

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Кафедра систем штучного інтелекту**

**Розрахункова робота**

з дисципліни

«Дискретна математика»

**Виконала:**

студентка групи КН-112

Максимець Віра

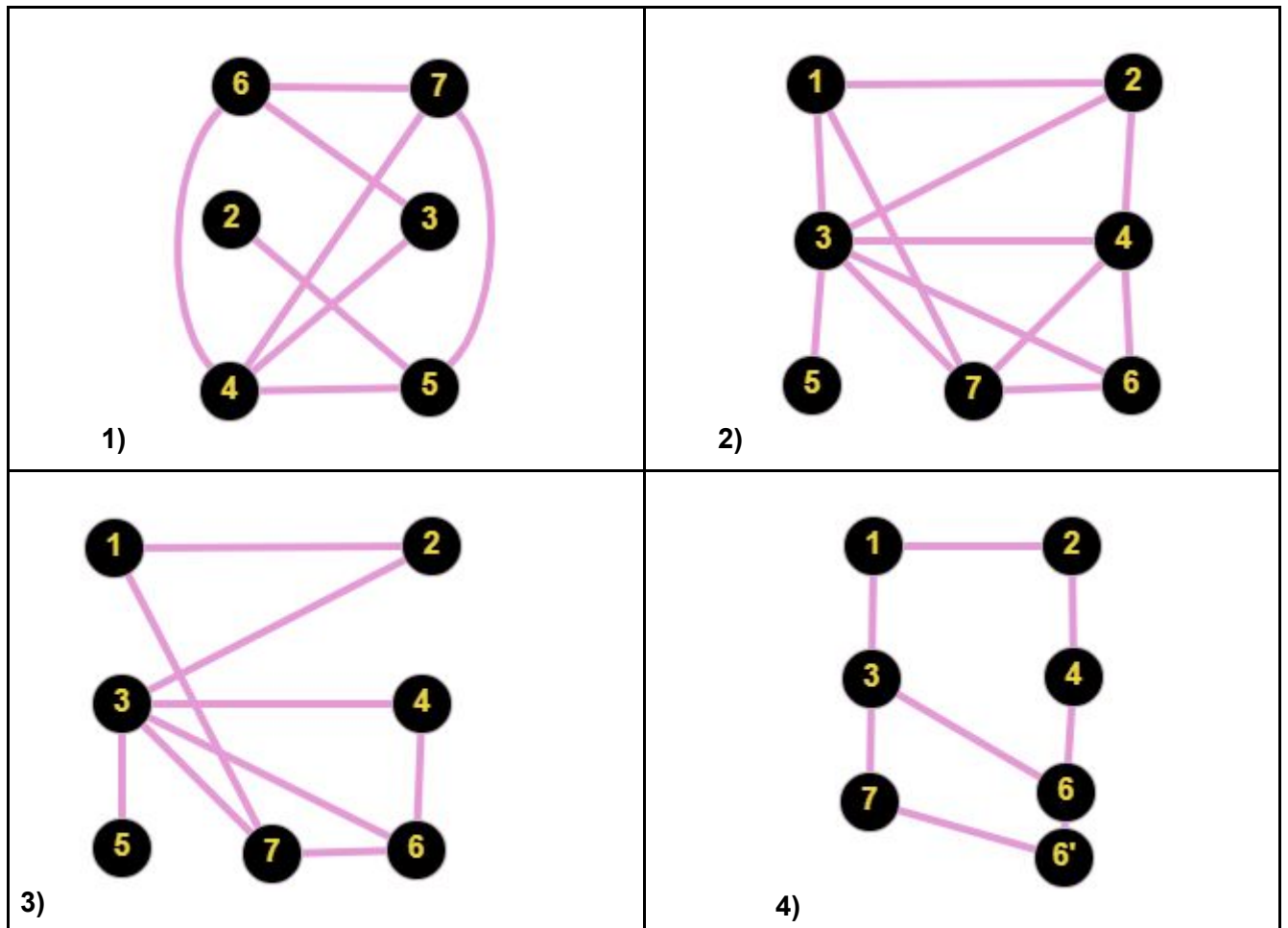
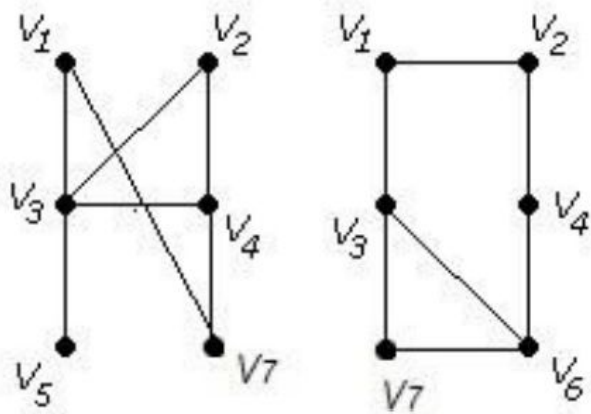
**Перевірила:**

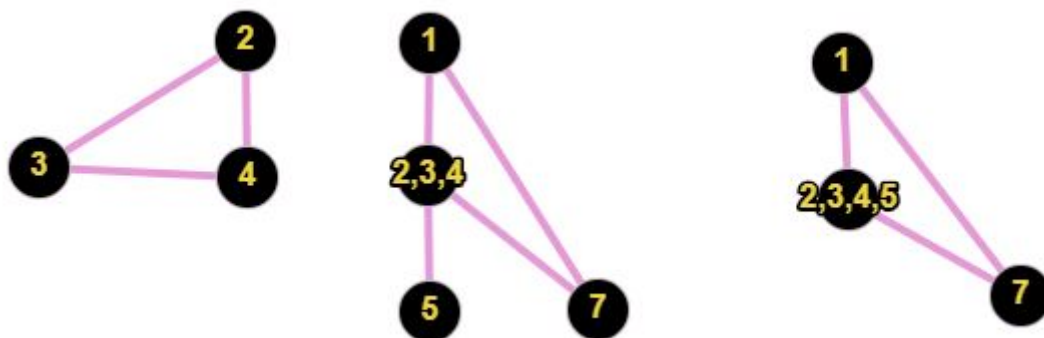
Мельникова Н. І.

Львів-2019

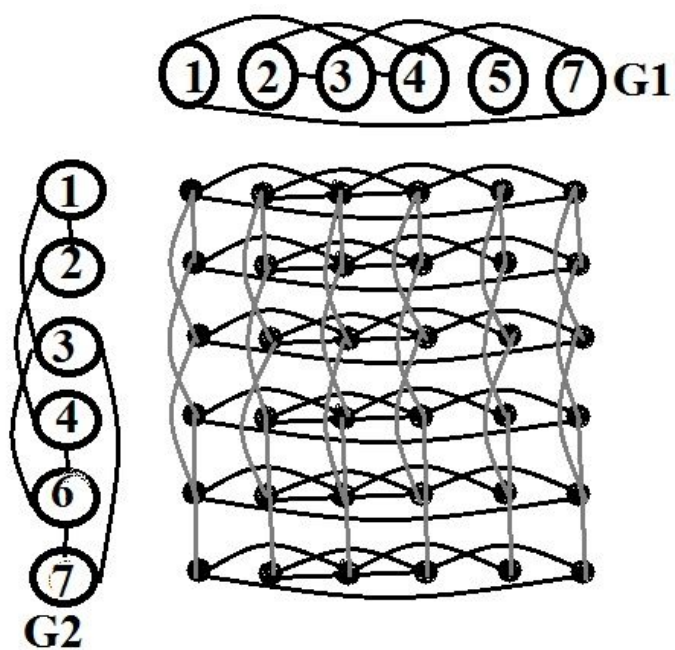
### Завдання № 1

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  $A$  - що складається з 3-х вершин в  $G1$  6) добуток графів.





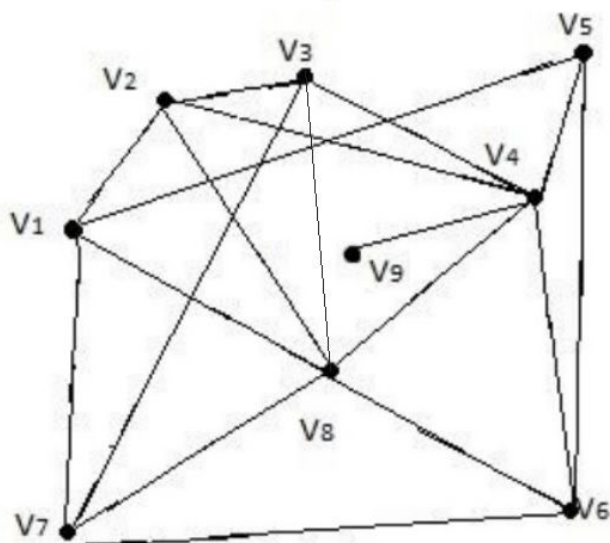
5)



6)

## Завдання № 2

Скласти таблицю суміжності для неографа



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

### Завдання № 3

Для графа з другого завдання знайти діаметр.

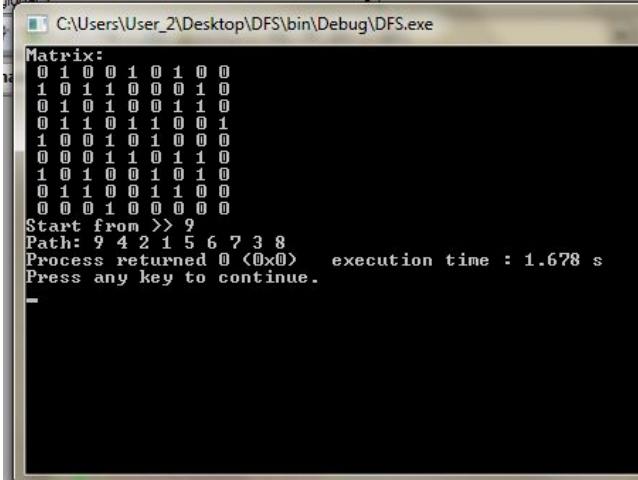
Відповідь: Діаметр = 3.

### Завдання № 4

Для графа з другого завдання виконати обхід дерева вглиб.

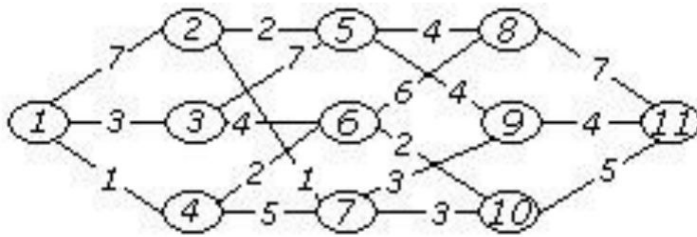
Почнемо з довільної вершини V9

Стек	Вершина, яку додали	Присвоїли номер	Стек	Вершина, яку додали	Присвоїли номер
9	9	1	9,4,3,2,8,1,7	-	-
9,4	4	2	9,4,3,2,8,1	-	-
9,4,3	3	3	9,4,3,2,8	-	-
9,4,3,2	2	4	9,4,3,2	-	-
9,4,3,2,8	8	5	9,4,3	-	-
9,4,3,2,8,1	1	6	9,4	-	-
9,4,3,2,8,1,7	7	7	9	-	-
9,4,3,2,8,1,7,6	6	8	стек порожній	-	-
9,4,3,2,8,1,7,6,5	5	9			

Програмна реалізація	Результат
<pre> #include &lt;iostream&gt; using namespace std; const int n=9; int i, j; bool *visited=new bool[n]; int graph[n][n] = { {0, 1, 0, 0, 1,0,1,0,0}, {1, 0, 1, 1, 0,0,0,1,0}, {0, 1, 0, 1, 0,0,1,1,0}, {0, 1, 1, 0, 1,1,0,0,1}, {1, 0, 0, 1, 0,1,0,0,0}, {0,0,0,1,1,0,1,1,0}, {1,0,1,0,0,1,0,1,0}, {0,1,1,0,0,1,1,0,0}, {0,0,0,1,0,0,0,0,0} }; void DFS(int st) { int r; cout&lt;&lt;st+1&lt;&lt;" "; visited[st]=true; for (r=0; r&lt;n; r++) if ((graph[st][r]!=0) &amp;&amp; (!visited[r])) DFS(r); } int main() { int start; cout&lt;&lt;"Matrix: "&lt;&lt;endl; for (i=0; i&lt;n; i++) { visited[i]=false; for (j=0; j&lt;n; j++) cout&lt;&lt;" "&lt;&lt;graph[i][j]; cout&lt;&lt;endl; } cout&lt;&lt;"Start from &gt;&gt; "; cin&gt;&gt;start; bool *vis=new bool[n]; cout&lt;&lt;"Path: "; DFS(start-1); delete []visited; } </pre>	

### Завдання № 5

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

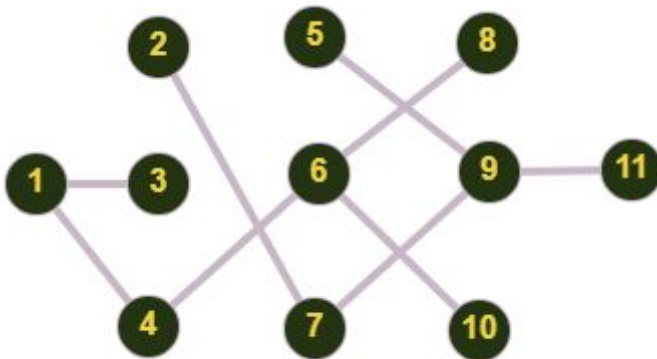


#### Метод Краскала:

1. V1+V4
2. V2+V7
3. V4+V6
4. V6+V10
5. V1+V3
6. V7+V9
7. V9+V5
8. V9+V11
9. V6+V8

#### Метод Прима:

1. V1+V4
2. V4+V6
3. V6+V10
4. V10+V7
5. V7+V2
6. V2+V5
7. V5+V9
8. V9+V11
9. V11+V8



#### Метод Краскала

```
#include <iostream>
using namespace std;
void Probe(int v, int AM[][11], int Values[], int Lines[]);
int main()
{ //матриця суміжності
  int AM[11][11] =
  {
    {99,7,3,1,99,99,99,99,99,99,99},
    {7,99,99,99,5,99,1,99,99,99,99},
    {3,99,99,99,7,4,99,99,99,99,99},
    {1,99,99,99,99,2,5,9,99,99,99},
    {99,2,7,99,99,99,99,4,4,99,99},
    {99,99,4,2,99,99,99,6,99,2,99},
    {99,1,99,5,99,99,99,99,3,3,99},
```

#### Метод Прими

```
#include <iostream>
#include <iomanip>
using namespace std;
int formation(int vershunu,int**arr, int* V, int*
counter_of_nodes);
void print(int vershunu, int** arr);
bool check_node(int* V, int vershunu, int Node);
void build(int vershunu, int* min, int* counter_of_nodes, int*
weight_of, int** arr, int* V, int* E);
int min_numb(int vershunu, int* min, int* counter_of_nodes,
int** arr, int* V);
void del(int vershunu, int** arr);
int main() {
  setlocale(LC_ALL, "rus");
```

```

{99,99,99,99,4,6,99,99,99,99,7},
{99,99,99,99,4,99,3,99,99,99,4},
{99,99,99,99,99,2,3,99,99,99,5},
{99,99,99,99,99,99,99,7,4,5,99}
};
cout << "Kruskal method" << endl;
//вара ребер
int Values[]={1,2,3,4,5,6,7}; // "v"
int value = (sizeof(Values))/4;
//масив для запису пройдених ребер
int Lines[11];
//занулити
for (int i=0;i<value;i++)
{
    Lines[i]=0;
}
//виклик функції
for (int weight=0;weight<value;weight++)
{
    Probe(weight, AM, Values, Lines);
}
return 0;
}
void Probe(int v, int AM[][11], int Values[], int Lines[])
{
    int counter1 = 0;
    int counter2 = 0;
    bool flag1, flag2;
    for (int i=0;i<11;i++)
    {
        for (int j=0;j<11;j++)
        {
            if (AM[i][j]==Values[v])
            {
                for (int x=0;x<11;x++)
                {
                    if (Lines[x]!=i) //якщо нема ще такого
                    {
                        counter1++;
                    }
                    if (Lines[x]!=j) //якщо нема ще такого
                    {
                        counter2++;
                    }
                }
            }
            if (counter1==11)
            {
                Lines[i]=i;
                flag1 = true;
            }
            if (counter2==11)
            {
                Lines[j]=j;
                flag2 = true;
            }
        }
    }
}

```

```

int vershunu, min, counter_of_nodes=0, weight_of=0;
cout << "Введіть кількість вершин графа: ";
cin >> vershunu;
int* V = new int[vershunu];
int* E = new int[vershunu - 1];
int** arr = new int*[vershunu];
for (int i = 0; i < vershunu; i++)
    arr[i] = new int[vershunu];
cout << "Введіть матрицю суміжності";
min=formation(vershunu, arr,V,&counter_of_nodes);
if (min == -1) {
    cout << "Program error" << endl;
}
else {print(vershunu, arr);
    for (int i = 0; i < vershunu; i++) {build(vershunu, &min,
&counter_of_nodes, &weight_of, arr, V, E);
        min = min_numb(vershunu, &min,
&counter_of_nodes, arr, V);}
    int sum = 0;
    for (int i = 0; i < vershunu-1; i++)
        sum += E[i];
    cout << "Вара остового дерева: " << sum << endl;}
    del(vershunu, arr);
    delete[] E;
    delete[] V;
    return 0;
}
int formation(int vershunu, int** arr,int *V,int
*counter_of_nodes) {
    int min = 99999;
    for (int i = 0; i < vershunu; i++)
        for (int j = 0; j < vershunu; j++) {
            cin >> arr[i][j];
            if (min > arr[i][j] && arr[i][j] != 0) {
                min = arr[i][j];
                V[*counter_of_nodes] = i;
                if (cin.fail()) {
                    cout << "Input error" << endl;
                    return -1;
                }
            }
        }
    (*counter_of_nodes)++;
    return min;
}
void print(int vershunu, int** arr) {
    for (int i = 0; i < vershunu; i++) {
        for (int j = 0; j < vershunu; j++) {
            cout << setw(4) << arr[i][j];
        }
        cout << endl;
    }
}
void del(int vershunu, int** arr) {
    for (int i = 0; i < vershunu; i++) {
        delete[] arr[i];
    }
    delete[] arr;}
bool check_node(int* V, int vershunu, int Node) {
    for (int i = 0; i < vershunu; i++)

```

```

        if ((flag1==false)&&(flag2==false))
        { }
        else {
            cout << "\n Connection between: {" <<
Lines[i]+1 << "," << Lines[j]+1 << "}\n";
            cout << "Line weight: " << v+1 << endl;;
        }
    }
    counter1=0;
    counter2=0;
    flag1 = false;
    flag2 = false;
}
}
}

```

```

        if (V[i] == Node)return false;
        return true;}

void build(int vershunu, int* min, int* counter_of_nodes, int*
weight_of, int** arr, int* V, int* E) {
    for (int i = 0; i < vershunu; i++)
        for (int j = 0; j < vershunu; j++) {
            for(int k=0;k< *counter_of_nodes;k++)
                if (arr[i][j] == *min && (i==V[k] || j==V[k])) {
                    if (check_node(V, vershunu, i) || check_node(V, vershunu, j)) {
                        if (check_node(V, vershunu, i)) {
                            V[*counter_of_nodes] = i;
                            (*counter_of_nodes)++;}
                        if (check_node(V, vershunu, j)) {
                            V[*counter_of_nodes] = j;
                            (*counter_of_nodes)++;
                            E[*weight_of] = arr[i][j];
                            (*weight_of)++;
                        }
                    }
                    cout << endl << setw(2) << i + 1 << "----" << j + 1 << " (" <<
arr[i][j] << ")" << endl;

                    *min = 9999;
                    arr[i][j] = 0;
                    arr[j][i] = 0;
                    return;}

                else {
                    cout << endl << setw(2) << "There is a loop" << endl;
                    *min = 9999;
                    arr[i][j] = 0;
                    arr[j][i] = 0;
                    return;}}}}

int min_numb(int vershunu, int* min, int* counter_of_nodes,
int** arr, int* V) {
    for (int i = 0; i < *counter_of_nodes; i++)
        for (int j = 0; j < vershunu; j++)
            if (*min > arr[V[i]][j] && arr[V[i]][j] != 0)* min = arr[V[i]][j];
    return *min;}

```



```

"C:\Users\User_2\Desktop\юы | Ёх\—шѐЁхЁр\—шѐ ырс4\bin\Debug\Dis444.exe"
Kruskal method
Connection between: <2;7>
Line weight: 1
Connection between: <1;6>
Line weight: 2
Connection between: <5;2>
Line weight: 2
Connection between: <6;10>
Line weight: 2
Connection between: <1;3>
Line weight: 3
Connection between: <7;9>
Line weight: 3
Connection between: <5;8>
Line weight: 4
Connection between: <9;11>
Line weight: 4
Connection between: <4;7>
Line weight: 5
Process returned 0 (0x0)   execution time : 0.015 s
Press any key to continue.

```

```

C:\Users\User_2\Desktop\Prima\bin\Debug\Prima.exe
Введіть кількість вершин графа: 11
Введіть матрицю суміжності
0 7 3 1 0 0 0 0 0 0 0
7 0 0 0 5 0 1 0 0 0 0
3 0 0 0 7 4 0 0 0 0 0
1 0 0 0 2 5 0 0 0 0 0
0 2 7 0 0 0 4 4 0 0 0
0 0 4 2 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 4 0 3 0 0 0 4
0 0 0 0 0 2 3 0 0 0 5
0 0 0 0 0 0 0 7 4 5 0
0 7 3 1 0 0 0 0 0 0 0
7 0 0 0 5 0 1 0 0 0 0
3 0 0 0 7 4 0 0 0 0 0
1 0 0 0 0 2 5 0 0 0 0
0 2 7 0 0 0 0 4 4 0 0
0 0 4 2 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 4 0 3 0 0 0 4
0 0 0 0 0 2 3 0 0 0 5
0 0 0 0 0 0 0 7 4 5 0
1---4 <1>
4---6 <2>
6---10 <2>
1---3 <3>
7---10 <3>
2---7 <1>
7---9 <3>
There is a loop
5---9 <4>
There is a loop
5---8 <4>
Варта остовного дерева: 23
Process returned 0 (0x0)   execution time : 129.881 s
Press any key to continue.

```

## Завдання № 6

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

23)

	1	2	3	4	5	6	7	8
1	$\infty$	4	6	5	1	5	6	5
2	4	$\infty$	6	5	5	5	5	7
3	6	6	$\infty$	1	2	3	2	5
4	5	5	1	$\infty$	1	5	7	5
5	1	5	2	1	$\infty$	5	5	6
6	5	5	3	5	5	$\infty$	7	1
7	6	5	2	7	5	7	$\infty$	2
8	5	7	5	5	6	1	2	$\infty$

### Почнемо з 1

- 1) 1-5-4-3-7-8-6-2---1 d=12
- 2) 1-2-4-5-3-7-8-6---1 d=22
- 3) 1-4-3-5-2-6-8-7---1 d=27
- 4) 1-6-3-4-5-2-7-8---1 d=27
- 5) 1-8-6-3-4-5-2-7---1 d=27
- 6) 1-3-4-5-6-2-7-8---1 d=30
- 7) 1-7-3-4-5-2-6-8---1 d=26

Очевидно, що шляхом Комівояжера є: **1-5-4-3-7-8-6-2---1 d=12**

Програма	Результат
<pre> #include &lt;stdlib.h&gt; #include &lt;time.h&gt; #include &lt;stdio.h&gt; #define SIZE 10  int wpchk(int w, int *wpts) {     int i=0;     int flg=0;     while(wpts[i]!=-1)     {         if(wpts[i]==w){flg=1;}         i++;     }     if (flg==0) {return 0;} else return 1; }  int main() {     srand( (unsigned)time( NULL ) );     int waypoint[11]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1};     int way[11]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};     int start=-1;     int end=-1;     int min;     int imin;      int prices[][10]={// 0 1 2 3 4 5 6 7 8 9         { 0, 0, 4, 6, 5, 1, 5, 6, 5, 0}, //0         { 0, 4, 0, 6, 5, 5, 5, 5, 7, 0}, //1         { 0, 6, 6, 0, 1, 2, 3, 2, 5, 0}, //2         { 0, 5, 5, 1, 0, 1, 5, 7, 5, 0}, //3         { 0, 1, 5, 2, 1, 0, 5, 5, 6, 0}, //4         { 0, 5, 5, 3, 5, 5, 0, 7, 1, 0}, //5         { 0, 6, 5, 2, 7, 5, 7, 0, 2, 0}, //6         { 0, 5, 7, 5, 5, 6, 1, 2, 0, 0}, //7         { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, //8         { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} //9     } </pre>	

```

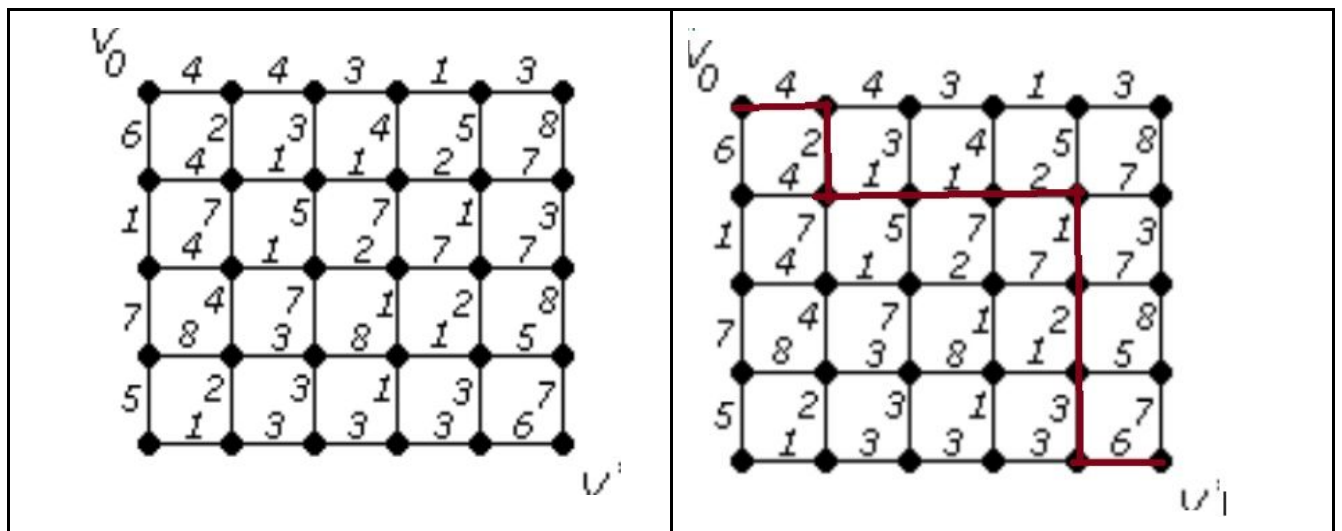
};

printf("Enter # of start location:");
scanf("%i",&start);
printf("Enter # of finish location:");
scanf("%i",&end);
waypoint[0]=start;
int n=0;
int w;
while(waypoint[n]!=end)
{
    min=0;
    w=waypoint[n];
    for(int i=0;i<SIZE;i++)
    {
        if(((min==0)||((prices[w][i]<min)&&(prices[w][i]>0)))&
        &wpchk(i,waypoint)==0)
        {min=prices[w][i];imin=i;}
    }
    n++;
    waypoint[n]=imin;
}
printf("\nThe way is:\n");
int i=0;
while(waypoint[i]!=-1)
{
    printf("%i ",waypoint[i]);
    i++; } }

```

### Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин  $V_0$  і  $V_1$ . Алгоритм Дейкстри полягає у тому, щоб розглянути всі варіанти "шляху" до вершини  $i$  і вибрати з них найоптимальніший (-> найкоротший)



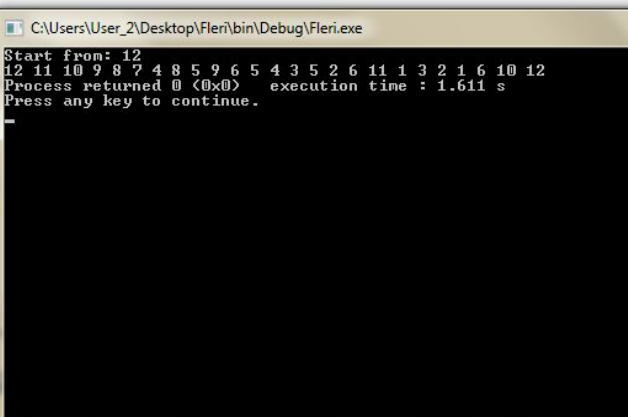
1 част	2 част
<pre> #include &lt;iostream&gt;  using namespace std;  const int SIZE = 30; void Show(int seen[], int k); int main() { int Matrix[][30] = {     { 0,4,0,0,0,0,6,0 },     { 4,0,4,0,0,0,0,2,0 },     { 0,4,0,3,0,0,0,0,3,0 },     { 0,0,3,0,1,0,0,0,0,4,0 },     { 0,0,0,1,0,3,0,0,0,0,5,0 },     { 0,0,0,0,3,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 6,0,0,0,0,0,4,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,4,0,1,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,3,0,0,0,0,1,0,1,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,4,0,0,0,0,1,0,2,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,5,0,0,0,0,2,0,7,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,8,0,0,0,0,7,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,1,0,0,0,0,0,4,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,0,7,0,0,0,0,4,0,1,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,5,0,0,0,0,1,0,2,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,7,0,0,0,0,2,0,7,0,0,0,0,1,0,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,7,0,7,0,0,0,0,2,0,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,7,0,0,0,0,0,0,8,0,0,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,8,0,0,0,0,0,5,0,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,8,0,3,0,0,0,0,2,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,3,0,8,0,0,0,0,2,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,8,0,1,0,0,0,0,1,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,5,0,0,0,0,3,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0,0,7,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,1,0,0,0,0,0 },     { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,3,0,0,0,0 },     { 0,3,0,0,0,0,3,0,3,0,0,0 },     { 0,3,0,0,0,0,3,0,3,0,0 },     { 0,3,0,0,0,0,3,0,6,0 },     { 0,7,0,0,0,0,6,0,0 } }; cout &lt;&lt; "The adjacency matrix.\n" &lt;&lt; endl; for (int i =0;i&lt;SIZE; i++) {     for (int j=0;j&lt;SIZE; j++)     {         cout &lt;&lt; Matrix[i][j] &lt;&lt; " ";     }     cout &lt;&lt; endl; } int dis[SIZE];    // відстань int visited[SIZE]; //відвідані вершини int minindex, min; int startpoint = 0; int split = SIZE/3; for (int i = 0; i&lt;SIZE; i++) </pre>	<pre> if (minindex != 10000) {     for (int i = 0; i&lt;SIZE; i++)     {         if (Matrix[minindex][i] &gt; 0)         {             int point = min + Matrix[minindex][i]; //додати             знайдену мін вагу до існуючої ваги вершини             if (point &lt; dis[i]) // порівняти з                 поточною вагою             {                 dis[i] = point;             }         }     }     visited[minindex] = 0; } } while (minindex &lt; 10000); // відновлення шляху int endy; cout &lt;&lt; "\nEnter the end-point: "; cin &gt;&gt; endy; int end = endy-1;  int seen[SIZE]; // масив відвіданих вершин  seen[0] = end + 1; // початковий елемент - кінцева вершина int weight = dis[end]; // вага кінцева вершини int k = 1; // індекс попередньої  while (split!=0) // пока не початок {     for (int i = 0; i&lt;SIZE; i++)     {         if ((Matrix[end][i] != 0)&amp;&amp;(Matrix[end][i] != seen[k]))         // якщо вершини суміжні         {             int point = weight - Matrix[end][i];             if (point == dis[i]) // якщо вага не співпадає             {                 weight = point;                 end = i; // зберігаємо попередню                 вершину                 seen[k] = i + 1; // і записуємо її в масив                 k++;             }         }     }     split--; } Show(seen,k); cout &lt;&lt; "\n Total weight is " &lt;&lt; dis[SIZE-1] &lt;&lt; endl; </pre>



```

{0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0},
{0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0},//5
{1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0},
{0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0},//7
{0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0},
{0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0},//9
{0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1},
{1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1},//11
{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0}
};
int k;
int Stack[STACK_SIZE];
void Search(int v)
{
    int i;
    for(i = 0; i < N; i++)
        if(G[v][i])
        {
            G[v][i] = G[i][v] = 0;
            Search(i); }
    Stack[++k] = v;}
int main()
{
    int T, p, q, s;
    int j, vv;
    T = 1;
    for(p = 0; p < N; p++) {
        s = 0;
        for(q = 0; q < N; q++)
        {
            s += G[p][q];
        }
        if(s%2) T = 0; }
    k = -1;
    printf("Start from: "); scanf("%d", &vv);
    if(T) {
        Search (vv-1);
        for(j = 0; j <= k; j++)
            printf("%d ", Stack[j]+1); }
    else
        printf("not Eulerian graph\n");
    return 0;

```

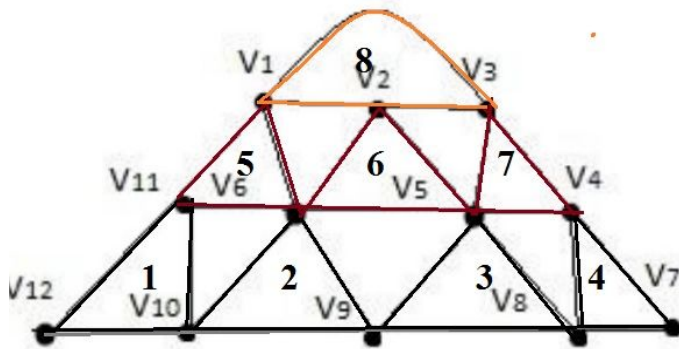


```

C:\Users\User_2\Desktop\Fleri\bin\Debug\Fleri.exe
Start from: 12
12 11 10 9 8 7 4 8 5 9 6 5 4 3 5 2 6 11 1 3 2 1 6 10 12
Process returned 0 (0x0)   execution time : 1.611 s
Press any key to continue.

```

**Знаходимо елементарні цикли і об'єднуємо їх в один**



### Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

23.  $(x \vee \bar{y})(\bar{y} \vee \bar{z})$

$$\begin{aligned}
 (x \vee \bar{y}) \wedge (\bar{y} \vee \bar{z}) &= ((x \vee \bar{y}) \wedge \bar{y}) \vee ((x \vee \bar{y}) \wedge \bar{z}) = (x \vee \bar{y}) \vee (\bar{y} \vee \bar{y}) \vee (x \wedge \bar{z}) \vee (\bar{y} \wedge \bar{z}) = \\
 &= (x \vee \bar{y}) \vee \bar{y} \vee (x \wedge \bar{z}) \vee (\bar{y} \wedge \bar{z}) = (x \vee \bar{y}) \vee \bar{y} \vee ((x \wedge \bar{z}) \vee \bar{y}) \wedge ((x \wedge \bar{z}) \wedge \bar{z}) = \\
 &= (x \vee \bar{y}) \vee \bar{y} \vee (((x \vee \bar{y}) \wedge (\bar{z} \vee \bar{y})) \wedge ((x \wedge \bar{z}) \wedge (\bar{z} \wedge \bar{z}))) = \\
 &= (x \vee \bar{y}) \vee \bar{y} \vee (((x \vee \bar{y}) \wedge (\bar{z} \vee \bar{y})) \wedge x \wedge \bar{z}) = x \vee \bar{y} \vee (x \wedge \bar{z} \wedge ((x \vee \bar{y}) \wedge (\bar{z} \vee \bar{y}))) = \\
 &= x \vee \bar{y} \vee (x \wedge \bar{z} \wedge (((x \vee \bar{y}) \wedge \bar{z}) \vee ((x \vee \bar{y}) \wedge \bar{y}))) = \\
 &= x \vee \bar{y} \vee x\bar{z}(x\bar{z} \vee \bar{y}\bar{z} \vee x \vee \bar{y}\bar{y})
 \end{aligned}$$