МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ
“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни
«Дискретна математика»

Виконав:
студент групи КН-112
Дуда Олександр
Викладач:
Мельникова Н.І.

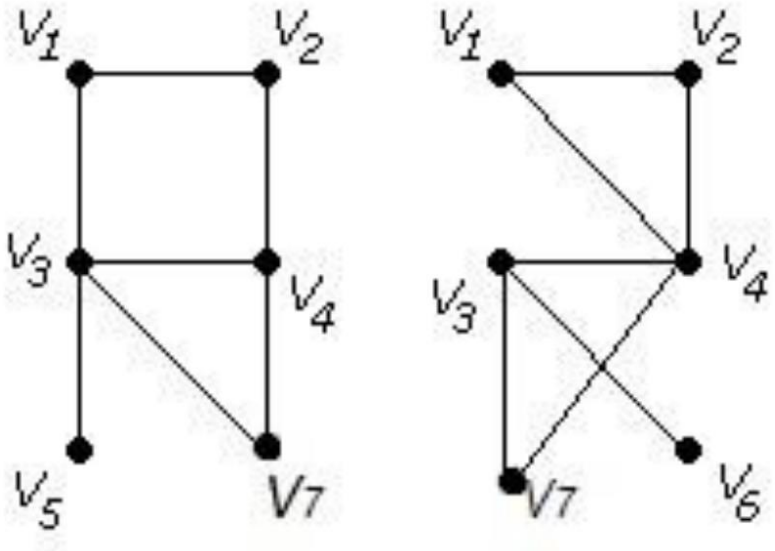
Львів-2019р.

Варіант №13

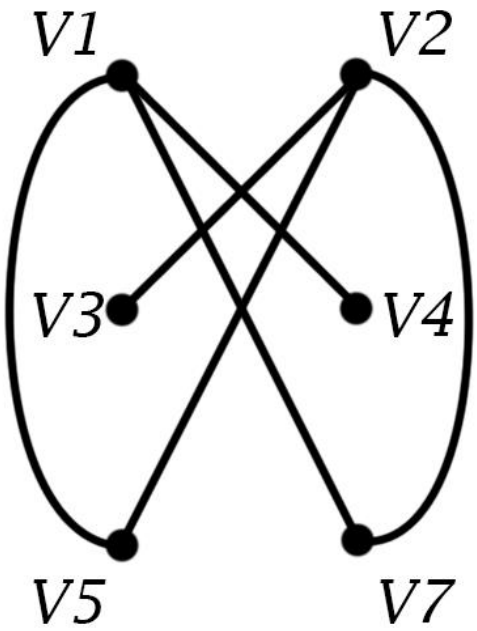
Завдання 1.

Виконати наступні операції над графами:

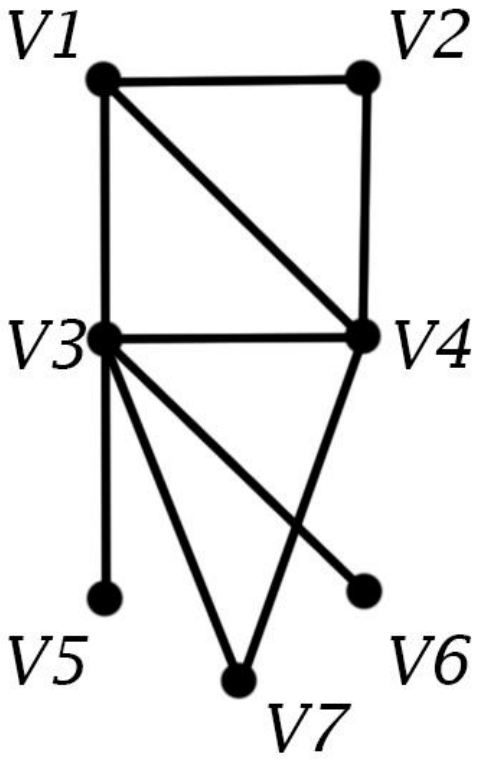
1) Знайти доповнення до першого графу;
2) Об’єднання графів;
3) Кільцеву суму G1 та G2 (G1+G2);
4) Розмножити вершину у другому графі;
5) Виділити підграф А - що складається з 3-х вершин в G1;
6) Добуток графів;



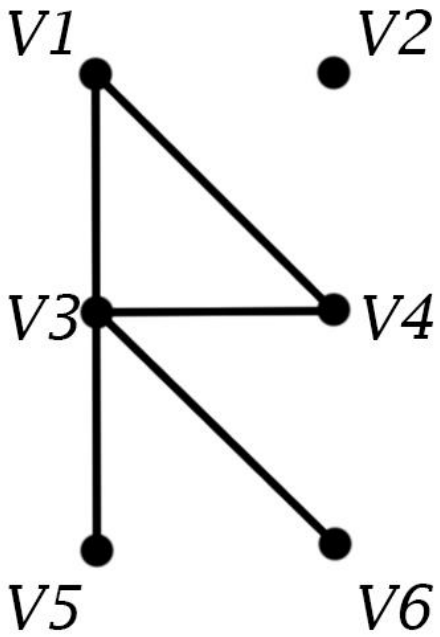
1)Доповнення до першого графу:



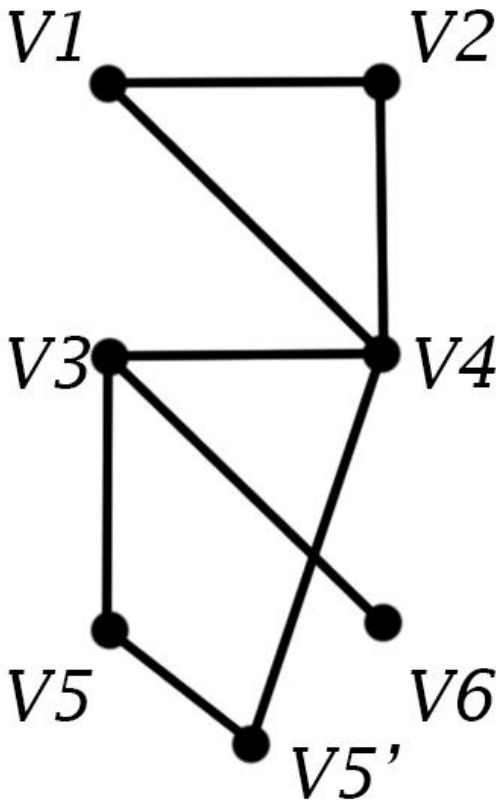
2) Об’єднання графів:



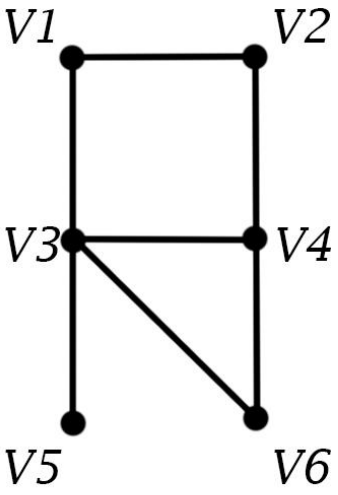
3) Кільцева сума G1 та G2 (G1+G2):

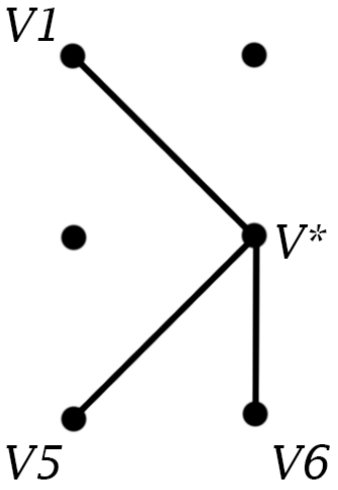


4) Розмножити вершину у другому графі:

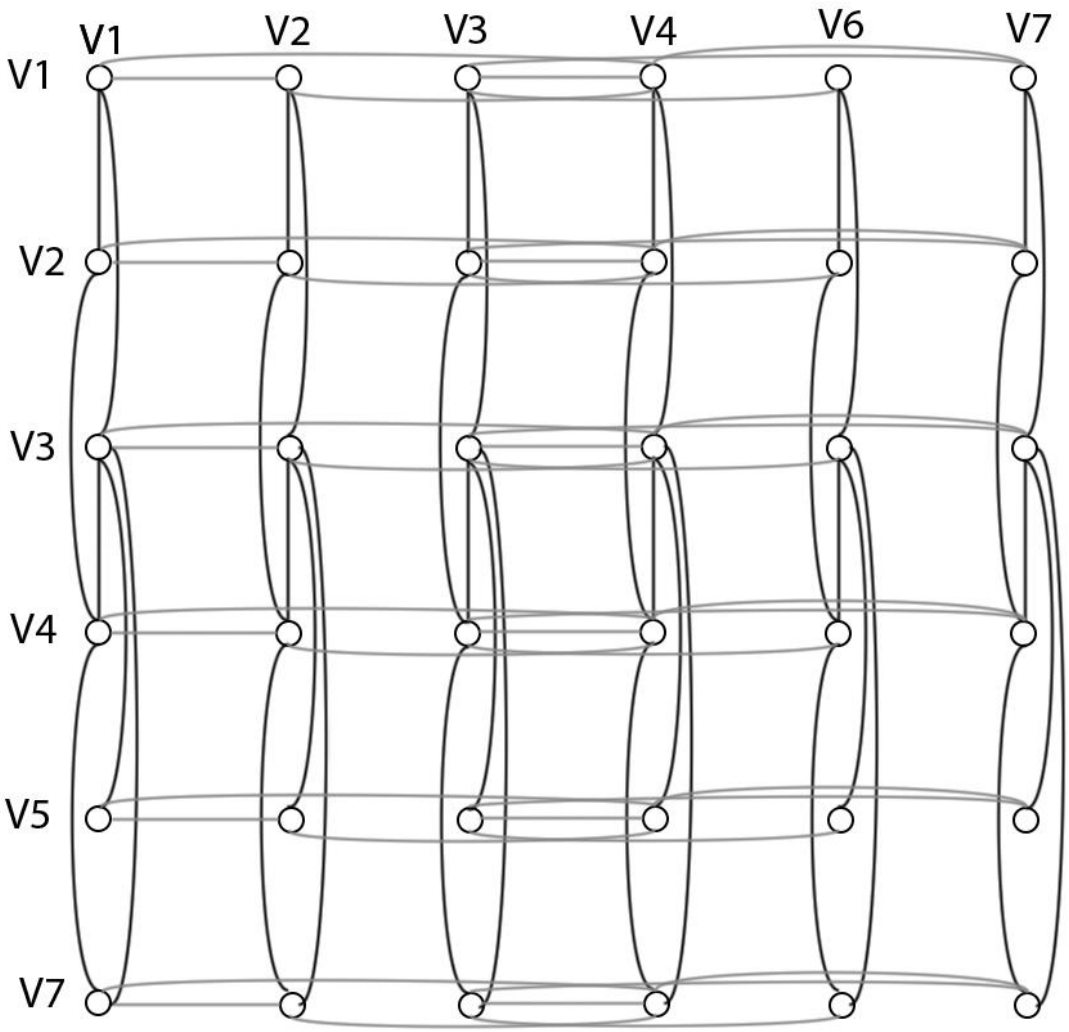


5) Виділити підграф А - що складається з 3-х вершин в G1:



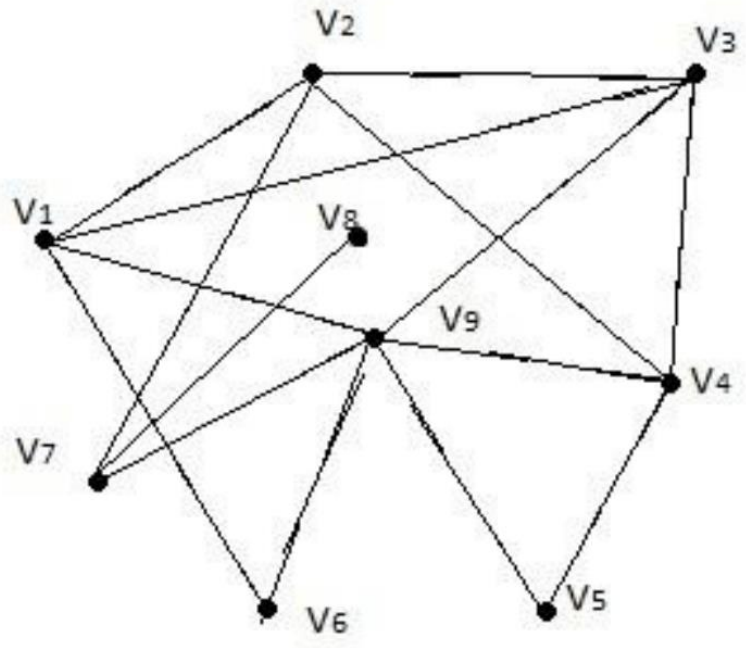


6) Добуток графів:



Завдання 2.

Скласти таблицю суміжності для орграфа:



Таблиця суміжності

	1	2	3	4	5	6	7	8	9
1	-	1	1	2	2	1	2	3	1
2	1	-	1	1	3	2	1	2	2
3	1	1	-	1	2	2	2	3	1
4	2	1	1	-	1	2	2	3	1
5	2	3	2	1	-	2	2	3	1
6	1	2	2	2	2	-	2	3	1
7	2	1	2	2	2	2	-	1	1
8	3	2	3	3	3	3	1	-	2
9	1	2	1	1	1	1	1	2	-

Завдання 3

Для графа з другого завдання знайти діаметр:

Діаметр графа рівний 3.

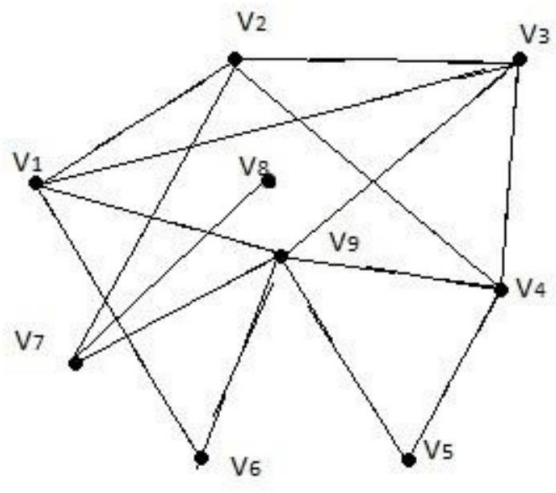
3

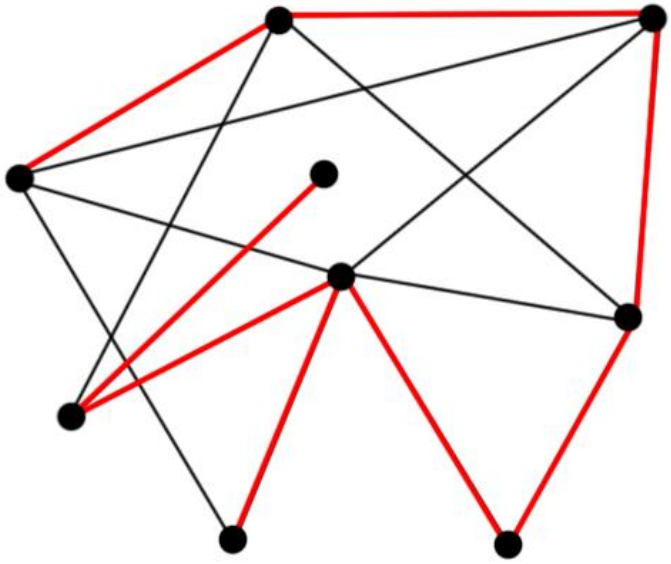
Завдання 4

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

Обхід графа в глиб:

V	№	Стек
1	1	V1
2	2	V1V2
3	3	V1V2V3
4	4	V1V2V3V4
5	5	V1V2V3V4V5
9	6	V1V2V3V4V5V9
6	7	V1V2V3V4V5V9V6
-	-	V1V2V3V4V5V9
7	8	V1V2V3V4V5V9V7
8	9	V1V2V3V4V5V9V7V8
-	-	V1V2V3V4V5V9
-	-	V1V2V3V4V5
-	-	V1V2V3V4
-	-	V1V2V3
-	-	V1V2
-	-	V1V2
-	-	V1
		Ø



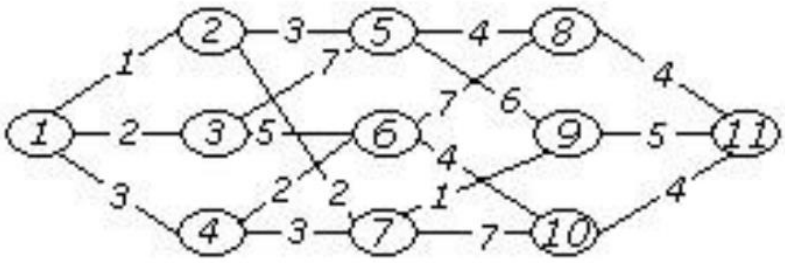


```cpp
#include <iostream>
using namespace std;
const int n = 9;
bool* visited = new bool[n];
int graph[n][n] =
{
	{ 0, 1, 1, 0, 0, 1, 0, 0, 1},
	{ 1, 0, 1, 1, 0, 0, 1, 0,0 },
	{ 1, 1, 0, 1, 0, 0, 0, 0, 1},
	{ 0, 1, 1, 0, 1, 0, 0, 0, 1},
	{ 0, 0, 0, 1, 0, 0, 0, 0, 1},
	{ 1, 0, 0, 0, 0, 0, 0, 0, 1},
	{ 0, 1, 0, 0, 0, 0, 0, 1, 1},
	{ 0, 0, 0, 0, 0, 0, 1, 0, 0},
	{ 1, 0, 1, 1, 1, 1, 1, 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()
{
	setlocale(LC_ALL, "Rus");
	int start;
	int i, j;
	cout << "Матриця суміжності графа: " << endl;
	for (i = 0; i < n; i++)
	{
		visited[i] = false;
		for (j = 0; j < n; j++)
			cout << " " << graph[i][j];
		cout << endl;
	}
	cout << "Стартова вершина >> "; cin >> start;

	bool* vis = new bool[n];
	cout << "Порядок обходу: ";
	DFS(start - 1);
	delete[]visited;
}
```

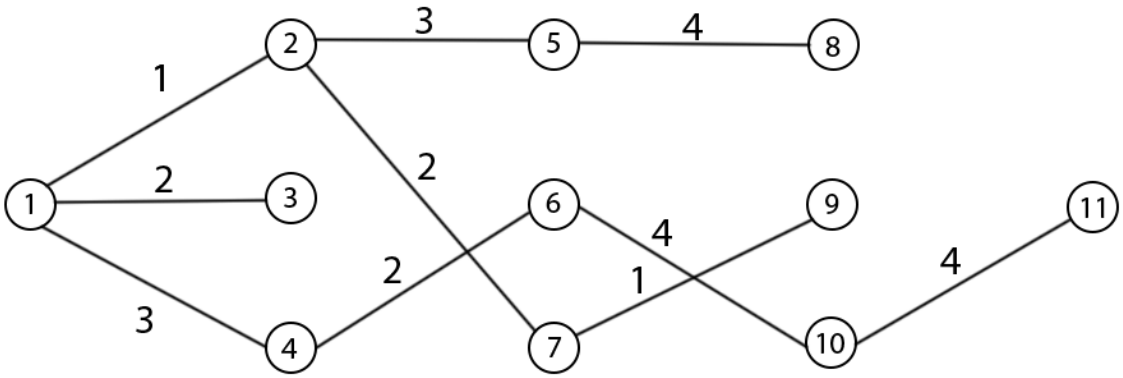
```
Матриця суміжності графа:
 0 1 1 0 0 1 0 0 1
 1 0 1 1 0 0 1 0 0
 1 1 0 1 0 0 0 0 1
 0 1 1 0 1 0 0 0 1
 0 0 0 1 0 0 0 0 1
 1 0 0 0 0 0 0 0 1
 0 1 0 0 0 0 0 1 1
 0 0 0 0 0 0 1 0 0
 1 0 1 1 1 1 1 0 0
Стартова вершина >> 1
Порядок обходу: 1 2 3 4 5 9 6 7 8
```

Завдання № 5

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



Краскала:



V={1,2,7,9,3,4,6,5,8,10,11}
E ={(1,2),(7,9),(1,3),(4,6),(2,7),(1,4),(2,5),(5,8),(6,10),(10,11)}
Вага остового дерева: 26

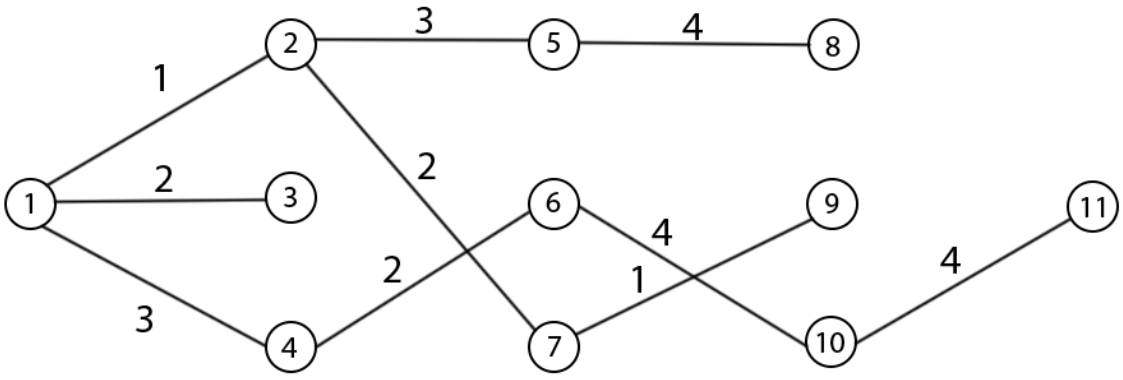
```
#include <iostream>
#include <cstdio>
using namespace std;

int create(int n, int A[11][11]) {
	for (int i = 0; i < 11; i++) {
		for (int j = 0; j < 11; j++) {
			A[i][j] = 0;
		}
	}
	for (int i = 0; i < 11; i++) {
		A[i][i] = i + 1;
	}
	return A[11][11];
}
void RemoveDuplicats(int n, int A[11][11]) {
	for (int i = 0; i < 11; i++) {
		for (int j = 0; j < 11; j++) {
			if (j < i) {
				A[i][j] = 0;
			}
		}
```

```
	}
	for (int i = 0; i < 11; i++) {
		for (int j = 0; j < 11; j++) {
			cout << A[i][j] << " ";
		}
		cout << endl;
	}
}
int NotOne(int n, int A[11][11],int f, int s) {
	int tmp,tmp1;
	for (int i = 0; i < 11; i++) {
		tmp = tmp1 = 0;
		for (int j = 0; j < 11; j++) {
			if (A[i][j] == f) {
				tmp = 1;
			}
		}
		for (int k = 0; k < 11; k++) {
			if (A[i][k] == s) {
				tmp1 = 1;
			}
		}
		if (tmp && tmp1) {
			return 0;
		}
	}


	return 1;
}

void add(int n, int A[11][11], int f, int s) {
	int tmp;
	for (int i = 0; i < 11; i++) {
		for (int j = 0; j < 11; j++) {
			if (A[i][j] == s) {
				tmp = i;
			}
		}
	}
	for (int i = 0; i < 11; i++) {
		for (int j = 0; j < 11; j++) {
			if (A[i][j] == f) {
				for (int k = 0; k < 11; k++) {
				if (A[tmp][k]) {
					A[i][k] = A[tmp][k];
					A[tmp][k] = 0;
					}
				}
			}
		}
	}

	}

int main()
{
	int MS[11][11];
	for (int i = 0; i < 11; i++) {
		for (int j = 0; j < 11; j++) {
			cin >> MS[i][j];
		}
	}

	RemoveDuplicats(11, MS);
	for (int i = 1; i <= 7; i++) {
		cout << endl << "Edges with weight  " << i <<": ";
		for (int j = 1; j <= 11; j++) {
```

```
                        for (int k = 1; k <= 11; k++) {
                                if (MS[j - 1][k - 1] == i) {
                                        cout << " (" << j << ";" << k << ") ";
                                }
                        }
                }
                cout << endl;
        }

        int B[11][11];
        create(11, B);
        cout << endl << endl << "New Tree:" << endl;
                for (int i = 1; i <= 7; i++) {

                        for (int j = 1; j <= 11; j++) {
                                for (int k = 1; k <= 11; k++) {
                                        if (MS[j - 1][k - 1] == i && NotOne(11,B,j,k)){
                                                add(11, B, j, k);
                                                cout << " (" << j << ";" << k << ") ";
                                                cout << endl;
                                        }

                                }
                        }
                }

                cout << endl;

                return 0;
}
```

```
Edges with weight  1:  (1;2)  (7;9)

Edges with weight  2:  (1;3)  (2;7)  (4;6)

Edges with weight  3:  (1;4)  (2;5)  (4;7)

Edges with weight  4:  (5;8)  (6;10)  (8;11)  (10;11)

Edges with weight  5:  (9;11)

Edges with weight  6:  (3;6)  (5;9)

Edges with weight  7:  (3;5)  (6;8)  (7;10)


New Tree:
 (1;2)
 (7;9)
 (1;3)
 (2;7)
 (4;6)
 (1;4)
 (2;5)
 (5;8)
 (6;10)
 (8;11)
```

Прима:



V={1,2,3,7,9,4,5,8,10,11}
E ={(1,2),(1,3),(2,7),(7,9),(1,4),(4,6),(2,5),(5,8),(6,10),(10,11)}
Вага остового дерева: 26

```cpp
#include <iostream>

using namespace std;

int main() {
	int v, count = 0, min = 0, k, t;
	bool check = false;
	cout << "Enter quantity of vertex : ";
	cin >> v;
	int* tops = new int[v];
	int** g = new int* [v];
	int** r = new int* [v - 1];

	for (int j = 0; j < v; j++) {
		g[j] = new int[v];
	}
	for (int j = 0; j < v - 1; j++) {
		r[j] = new int[2];
	}
	for (int i = 0; i < v; i++) {
		for (int j = 0; j < v; j++) {
			cin >> g[i][j];
		}
	}
	tops[count] = 1;
	count++;
	for (int i = 0; count < v; i++) {
		for (int j = 0; j < count; j++) {
			for (int a = 0; a < v; a++) {
				for (int m = 0; m < count; m++) {
					if (tops[m] == a + 1) {
						check = true;
					}
				}
				if (check) {
					check = false;
					continue;
				}
				if (min == 0 && g[tops[j] - 1][a] > 0) {
					min = g[tops[j] - 1][a];
					k = r[count - 1][0] = tops[j];
					t = r[count - 1][1] = a + 1;
					continue;
				}
				if (g[tops[j] - 1][a] > 0 && g[tops[j] - 1][a] < min) {
					min = g[tops[j] - 1][a];
					k = r[count - 1][0] = tops[j];
					t = r[count - 1][1] = a + 1;
				}
			}
		}
		g[k - 1][t - 1] = 0;
		g[t - 1][k - 1] = 0;

		tops[count] = t;
		count++;
		min = 0;
	}

	cout << "V: { ";
	for (int j = 0; j < v; j++) {
		cout << tops[j] << ", ";
	}
	cout << "}";
	cout << endl << "E:{ ";
	for (int j = 0; j < v - 1; j++) {
```

```
                cout << "( " << r[j][0] << ", " << r[j][1] << " ), ";
        }
        cout << "}";


        delete[] tops;
        for (int i = 0; i < v; i++)
                delete[] g[i];
        for (int i = 0; i < v; i++)
                delete[] r[i];

        return 0;
}
```

```
0 1 2 3 0 0 0 0 0 0 0
1 0 0 0 3 0 2 0 0 0 0
1 0 0 0 7 6 0 0 0 0 0
3 0 0 0 0 2 3 0 0 0 0
0 3 7 0 0 0 0 4 6 0 0
0 0 5 2 0 0 0 7 0 4 0
0 2 0 3 0 0 0 0 1 7 0
0 0 0 0 4 7 0 0 0 0 4
0 0 0 0 6 0 1 0 0 0 5
0 0 0 0 0 4 7 0 0 0 4
0 0 0 0 0 0 0 4 5 4 0
V: { 1, 2, 3, 7, 9, 4, 6, 5, 10, 8, 11, }
E:{ ( 1, 2 ), ( 1, 3 ), ( 2, 7 ), ( 7, 9 ), ( 1, 4 ), ( 4, 6 ), ( 2, 5 ), ( 6, 10 ), ( 5, 8 ), ( 10, 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	∞	1	5	1	5	1	6	1
2	1	∞	7	5	6	1	2	3
3	5	7	∞	5	6	2	1	2
4	1	5	5	∞	6	5	1	5
5	5	6	6	6	∞	7	7	7
6	1	1	2	5	7	∞	1	1
7	6	2	1	1	7	1	∞	2
8	1	3	2	5	7	1	2	∞

1) 1=>2=>6=>7=>3=>8=>4=>5=>1{22}
2) 2=>1=>8=>6=>7=>3=>4=>5=>2{22}
3) 3=>7=>4=>1=>8=>6=>2=>5=>3{18}
4) 4=>1=>6=>7=>3=>8=>2=>5=>4{21}
5) 5=>1=>8=>6=>2=>7=>4=>3=>5{22}
6) 6=>7=>4=>1=>8=>3=>5=>2=>6{19}
7) 7=>3=>8=>6=>1=>4=>2=>5=>7{24}
8) 8=>6=>2=>1=>4=>7=>3=>5=>8{19}

```
#include<iostream>
#include<vector>
using namespace std;
int counter = 0, Inf = 9999;

bool check(vector<int> q, int Node);

int F_Min(vector<int>* q, int** arr, int n, int i);

void Find(vector<int>* q, int** arr, int n, int pos, vector<int>* qq);

int main()
{
	int n;
	cin >> n;
	int** arr = new int* [n];
	for (int i = 0; i < n; i++) {
		arr[i] = new int[n];
	}

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

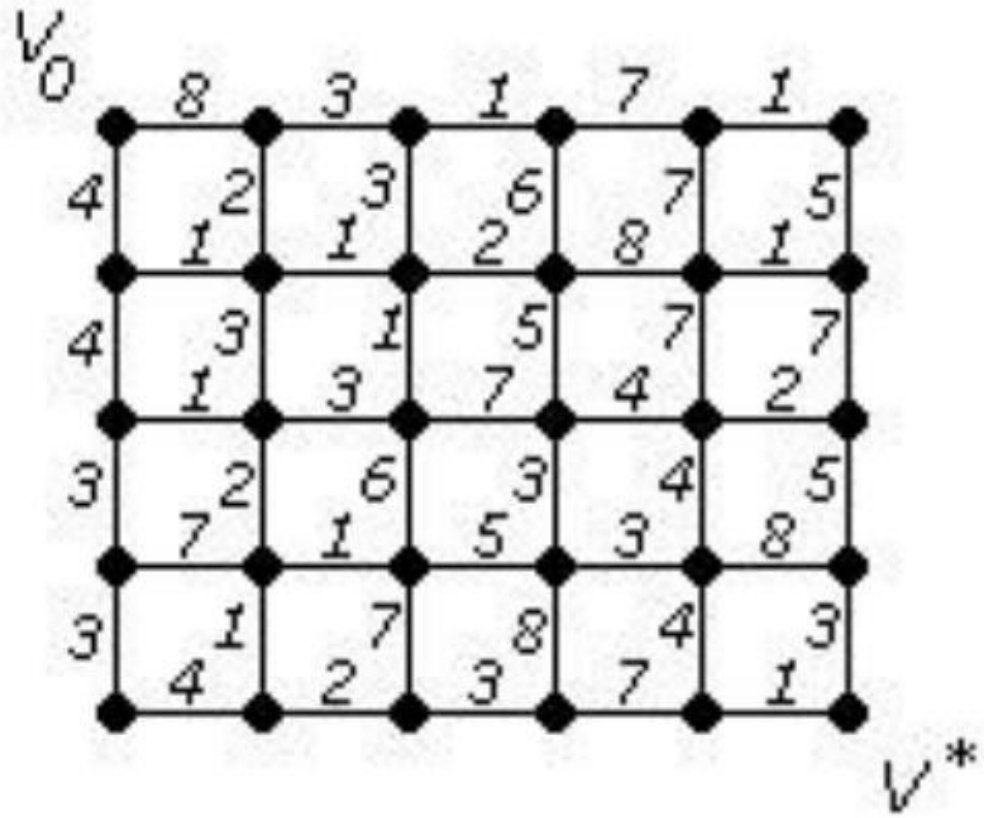
```
			for (int j = 0; j < n; j++) {
				cin >> arr[i][j];
			}
		}
		vector<int> q;
		vector<int> qq;
		cout << endl;

		for (int i = 0; i < n; i++) {
			q.clear();
			q.push_back(i);
			Find(&q, arr, n, i, &qq);
		}
		for (int i = 1; i <= qq.size(); i++) {
			if (i != 0 && i % (n + 2) == 0)
				cout << " {" << qq[i - 1] << "}" << endl;
			else
				cout << qq[i - 1] + 1 << " ";
		}
		return 0;
}

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


int F_Min(vector<int>* q, int** arr, int n, int i) {
		int min = 999;
		for (int j = 0; j < n; j++) {
			if (arr[i][j] < min && arr[i][j] != 0 && check((*q), j))min = arr[i][j];
		}
		return min;
}

void Find(vector<int>* q, int** arr, int n, int pos, vector<int>* qq) {
		int min;
		for (int i = pos, k = 0; k < 1; i++, k++) {
			min = F_Min(q, arr, n, i);
			for (int j = 0; j < n; j++) {
				if (arr[i][j] == min && check((*q), j)) {
					(*q).push_back(j);
					Find(q, arr, n, j, qq);
				}
			}
			if (q->size() == n) {
				(*q).push_back((*q)[0]);
				counter = 0;

				for (int l = 1; l <= n; l++) {
					counter += arr[(*q)[l - 1]][(*q)[l]];
				}

				if (Inf == counter) {
					for (int b = 0; b <= n; b++) {
						(*qq).push_back((*q)[b]);
					}
					(*qq).push_back(counter);
				}

				else if (Inf > counter) {
					(*qq).clear();

					for (int b = 0; b <= n; b++) {
						(*qq).push_back((*q)[b]);
```

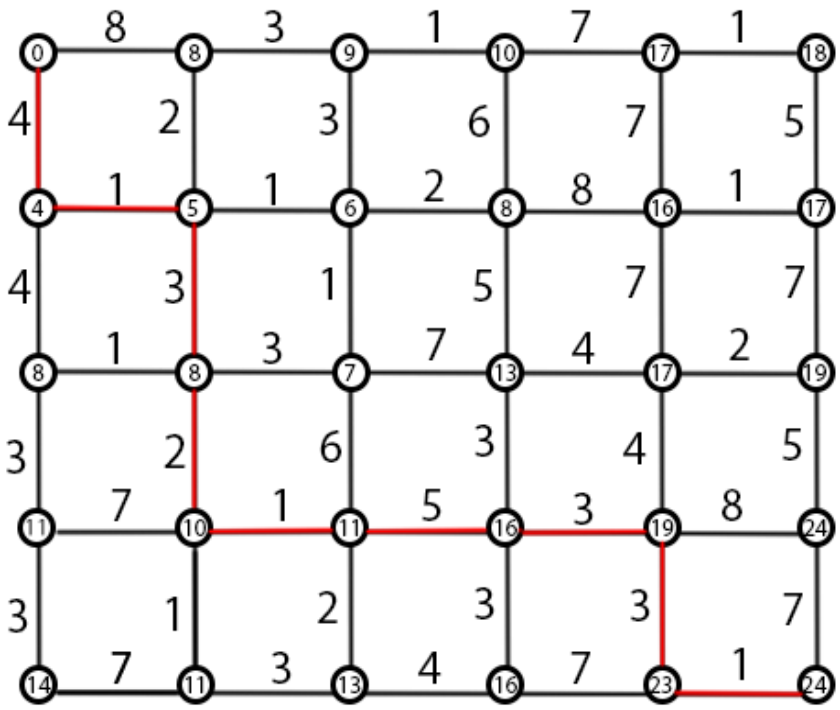
```
                        }
                        (*qq).push_back(counter);

                        Inf = counter;
                    }
                    q->pop_back();
                }
            }
            q->pop_back();
}
```

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

1 2 6 8 3 7 4 5 1  {18}
1 4 7 3 8 6 2 5 1  {18}
2 6 8 1 4 7 3 5 2  {18}
3 7 4 1 8 6 2 5 3  {18}
4 7 3 8 6 2 1 5 4  {18}
5 1 2 6 8 3 7 4 5  {18}
5 1 4 7 3 8 6 2 5  {18}
6 8 1 4 7 3 5 2 6  {18}
7 4 1 8 6 2 5 3 7  {18}
```

Завдання № 7.

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






```
#include <iostream>
#define inf 1000000
using namespace std;

int min_top(int** arr, int v) {
	int m = 0;
	for (int i = 0; i < v; i++) {
		if (arr[i][1]) {
			m = i; break;
		}
	}

	for (int i = 1; i < v; i++) {
		if (arr[m][0] >= arr[i][0] && arr[i][1] == 1) {
			m = i;
		}
	}
	return m;
}

int main()
{
	setlocale(LC_ALL, "Ukrainian");

	int a, b, c;
	int v = 30;
	int** graph = new int* [v];

	for (int j = 0; j < v; j++) {
		graph[j] = new int[v];
	}

	for (int a = 0; a < v; a++) {
```

```
	for (int j = 0; j < v; j++) {
		graph[a][j] = 0;

	}
}
cout << "\nВведіть вагу ребер графа : " << endl;

while(true){
	cin >> a;
	if (a == 111) {
		break;
	}
	cin >> b;
	cin >> c;
	graph[a - 1][b - 1] = graph[b - 1][a - 1] = c;
}
for (int i = 0; i < v; i++)
{
	for (int j = 0; j < v; j++) {
		cout << graph[i][j] << " ";
	}
	cout << endl;
}
int p;
int** tops = new int* [v];
for (int j = 0; j < v; j++) {
	tops[j] = new int[2];
}
int* tops_path = new int[v];

cout << "Вихідна вершина: ";
cin >> p;
//надає вершинам значення 100000 крім тої з якої вказано почати
for (int i = 0; i < v; i++) {
	if (i == p - 1) {
		tops[i][0] = 0;
		tops[i][1] = 1;
	}
	else {
		tops[i][0] = inf;
		tops[i][1] = 1;
	}
}
tops_path[p - 1] = 0;
int m;

for (int i = 0; i < v; i++) {
	m = min_top(tops, v);
	for (int j = 0; j < v; j++) {
		if (graph[m][j]) {
			if (tops[j][0] > tops[m][0] + (graph[m][j])) {
				tops[j][0] = tops[m][0] + (graph[m][j]);
				tops_path[j] = m;
			}
		}

	}
	tops[m][1] = 0;

}
////шлях
cout << "Введіть потрібну вершину: ";
int k; cin >> k;
cout << "Мінімальний шлях: ";
cout << tops[k - 1][0];
cout << endl << k << " <-- ";
k--;
for (int a = 0; tops_path[k] != p - 1; a++) {
```

```
            cout << tops_path[k] + 1 << " <-- ";
            k = tops_path[k];
        }
        cout << p << endl;


        return 0;
}
```

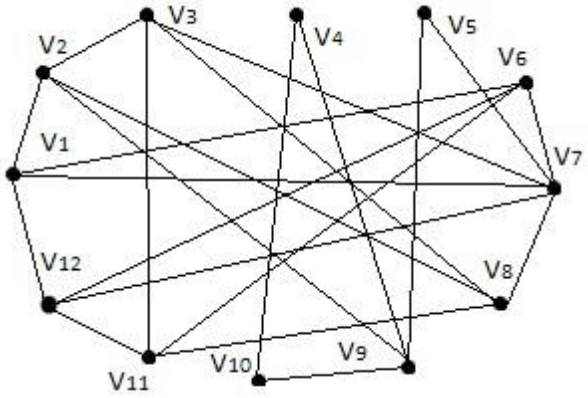
```
Вихідна вершина: 1
Введіть потрібну вершину: 30
Мінімальний шлях: 24
30 <-- 29 <-- 28 <-- 27 <-- 26 <-- 20 <-- 14 <-- 8 <-- 7 <-- 1
```

Завдання № 8.

Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері;

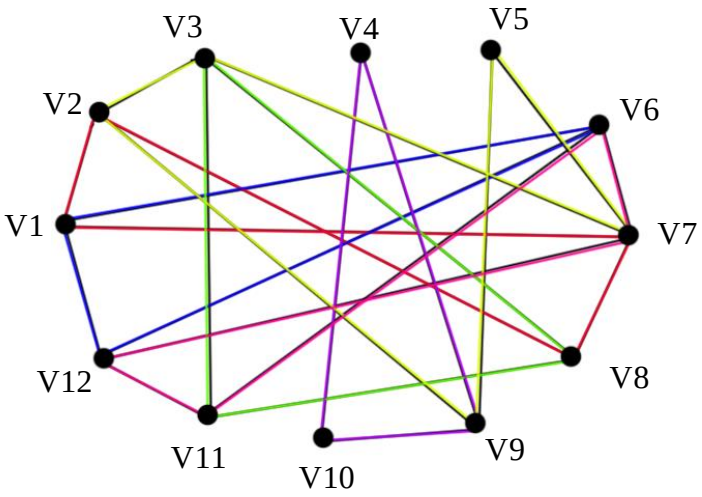
б) елементарних циклів.



а) Флері:

1=>2=>3=>7=>8=>2=>9=>10=>4=>9=>5=>7=>1=>6=>12=>11=>3=>8=>11=>6=>7=>12=>1

б) Елементарні цикли.



Елементарні цикли:

1)2=>3=>7=>5=>9=>2

2)1=>2=>8=>7=>1

3)7=>6=>11=>12=>7

3)6=>12=>1=>6

4)3=>11=>8=>3

5)9=>4=>10=>9

1=>2=>3=>11=>8=>3=>7=>6=>12=>1=>6=>11=>12=>7=>5=>9=>4=>10=>9=>2=>8=>7=>1

Реалізація Флері:

```
#include <iostream>
#include <vector>
#include <stack>
using namespace std;
int main()
{
	setlocale(LC_ALL, "Ukrainian");
	int v = 0;
	cout << "Кількість вершин графа : ";
	cin >> v;
	cout << "Введіть матрицю суміжності:" << endl;
	int** graph = new int* [v];
	for (int j = 0; j < v; j++) {
		graph[j] = new int[v];
	}
	for (int a = 0; a < v; a++) {
		for (int j = 0; j < v; j++) {
			cin >> graph[a][j];
		}
	}
	vector<int> Stack;
	vector<int> path;
	int m, ver;
	Stack.push_back(1);
	while (!Stack.empty()) {
		m = 0;
		ver = Stack[Stack.size() - 1];
		for (int i = 0; i < v; i++) {
			if (graph[ver - 1][i]) {
				m = i + 1;
				graph[ver - 1][i] = 0;
				graph[i][ver - 1] = 0;
				Stack.push_back(m);
				break;
			}
		}
		if (m == 0) {
			path.push_back(ver);
			Stack.pop_back();
		}

	}
	for (int i = path.size() - 1; i > 0; i--) {
```

```
            cout << path[i] << "->";
        }
        cout << path[0];
        return 0;
}
```

```
Кількість вершин графа : 12
Введіть матрицю суміжності:
0 1 0 0 0 1 1 0 0 0 0 1
1 0 1 0 0 0 0 1 1 0 0 0
0 1 0 0 0 0 1 1 0 0 1 0
0 0 0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 1 0 1 0 0 0
1 0 0 0 0 0 1 0 0 0 1 1
1 0 1 0 1 1 0 1 0 0 0 1
0 1 1 0 0 0 1 0 0 0 1 0
0 1 0 1 1 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 1 0 0 0
0 0 1 0 0 1 0 1 0 0 0 1
1 0 0 0 0 1 1 0 0 0 1 0
1->2->3->7->1->6->7->5->9->4->10->9->2->8->3->11->6->12->7->8->11->12->1
```

Завдання №9.
Спростити формулу (привести їх до скороченої ДНФ):

$x\bar{y} \vee x\bar{z} \vee z$

x	y	z	$\bar{y}$	$x\bar{y}$	$\bar{z}$	$x\bar{z}$	$x\bar{y} \vee x\bar{z}$	$x\bar{y} \vee x\bar{z} \vee z$
0	0	0	1	0	1	0	0	0
0	0	1	1	0	0	0	0	1
0	1	0	0	0	1	0	0	0
0	1	1	0	0	0	0	0	1
1	0	0	1	1	1	1	1	1
1	0	1	1	1	0	0	1	1
1	1	0	0	0	1	1	1	1
1	1	1	0	0	0	0	0	1

Карти Карно:

x\yz	*00*	*01*	*11*	*10*
0	0	1	1	0
1	1	1	1	1

СДНФ

$x \vee z$