

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни:

“Дискретна математика”

Виконав:

Студент групи КН-113

Вовчак Л. В.

Викладач:

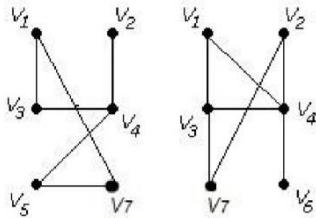
Мельникова Н.І.

Теоретичні питання

Варіант 17

Завдання 1.

Дано два графи:

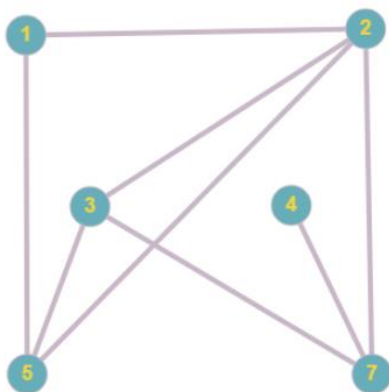


Розв'язати на графах наступні задачі:

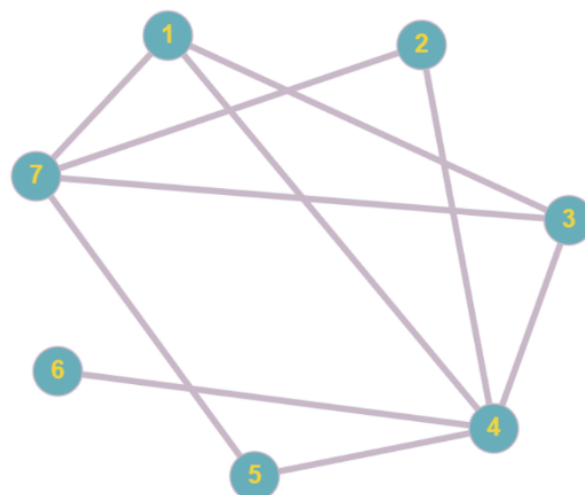
1. знайти доповнення до першого графу.
2. виконати об'єднання графів
3. знайти кільцеву суму $G1$ та $G2$
4. розщепити вершину у другому графі
5. виділити підграф A , що складається з 3-х вершин в $G1$ і знайти стягнення A в $G1$
6. добуток графів

Розв'язання:

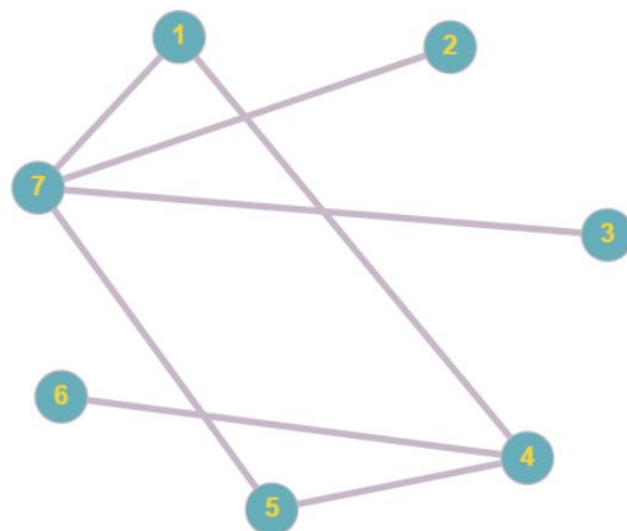
1. Доповнення до першого графу:



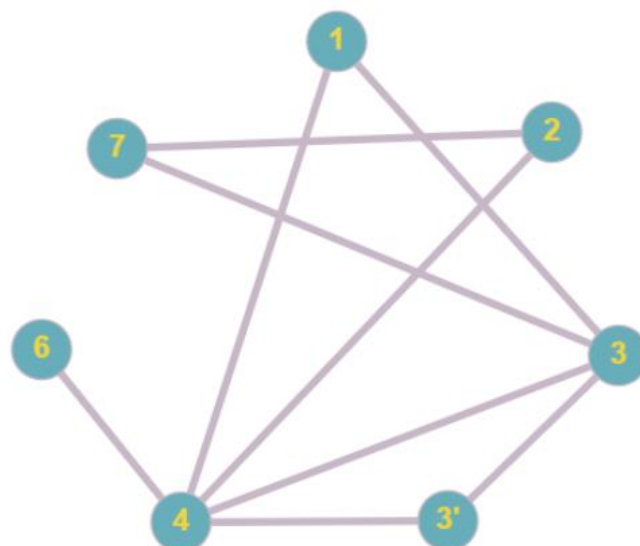
2. Об'єднання графів:



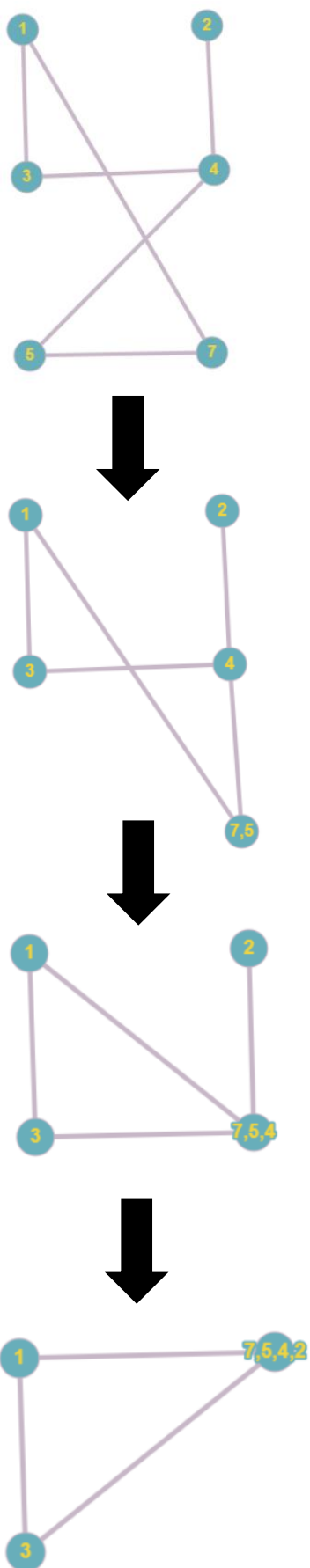
3. Кільцева сума G_1 та G_2 :



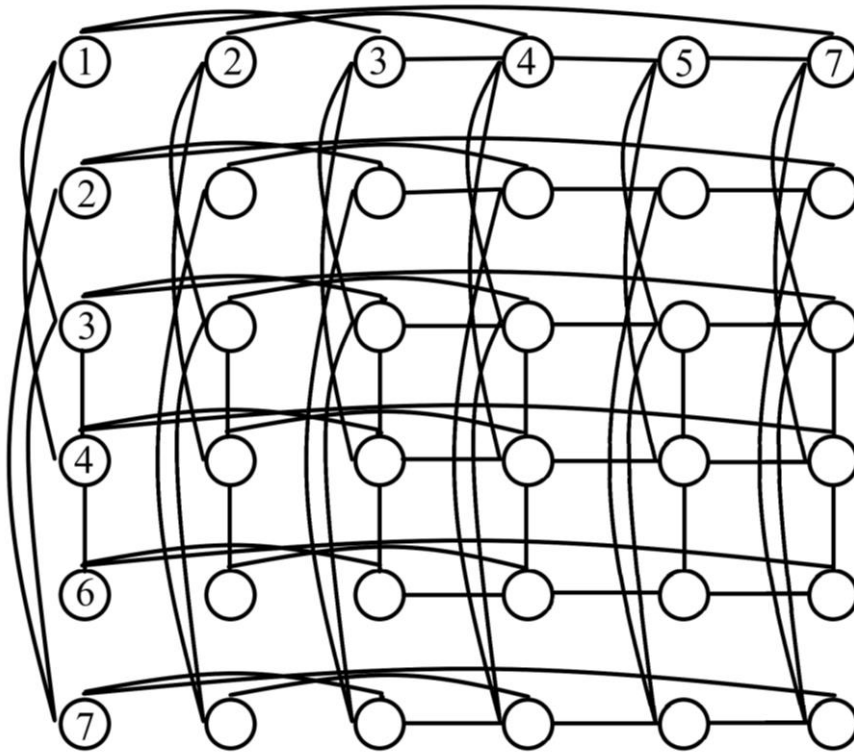
4. Розщеплення вершини 3 у другому графі:



5. Виділення підграфа $A = \{V_4, V_7, V_5\}$, що складається з 3-х вершин в G_1 і знайти стягнення A в G_1

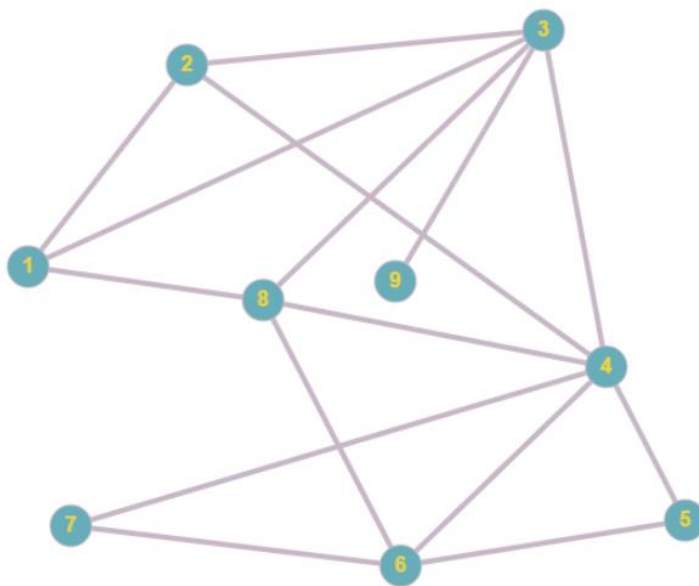


6. Множення графів:



Завдання 2.

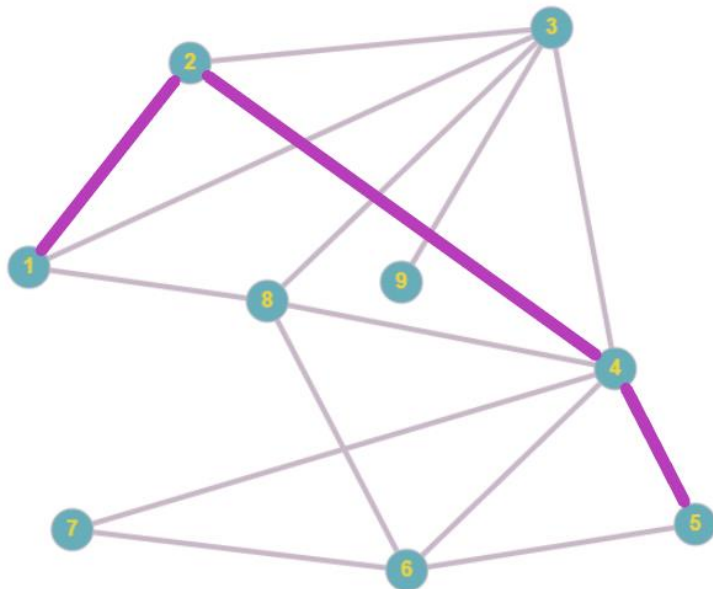
Скласти таблицю суміжності для орграфа



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

Завдання 3.

Для графа з другого завдання знайти діаметр

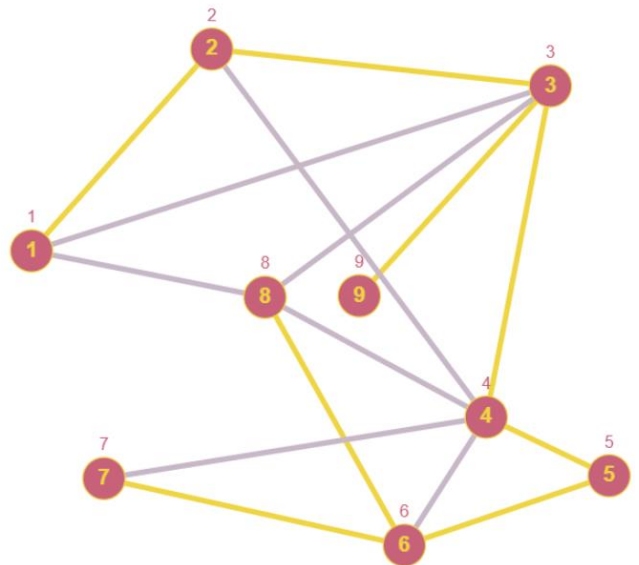


Діаметр = 3

Завдання 4.

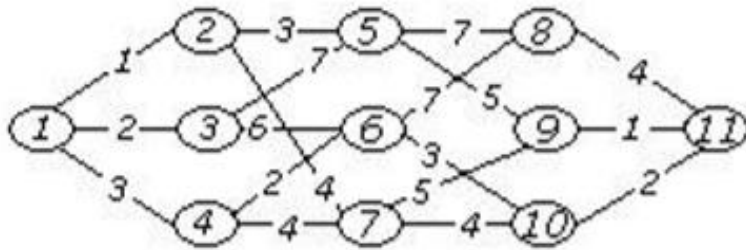
Для графа з другого завдання виконати обхід дерева вглиб
(варіант закінчується на непарне число)

1	1	1
2	2	12
3	3	123
4	4	1234
5	5	12345
6	6	123456
7	7	1234567
8	-	123456
9	8	1234568
10	-	123456
11	-	12345
12	-	1234
13	-	123
14	9	1239
15	-	123
16	-	12
17	-	1
18	-	

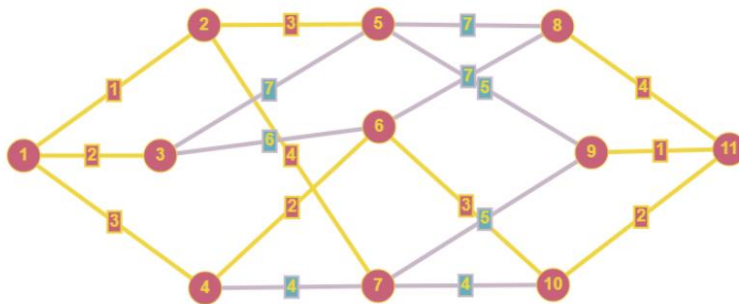


Завдання 5.

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



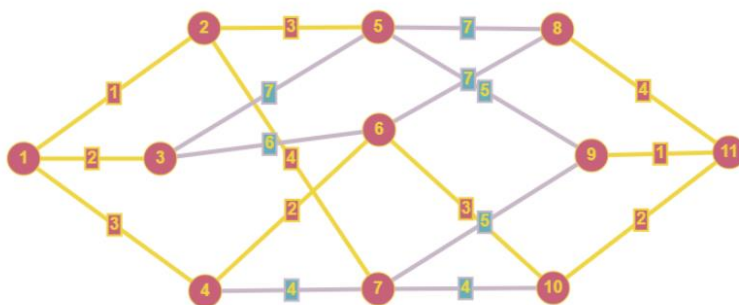
Метод Прима:



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

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

Метод Краскала:



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

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

Загальна вага остового дерева = 25.

Завдання 6.

Розв'язати задачу комівояжера для повного 8-ми вершин- ного 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	∞

1. Вихідна вершина 1:

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

Довжина шляху = 1 + 1 + 1 + 1 + 2 + 2 + 7 + 6 = 21

2. Вихідна вершина 2:

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

Довжина шляху = 3 + 1 + 1 + 2 + 3 + 5 + 5 + 7 = 27

3. Вихідна вершина 3:

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

Довжина шляху = 4 + 1 + 1 + 2 + 3 + 5 + 5 + 7 = 28

4. Вихідна вершина 4:

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

Довжина шляху = $1 + 1 + 2 + 3 + 5 + 5 + 5 + 3 = 25$

5. Вихідна вершина 5:

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

Довжина шляху = $1 + 1 + 3 + 2 + 4 + 5 + 5 + 3 = 24$

6. Вихідна вершина 6:

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

Довжина шляху = $2 + 1 + 1 + 1 + 5 + 5 + 7 + 6 = 28$

7. Вихідна вершина 7:

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

Довжина шляху = $1 + 1 + 1 + 3 + 5 + 5 + 5 + 4 = 25$

8. Вихідна точка 8

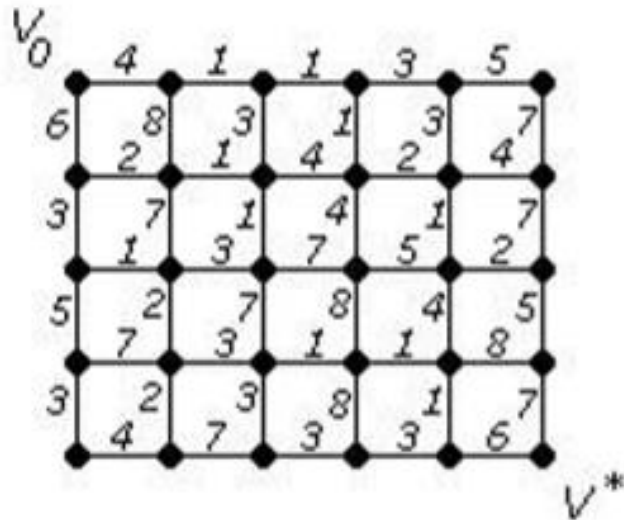
8 -> 4 -> 7 -> 1 -> 5 -> 2 -> 3 -> 6 -> 8

Довжина шляху = $2 + 1 + 1 + 1 + 1 + 5 + 7 + 2 = 20$

Проаналізувавши результати виконання алгоритму для всіх вихідних вершин, можна стверджувати, що найоптимальніше починати з вершини №8. Тоді весь шлях складатиме 20.

Завдання 7.

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V_0 і V^* :



Найкоротша відстань від вершини V_0 до кожної з вершин:

V_1 - 0

V_2 - 4

V_3 - 5

V_4 - 6

V_5 - 9

V_6 - 14

V_7 - 6

V_8 - 8

V_9 - 8

V_{10} - 7

V_{11} - 9

V_{12} - 13

V13 - 9

V14 - 10

V15 - 12

V16 - 11

V17 - 10

V18 - 12

V19 - 14

V20 - 12

V21 - 15

V22 - 15

V23 - 14

V24 - 17

V25 - 17

V26 - 14

V27 - 18

V28 - 18

V29 - 15

V30 - 21

Найкоротшого шлях:

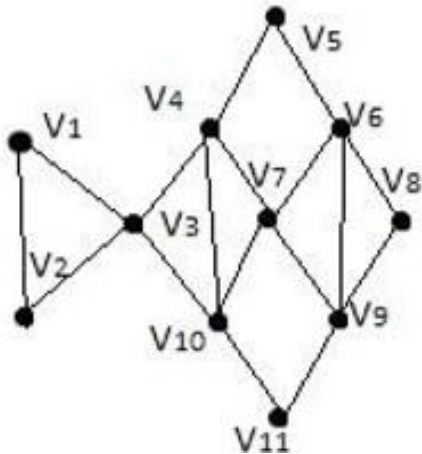
V1 -> V2 -> V3 -> V4 -> V10 -> V11 -> V17 -> V23 -> V29 -> V30

Завдання 8.

Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері;

б) Елементарних циклів.



а) $3 \Rightarrow 1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6 \Rightarrow 7 \Rightarrow 4 \Rightarrow 10 \Rightarrow 7 \Rightarrow 9 \Rightarrow 6 \Rightarrow 8 \Rightarrow 9 \Rightarrow 11 \Rightarrow 10$;

б)

3 - 1 - 2 - 3;

3 - 1 - 2 - 3 - 4 - 5;

3 - 1 - 2 - 3 - 4 - 5 - 6 - 7;

3 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 4 - 10;

3 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 4 - 10 - 7 - 9;

3 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 4 - 10 - 7 - 9 - 6 - 8;

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

Завдання 9.

Спростити формули (привести їх до скороченої ДНФ).

$$17. \quad x\bar{y} \vee \bar{x}\bar{z} \vee yz$$

$$K_1: \{0, 0, 0\} \text{ — } \bar{x}\bar{y}\bar{z}$$

$$K_2: \{0, 1, 0\} \text{ — } \bar{x}y\bar{z}$$

$$K_3: \{0, 1, 1\} \text{ — } \bar{x}yz$$

$$K_4: \{1, 0, 0\} \text{ — } x\bar{y}\bar{z}$$

$$K_5: \{1, 0, 1\} \text{ — } x\bar{y}z$$

$$K_6: \{1, 1, 1\} \text{ — } xyz$$

$$\text{ДНФ: } \bar{x}\bar{y}\bar{z} \vee \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee x\bar{y}z \vee xyz$$

Завдання додатку №2

Напишіть алгоритм

60. Обхід графа вглиб та вшир.

61. Прима знаходження найменшого остову.

62. Краскала знаходження найменшого остову.

63. Дейкстра знаходження найкоротшого ланцюга між парою вершин.

64. «Іди в найближчий» для розв'язання задачі комівояжера.

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

60. Обхід графа вглиб:

```
#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[] vis;
system("pause>>void");
}

```

Результат програми:

```

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 >> 3
Path: 3 2 1 7 4 5 6 8 10 9
Process returned 0 (0x0)   execution time : 5.669 s
Press any key to continue.

```


61. Прима знаходження найменшого остову:

```
#include <stdio.h>
#include <iostream>

using namespace std;

#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;
}

```

Результат програми:

```

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

Process returned 0 (0x0)   execution time : 0.090 s
Press any key to continue.

```

62. Краскала знаходження найменшого остову:

```
#include <cstdio>
#include <iostream>

#define size 11

using namespace std;

void Del_duplicats(int Arr[size][size]) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (j < i) {
                Arr[i][j] = 0;
            }
        }
    }
}

int Create(int A[size][size]){
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            A[i][j] = 0;
        }
    }
    for (int i = 0; i < size; i++)
    {
        A[i][i] = i + 1;
    }
    return A[size][size];
}

int NotOne(int n, int A[size][size],int f, int s) {
    int t1,t2;
    for (int i = 0; i < size; i++) {
        t1 = t2 = 0;
        for (int j = 0; j < size; j++) {
```

```

        if (A[i][j] == f) {
            t1 = 1;
        }
    }

    for (int k = 0; k < size; k++) {
        if (A[i][k] == s) {
            t2 = 1;
        }
    }

    if (t1 && t2) {
        return 0;
    }
    return 1;
}

void Add(int n, int Arr[size][size], int f, int s) {
    int scndLine;

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (Arr[i][j] == s) {
                scndLine = i;
            }
        }
    }

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (Arr[i][j] == f) {
                for (int p = 0; p < size; p++) {
                    if (Arr[scndLine][p]) {
                        Arr[i][p] = Arr[scndLine][p];
                        Arr[scndLine][p] = 0;
                    }
                }
            }
        }
    }
}

```

```

    }
}
}

```

```

int main()

```

```

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

```

```

Del_duplicats(A);

```

```

for (int k = 1; k <= 7; k++)
{
    cout << endl << "Edges with weight " << k << ": ";
    for (int i = 1; i <= size; i++){
        for (int j = 1; j <= size; j++){
            if (A[i - 1][j - 1] == k)
            {
                cout << "(" << i << "; " << j << ") ";
            }
        }
    }
}
}

```

```

int B[size][size];

```

```

Create(B);

```

```

cout << endl << endl << "Tree:";

```

```

for (int k = 1; k <= 7; k++)
{
    for (int i = 1; i <= size; i++) {
        for (int j = 1; j <= size; j++) {
            if (A[i - 1][j - 1] == k && NotOne(size, B, i, j))
            {
                Add(size, B, i, j);
                cout << "(" << i << "; " << j << ") ";
            }
        }
    }
}
cout << endl;
return 0;
}

```

Результат програми:

```

Edges with weight 1: (1;4) (2;7)
Edges with weight 2: (1;3) (2;5) (6;10)
Edges with weight 3: (4;6) (7;9) (7;10)
Edges with weight 4: (3;6) (5;8) (9;11) (10;11)
Edges with weight 5: (4;7) (5;9)
Edges with weight 6: (6;8)
Edges with weight 7: (1;2) (3;5) (8;11)

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

Process returned 0 (0x0)   execution time : 0.119 s
Press any key to continue.

```

63. Дейкстра знаходження найкоротшого ланцюга між парою вершин:

```
#include <iostream>
#include <bits/stdc++.h>

#define size 30

using namespace std;

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

    int A[size][size] =
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
// 27 28 29 30
    {0, 4, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    4, 0, 1, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 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, 1, 0, 3, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 3, 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, 5, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    6, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 8, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 7, 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, 4, 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, 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, 7, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 1, 0, 7, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 5, 0, 5, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 5, 0, 2, 0, 0, 0, 0, 4, 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, 5, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 3, 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, 2, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 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, 0, 8, 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, 0, 0, 0, 4, 0, 0, 0, 0, 1, 0, 8, 0, 0, 0, 0, 1, 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, 7,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 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, 2, 0, 0, 0, 0, 4, 0, 7, 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, 7, 0, 3, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 3, 0, 3, 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, 0, 3, 0, 6,
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, 0, 6, 0,
};

```

```

int d[size], v[size], t, min_i, min, beg_i(0);

```

```

for (int i = 0; i < size; i++)

```

```

{
    d[i] = 10000;
    v[i] = 1;
}

```

```

d[beg_i] = 0;

```

```

do {

```

```

    min_i = 10000;

```

```

    min = 10000;

```

```

    for (int i = 0; i < size; i++)

```

```

    {
        if ((v[i] == 1) && (d[i] < min))
        {
            min = d[i];
            min_i = i;
        }
    }
}

```

```

if (min_i != 10000)

```

```

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

```

```

        if (A[min_i][i] > 0)

```

```

        {
            t = min + A[min_i][i];
            if (t < d[i])
            {

```



```

        d[i] = t;
    }
}
}
v[min_i] = 0;
}
} while (min_i < 10000);

```

cout<<"Найкоротша відстань від вершини V0 до кожної з вершин: "<<endl;

```

for (int i = 0; i<size; i++)
    cout<<"V"<<i+1<<" - "<<d[i]<<endl;

```

```

int ver[size];
int end = 29;
ver[0] = end + 1;
int k = 1;
int weight = d[end];

```

```

while (end != beg_i)
{
    for (int i = 0; i<size; i++){
        if (A[end][i] != 0)
        {
            int t = weight - A[end][i];
            if (t == d[i])
            {
                weight = t;
                end = i;
                ver[k] = i + 1;
                k++;
            }
        }
    }
}
}

```

cout<<"Вивід найкоротшого шляху: "<<endl;

```

for (int i = k - 1; i >= 0; i--){
    if(ver[i]!=size)
        cout<<"V"<<ver[i]<<" -> ";
    else
        cout<<"V"<<ver[i];
}

cout<<endl;
return 0;
}

```

Результат програми:

```

Найкоротша відстань від вершини V0 до кожної з вершин:
V1 - 0
V2 - 4
V3 - 5
V4 - 6
V5 - 9
V6 - 14
V7 - 6
V8 - 8
V9 - 8
V10 - 7
V11 - 9
V12 - 13
V13 - 9
V14 - 10
V15 - 12
V16 - 11
V17 - 10
V18 - 12
V19 - 14
V20 - 12
V21 - 15
V22 - 15
V23 - 14
V24 - 17
V25 - 17
V26 - 14
V27 - 18
V28 - 18
V29 - 15
V30 - 21
Вивід найкоротшого шляху:
V1 -> V2 -> V3 -> V4 -> V10 -> V11 -> V17 -> V23 -> V29 -> V30

```

64. «Іди в найближчий» для розв'язання задачі комівояжера:

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

const int gSize=8;

int matrix[gSize][gSize];
bool visited[gSize];

int next(int dot){
    int min(999999);
    int minDot=-1;
    for(int i=0;i<gSize;i++){
        if(matrix[dot][i]<min && !visited[i] && matrix[dot][i]){
            min=matrix[dot][i];
            minDot=i;
        }
    }
    return minDot;
}

void findOst(int dot){
    while(dot!=-1){
        visited[dot]=true;
        int prevDot=dot;
        dot=next(prevDot);
        if(dot!=-1)
            cout << "(" << prevDot+1 << ";" << dot+1 << ")" ";
    }
}

int main()
{
    ifstream f1("input.txt");
    for(int i=0;i<gSize;i++){
        visited[i]=false;
        for(int j=0;j<gSize;j++){
            f1 >> matrix[i][j];
        }
    }
}
```

```
}  
findOst(0);  
return 0;  
}
```

Вносимі дані

```
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
```

Результат програми:

```
(1;5) (5;2) (2;7) (7;4) (4;8) (8;6) (6;3)  
Process returned 0 (0x0)   execution time : 0.458 s  
Press any key to continue.  
_
```

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

А)Флері:

```
#include<iostream>
#include<vector>
#define NODE 132
using namespace std;
int graph[NODE][NODE] = {
    {0,0,0,0,0,0,1,1,0,0,0,0},
    {0,0,1,0,0,1,1,1,0,0,0,0},
    {0,1,0,1,1,0,1,0,0,0,0,0},
    {0,0,1,0,0,1,0,0,0,0,0,0},
    {0,0,1,0,0,1,0,0,0,0,1,1},
    {0,1,0,1,1,0,0,0,0,1,1,1},
    {1,1,1,0,0,0,0,1,1,0,1,0},
    {1,1,0,0,0,0,1,0,0,1,0,0},
    {0,0,0,0,0,0,1,0,0,1,0,0},
    {0,0,0,0,0,1,0,1,1,0,1,0},
    {0,0,0,0,1,1,1,0,0,1,0,0},
    {0,0,0,0,1,1,0,0,0,0,0,0},
};
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]){ //when (u,v) edge is presnt and not forming bridge
            if(edge <= 1 || !isBridge(start, v)){
                cout << start+1 << "--" << v+1 << " ";
                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: ";
    fleuryAlgorithm(findStartVert());
}

```

Результат програми:

```

Euler Path Or Circuit: 1--7 7--2 2--3 3--4 4--6 6--2 2--8 8--7 7--3 3--5 5--6
6--10 10--8 10--9 9--7 7--11 11--5 5--12 12--6 6--11
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.

```

Б)Елементарних циклів

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

using namespace std;

vector<int> matr[100000];
vector<bool> visited;
vector<bool> visited2;
vector<int> path;
int previous[100000];
int n;
int start;
bool stop=false;

void dfs(int v){
    visited[v] = true;
    path.push_back(v);
    for (int i=0;i<matr[v].size() && !stop;i++){
        if(previous[v]!=matr[v][i] && matr[v][i]==start){
            stop=true;
        }
        if ((!visited[matr[v][i]]) && !visited2[matr[v][i]] && !stop){
            previous[matr[v][i]]=v;
            dfs(matr[v][i]);
        }
    }
    if(!stop){
        path.erase(path.begin()+path.size()-1);
    }
}

int main()
{
    ifstream cin("input.txt");
    cin >> n;
    for(int i=0;i<n;i++){
        visited.push_back(false);
        int a,b;
        cin >> a >> b;
```

```

    matr[a-1].push_back(b-1);
    matr[b-1].push_back(a-1);
}
visited2.assign(n,false);
for(int i=0;i<n;i++){
    if(!visited[i]){
        visited.assign(n,false);
        path.clear();
        start=i;
        dfs(i);
        if(stop){
            sort(path.begin(), path.end());
            cout << path.size() << endl;
            for(int i=0;i<path.size();i++){
                cout << path[i]+1 << " ";
            }
            cout << endl;
            stop=false;
        }else{
            visited2[start]=true;
        }
    }
}

return 0;
}

```

Вхідні дані:

```

8
1 2
2 3
3 1
3 4
4 5
5 6
6 3
6 1

```

Результат програми:

```

3
1 2 3
6
1 2 3 4 5 6

```