g.addEdge(2, 1);
    g.addEdge(2, 10);
    g.addEdge(2, 3);
    g.addEdge(2, 9);

    g.addEdge(3, 2);
    g.addEdge(3, 1);
    g.addEdge(3, 5);
    g.addEdge(3, 4);

    g.addEdge(4, 3);
    g.addEdge(4, 9);
    g.addEdge(4, 5);

    g.addEdge(5, 4);
    g.addEdge(5, 9);
    g.addEdge(5, 6);

    g.addEdge(6, 9);
    g.addEdge(6, 5);
```

```
    g.addEdge(8, 1);
    g.addEdge(8, 9);

    g.addEdge(9, 1);
    g.addEdge(9, 2);
    g.addEdge(9, 5);
    g.addEdge(9, 4);
    g.addEdge(9, 6);
    g.addEdge(9, 8);
    g.addEdge(10, 2);


    g.BFS(1);

    return 0;
```

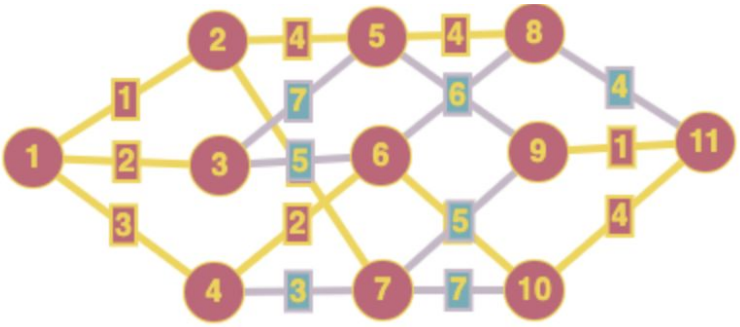
```
1 2 8 9 3 7 10 5 4 6
```

Завдання № 5

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



1) Краскала



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

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

2) Прима

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

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

Мінімальне остове дерево графа: 26

```cpp
#include<cstdio>
#include <iostream>
using namespace std;
int disjoint(int * Source, int * Dest, int source, int dest) {
    int verdict = 0;
    for (int i = 0; i < 18; i++) {
        if(Source[i] == source) {
            verdict += 1;
            break;
        }
    }
    for (int i = 0; i < 18; i++) {
        if(Dest[i] == dest) {
            verdict += 1;
            break;
        }
    }
    return verdict;
}

int main()
{

    int Source[18],Destination[18],Weight[18];

    for (int i = 0; i < 18; i++) {
        scanf("%d %d %d", &Source[i],&Destination[i],&Weight[i]);
    }
    for (int i = 0; i < 18; i++) {
        int select = Weight[i];
        int index = i;
        for (int j = i; j < 18; j++) {
            if(select > Weight[j]) {
                select = Weight[j];
                index = j;
            }
        }
        int temp;
        temp = Weight[i];
        Weight[i] = Weight[index];
        Weight[index] = temp;
```

```
        temp = Source[i];
        Source[i] = Source[index];
        Source[index] = temp;

        temp = Destination[i];
        Destination[i] = Destination[index];
        Destination[index] = temp;
    }
    for (int i = 0; i < 18; i++) {
        printf("%d %d %d\n", Source[i],Destination[i],Weight[i]);
    }
    cout << "\n";

    int finalSource[100],finalDest[100],finalWeight[100];
    int fIndex = 0;
            for (int i = 0; i < 18; i++) {
                if(disjoint(finalSource, finalDest,Source[i],Desti
                    finalSource[i] = Source[i];
                    finalDest[i] = Destination[i];
                    finalWeight[i] = Weight[i];
                } else {
                    finalSource[i] = 0;
                    finalDest[i] = 0;
                    finalWeight[i] = 0;
                };
    }
            int result;
    for (int i = 0; i < 11; i++) {
        printf("%d %d %d\n", finalSource[i],finalDest[i],finalWeig
        result  += finalWeight[i];
    }
    cout << endl;
    cout << result;



    return 0;
}
```

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

```
1 2 1
9 11 1
2 7 2
4 6 2
1 3 2
0 0 0
6 10 3
1 4 3
5 8 4
8 11 4
2 5 4
мінімальне остове дерево графа:
26
```

Завдання № 6

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

	1	2	4	5	6	37	8
1	∞	1	1	1	1	3	1
2	1	∞	1	2	1	3	3
4	1	1	∞	5	5	6	1
5	1	2	5	∞	1	5	1
6	1	1	5	1	∞	5	6
37	3	3	6	5	5	∞	1
8	1	3	1	1	6	1	∞
	1	237	4	5	6	8	
1	∞	1	1	1	1	1	
237	1	∞	1	2	1	3	
4	1	1	∞	5	5	1	
5	1	2	5	∞	1	1	
6	1	1	5	1	∞	6	
8	1	3	1	1	6	∞	

	1	4	2357	6	8
1	∞	1	1	1	1
4	1	∞	5	5	1
2357	1	5	∞	1	1
6	1	5	1	∞	6
8	1	1	1	6	∞

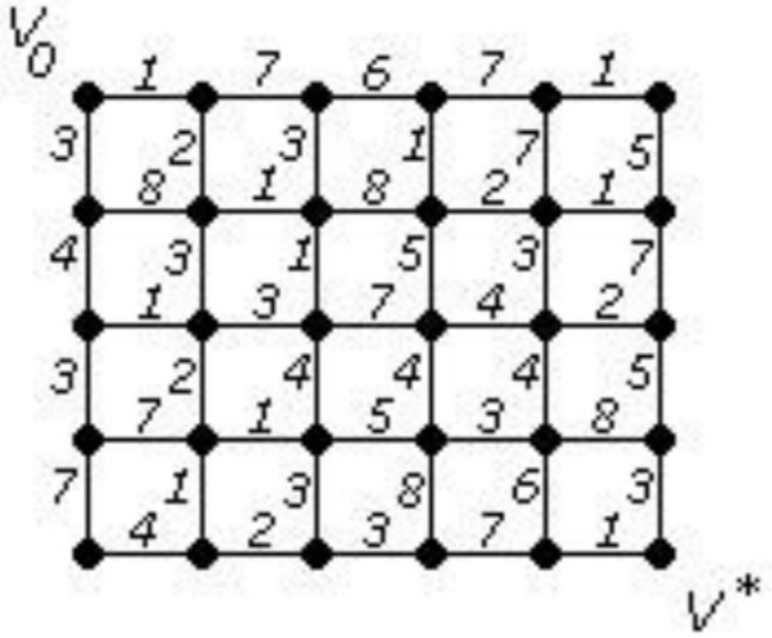
	1	23457	6	8
1	∞	1	1	1
23457	1	∞	5	1
6	1	5	∞	6
8	1	1	6	∞

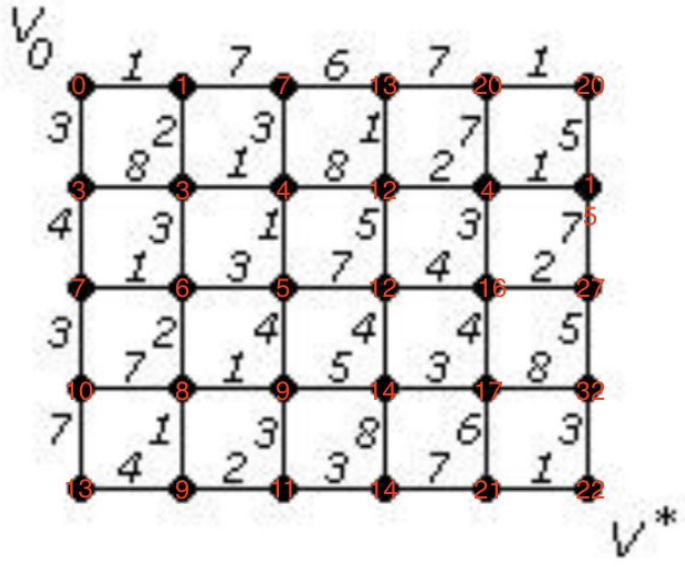
	1	234567	8
1	∞	1	1
234567	1	∞	6
8	1	6	∞

	1	2345678
1	∞	1
2345678	1	∞

Завдання № 7

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





Найкоротший шлях у графі між парою вершин V0 і V* - 22

```cpp
#include<iostream>

using namespace std;

#define INF 999

int V, start;
int pnode;
int matrix[30][30] = {
        {0,3,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {3,0,7,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,7,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,1,0,4,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,4,0,5,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,5,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {6,0,0,0,0,0,0,8,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,2,0,0,0,0,8,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,3,0,0,0,0,1,0,4,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,1,0,0,0,0,4,0,2,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,3,0,0,0,0,2,0,4,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,1,0,0,0,0,4,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,7,0,0,0,0,0,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,3,0,0,0,0,1,0,3,0,0,0,0,2,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,8,0,0,0,0,3,0,7,0,0,0,0,5,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,5,0,0,0,0,7,0,4,0,0,0,0,3,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,4,0,2,0,0,0,0,4,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,2,0,0,0,0,0,0,5,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,7,0,0,0,0,7,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,7,0,3,0,0,0,0,7,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,3,0,1,0,0,0,0,3,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,1,0,1,0,0,0,0,8},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,1,0,8,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,8,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,4,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,4,0,2,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,2,0,3},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,3,0,1},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,6,0}
};
```

```
int shortest_dist[30];
bool visited[30] = {0};
int parent[30];

int getNearest() {
    int minValue = 999, minNode = 0;
    for (int i = 0; i < V; i++) {
        if (!visited[i] && shortest_dist[i] < minValue) {
            minValue = shortest_dist[i];
            minNode = i;
        }
    }
    return minNode;

}

void dijkstra() {
    for (int i = 0; i < V; i++) {
        int nearest = getNearest();
        visited[nearest] = true;
        for (int adj = 0; adj < V; adj++) {
            if (matrix[nearest][adj] != INF && shortest_dist[adj] >
                shortest_dist[nearest] + matrix[nearest][adj]) {
                shortest_dist[adj] = shortest_dist[nearest] + matrix[nearest
                parent[adj] = nearest;
            }
        }
    }
}


int main(void) {
    V = 30;
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            if (matrix[i][j] == 0 && i != j) {
                matrix[i][j] = 999;
            }
        }
    }
```

```
    start = 0;
    for (int i = 0; i < V; i++) {
        parent[i] = i;
        shortest_dist[i] = INF;
    }
    shortest_dist[start] = 0;

    dijkstra();

    cout << "Мінімальний шлях : ";
    cout << shortest_dist[V - 1] << endl;
    cout << "Шлях : ";
    cout << endl;
    cout << V << " ";
    pnode = parent[V - 1];
    cout << endl;

    while (pnode != start) {
        cout <<  pnode + 1 << endl;
        pnode = parent[pnode];
    }
    cout << '1';
    cout << endl;

    return 0;
}
```

```
Мінімальний шлях : 22
Шлях :
30
29
23
22
21
20
14
8
2
1
```

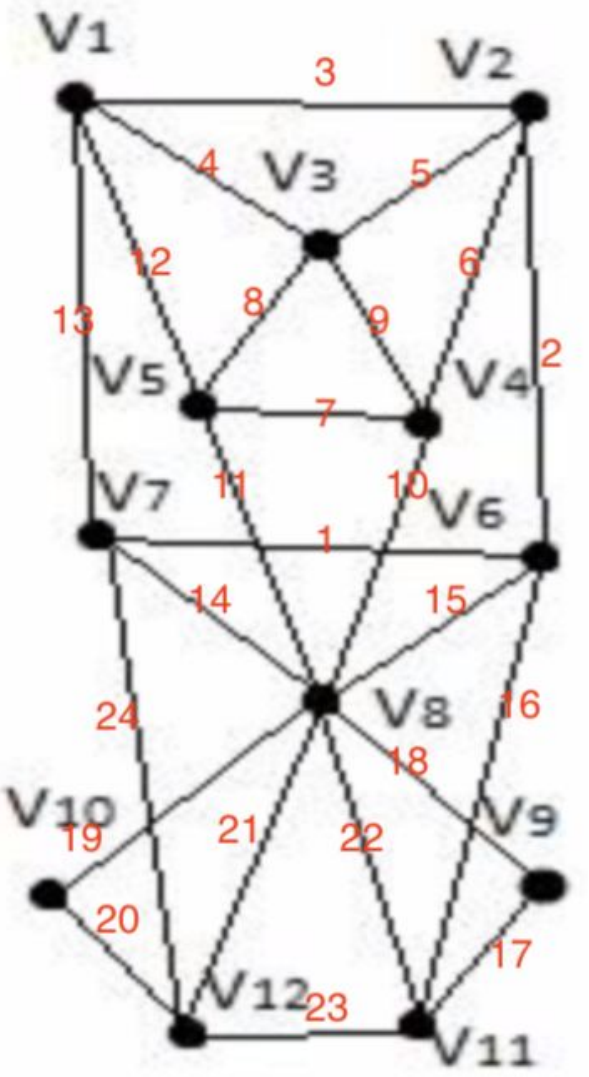
Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері;

б) елементарних циклів.

а) Флері



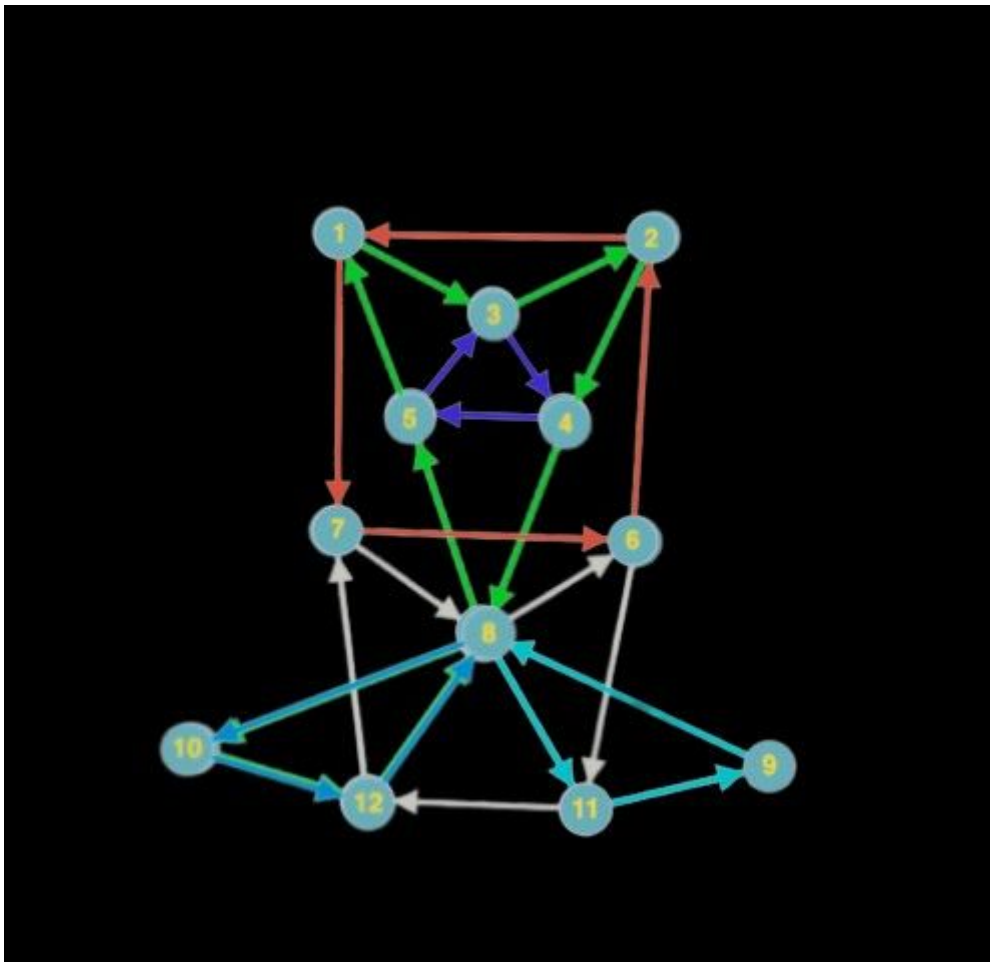
```cpp
#include<iostream>
#include<vector>
#define NODE 100
using namespace std;
int graph[NODE][NODE] = {
//  0  1  2  3  4  5  6  7  8  9  10 11
   {0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0}, // 0
   {1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0}, // 1
   {1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0}, // 2
   {0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0}, // 3
   {1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0}, // 4
   {0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1}, // 5
   {1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0}, // 6
   {0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1}, // 7
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1}, // 8
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0}, // 9
   {0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, // 10
   {0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0}, // 11
};
int tempGraph[NODE][NODE];
int findStartVert(){
   for(int i = 0; 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 fleury(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 << "--" << v << " ";
            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 << "Шлях: ";
    fleury(findStartVert());
}
```

```
Шлях:
10--7
7--3
3--1
1--0
0--2
2--1
1--5
5--6
6--0
0--4
4--2
2--3
3--4
4--7
7--5
5--11
11--7
7--6
6--10
10--9
9--7
7--8
8--11
11--10
```

б) елементарних циклів



Я виділив 6 простих циклів.

Завдання № 9

Спростити формули (привести їх до скороченої ДНФ).

14. $x\bar{y}z \vee \bar{x}\bar{z} \vee xy$

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