МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-114

Бідак Юлія

Викладач:

Мельникова Н.І.

Львів – 2019 р.

Варіант 19

Завдання № 1 Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

2) об’єднання графів,

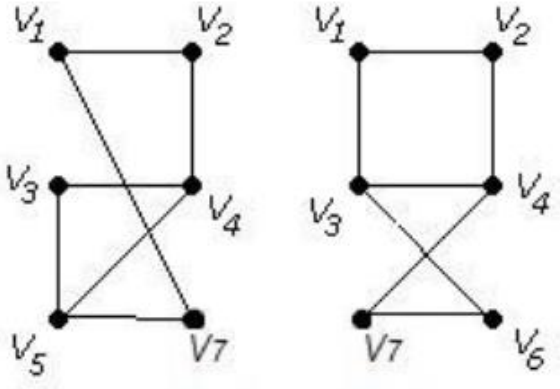
3) кільцеву суму G1 та G2 (G1+G2),

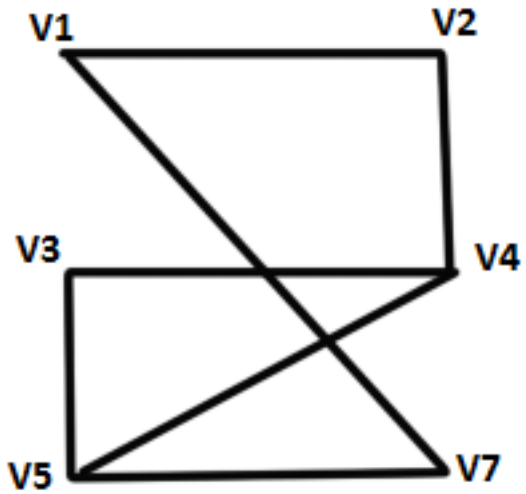
4) розмножити вершину у другому графі,

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

6) добуток графів.

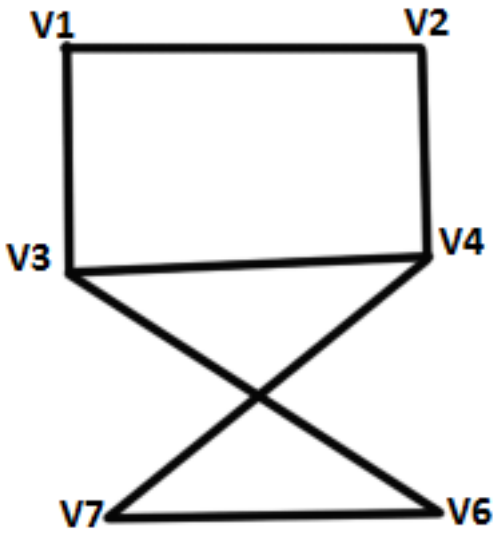
19)





V1 = { V1,V2,V3,V4 , V5, V7 }

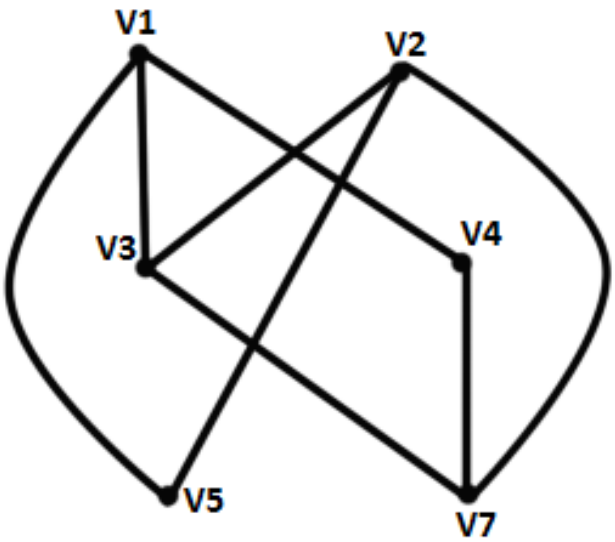
E1 = { (V1,V2), (V1,V7), (V2,V4), (V3,V4) , (V3,V5), (V4,V5), (V5,V7)}



V2 = { V1, V2, V3,V4, V6, V7}

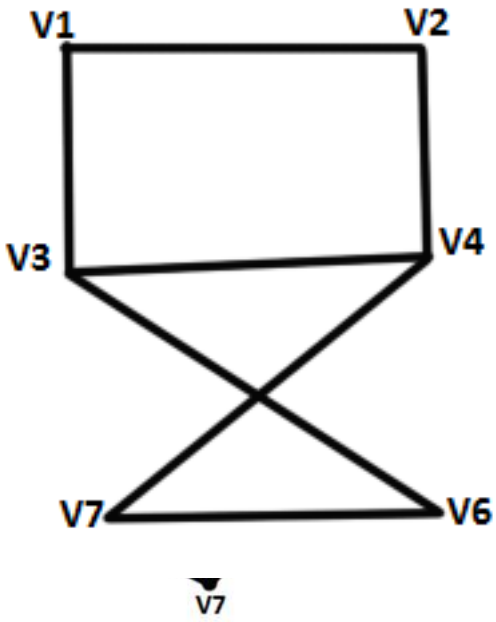
E2 = { (V1,V2), (V1,V3), (V2,V4), (V3,V4), (V3,V6), (V4,V7), (V6,V7) }

1) доповнення до першого графу



V3 = { V1, V2,V3,V4 , V5, V7 }

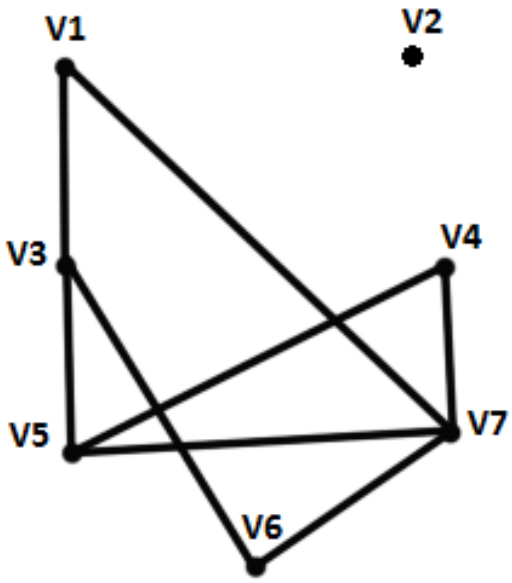
E3 = { (V1,V3), (V1, V4), (V1,V5), (V2,V3), (V2,V5), (V2,V7), (V3,V7), (V4,V7)}



2) об’єднання графів

V4 = { V1,V2,V3,V4 , V5,V6, V7 }

E4 = { (V1,V2), (V1,V3), (V1,V7), (V2,V4), (V3,V4), (V3,V5), (V3,V6), (V4,V5), (V4,V7), (V5,V7),(V6,V7)}

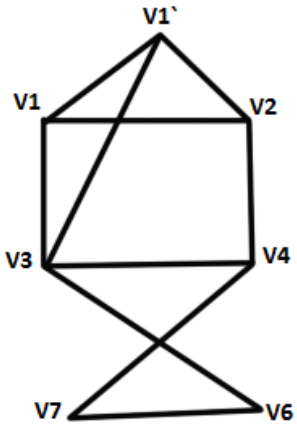


3) кільцева сума G1 та G2 (G1+G2)

V5 = { V1,V2,V3,V4 , V5,V6, V7 }

E5 = { (V1,V3), (V1,V7),(V3,V5), (V3,V6), (V4,V5), (V4,V7),(V6,V7)}

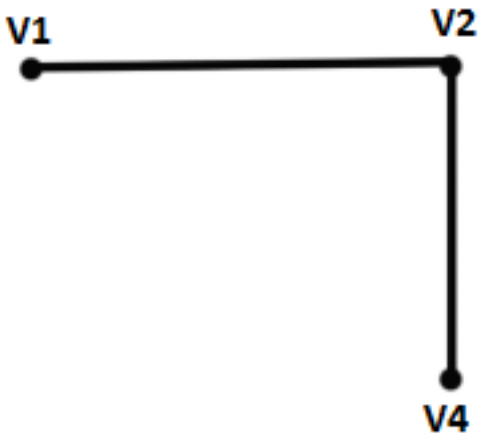
4) розмножимо вершину у другому графі



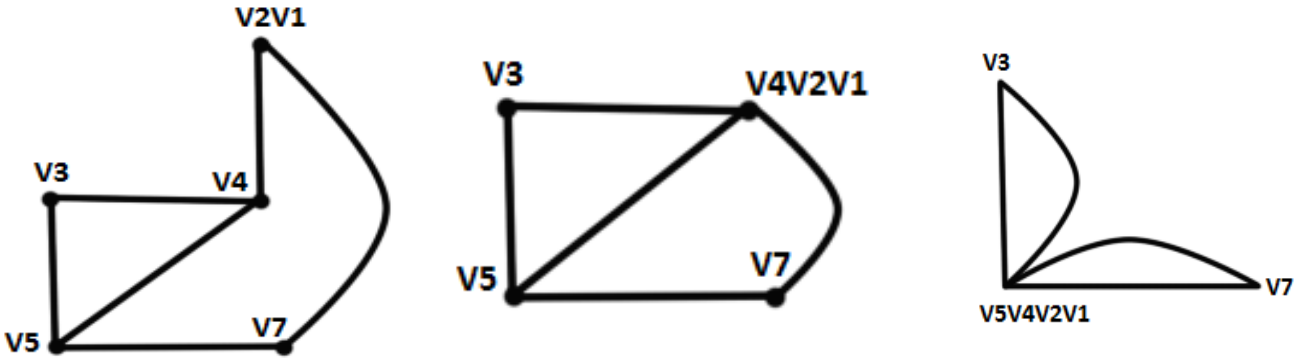
V6 = { V1, V1`, V2, V3,V4, V6, V7}

E6 = { (V1,V2), (V1,V3), (V1,V1`),(V1`,V2), (V1`,V3), (V2,V4), (V3,V4), (V3,V6), (V4,V7), (V6,V7) }

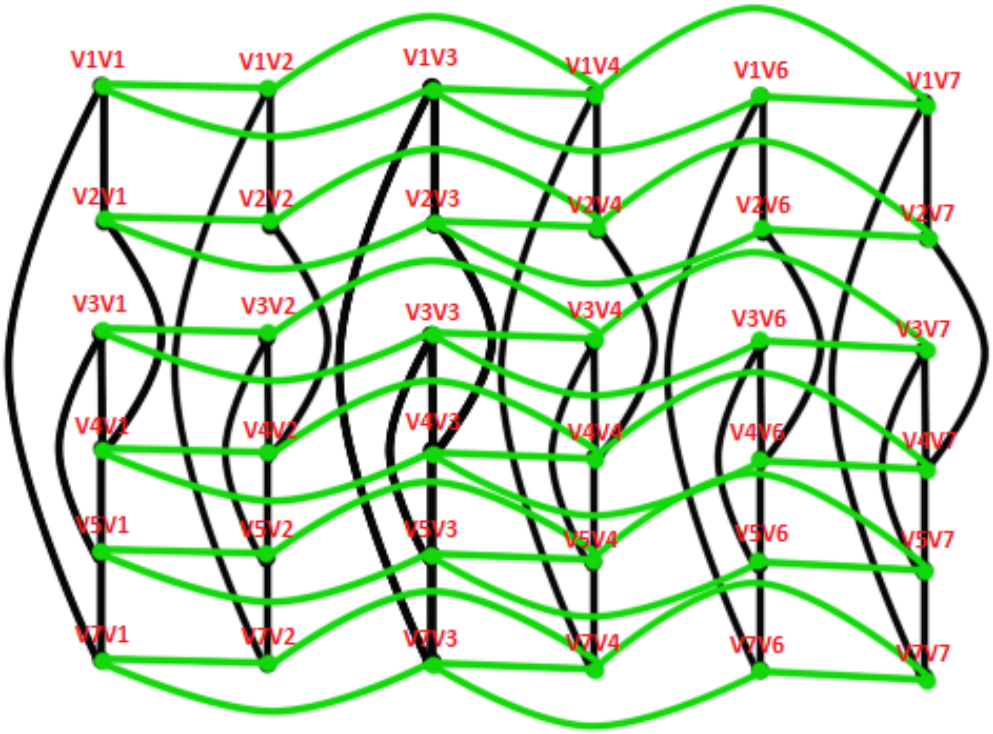
5) виділити підграф А - що скадається з 3-х вершин в G1



G1 і знайдемо стягнення А в G1 (G1\ A),

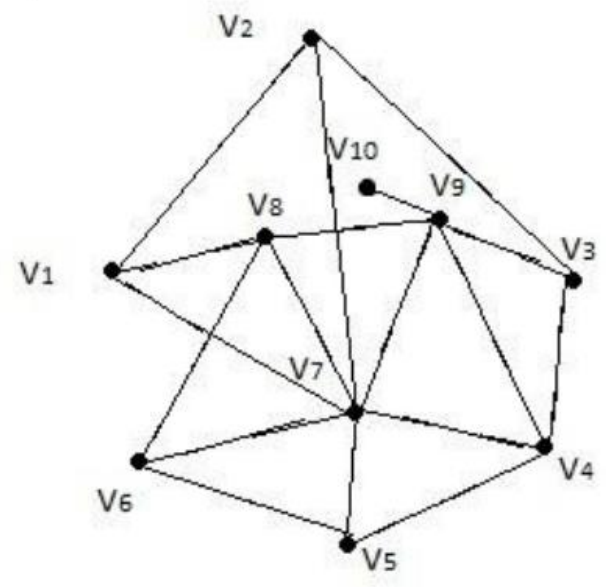


6) добуток графів.



Завдання № 2 Таблицю суміжності для орграфа.

19)



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	1	0	0	0	0	1	1	0	0
V2	1	0	1	0	0	0	1	0	0	0
V3	0	1	0	1	0	0	0	0	1	0
V4	0	0	1	0	1	0	1	0	1	0
V5	0	0	0	1	0	1	1	0	0	0
V6	0	0	0	0	1	0	1	1	0	0
V7	1	1	0	1	1	1	0	1	1	0
V8	1	0	0	0	0	1	1	0	1	0
V9	0	0	1	1	0	0	1	1	0	1
V10	0	0	0	0	0	0	0	0	1	0

Завдання № 3 Для графа з другого завдання знайти діаметр.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	-	1	2	2	2	2	1	1	2	3
V2	1	-	1	2	2	2	1	2	2	3
V3	2	1	-	1	2	3	2	2	1	2
V4	2	2	1	-	1	2	1	2	1	2
V5	2	2	2	1	-	1	1	2	2	3
V6	2	2	3	2	1	-	1	1	2	3
V7	1	1	2	1	1	1	-	1	1	2
V8	1	2	2	2	2	1	1	-	1	2
V9	2	2	1	1	2	2	1	1	-	1
V10	3	3	2	2	3	3	2	2	1	-

Діаметр графа : 3

Завдання № 4 Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число)

Вершина	DFS- номер	Вміст стеку
V1	1	V1
V2	2	V1V2
V3	3	V1V2V3
V4	4	V1V2V3V4
V5	5	V1V2V3V4V5
V6	6	V1V2V3V4V5V6
V7	7	V1V2V3V4V5V6 V7
V8	8	V1V2V3V4V5V6V7V8
V9	9	V1V2V3V4V5V6V7V8V9
V10	10	V1V2V3V4V5V6V7V8V9V10
-	-	V1V2V3V4V5V6V7V8V9
-	-	V1V2V3V4V5V6V7V8
-	-	V1V2V3V4V5V6 V7
-	-	V1V2V3V4V5V6
-	-	V1V2V3V4V5
-	-	V1V2V3V4

-	-	V1V2V3
-	-	V1V2
-	-	V1
-	-	0

Програмна реалізація:

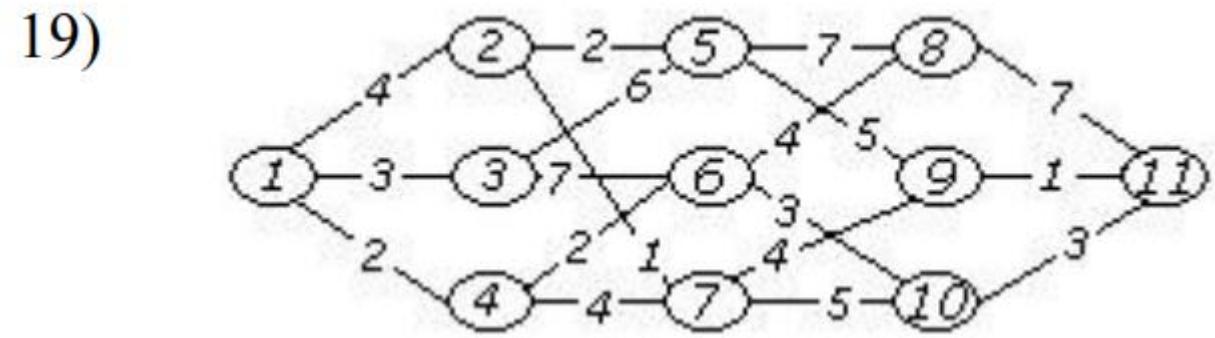
```
#include <iostream>
using namespace std;
const int n=10;
bool *visited=new bool[n];
int graph[n][n] =
{
{0, 1, 0, 0, 0, 0, 1, 1, 0, 0},
{1, 0, 1, 0, 0, 0, 1, 0, 0, 0},
{0, 1, 0, 1, 0, 0, 0, 0, 1, 0},
{0, 0, 1, 0, 1, 0, 1, 0, 1, 0},
{0, 0, 0, 1, 0, 1, 1, 0, 0, 0},
{0, 0, 0, 0, 1, 0, 1, 1, 0, 0},
{1, 1, 0, 1, 1, 1, 0, 1, 1, 0},
{1, 0, 0, 0, 0, 1, 1, 0, 1, 0},
{0, 0, 1, 1, 0, 0, 1, 1, 0, 1},
{0, 0, 0, 0, 0, 0, 0, 0, 1, 0}
};
void DFS(int first)
{
int r;
cout<<first+1<<" ";
visited[first]=true;
for (r=0; r<=n; r++)
if ((graph[first][r]!=0) && (!visited[r]))
DFS(r);
}
int main()
{
setlocale(LC_ALL, "ukr");
int start;
```

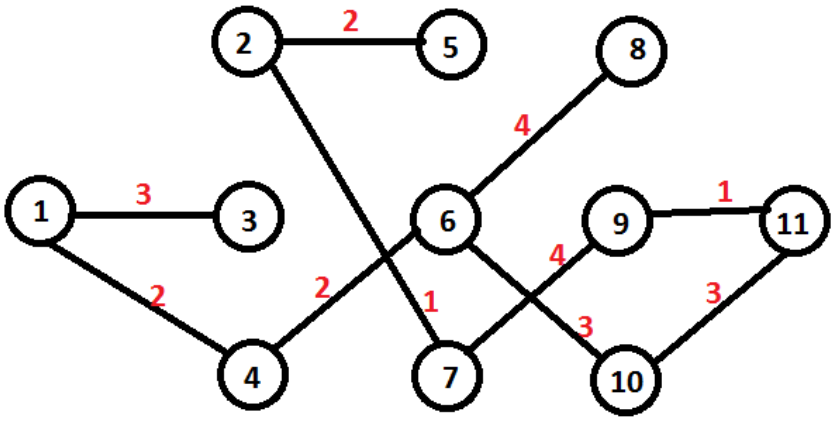
```
int start;
cout<<"Матриця суміжності графа: "<<endl;
for (int i=0; i<n; i++)
{
visited[i]=false;
for (int j=0; j<n; j++)
cout<<" "<<graph[i][j];
cout<<endl;
}
cout<<"Початкова вершина >> ";
 cin>>start;
cout<<"Обхід : ";
DFS(start-1);
delete []visited;

}
```

Результат виконання програми :

```
Матриця сум?жност? графа:
 0 1 0 0 0 0 1 1 0 0
 1 0 1 0 0 0 1 0 0 0
 0 1 0 1 0 0 0 0 1 0
 0 0 1 0 1 0 1 0 1 0
 0 0 0 1 0 1 1 0 0 0
 0 0 0 0 1 0 1 1 0 0
 1 1 0 1 1 1 0 1 1 0
 1 0 0 0 0 1 1 0 1 0
 0 0 1 1 0 0 1 1 0 1
 0 0 0 0 0 0 0 0 1 0
Початкова вершина >> 1
Обх?д : 1 2 3 4 5 6 7 8 9 10 11
Process returned 0 (0x0)   execution time : 20.778 s
Press any key to continue.
```

Завдання № 5 Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.





Алгоритм Прима :

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

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

Алгоритм Краскала:

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

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

Вага мінімального остового дерева : 25

Програмна реалізація Краскала :

```cpp
#include <iostream>
using namespace std;
#define  A 1000

const int V = 11;
int parent[V];
int finds(int i)
{
    while (parent[i] != i)
        i = parent[i];
    return i;
}
void union1(int i, int j) /// перевірка чи існує вже пара (i,j)
{
    int a = finds(i);
    int b = finds(j);
    parent[a] = b;
}

void kraskal(int matrix[][V])
{
    int mincost = 0;
    for (int i = 0; i < V; i++)
        parent[i] = i;
    int edge_count = 0;
    while (edge_count < V - 1) {
        int min = A, a = -1, b = -1;
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                if (finds(i) != finds(j) && matrix[i][j] < min) {
                    min = matrix[i][j];
                    a = i;
                    b = j;
                }
            }
        }

        union1(a, b);
        cout << "( "<< a+1<<", "<< b+1<<" ) "<<min <<endl;
        edge_count++,
        mincost += min;/// найменші ваги
    }
    cout << "Minimal weight = " << mincost;
}

int main()
{
    int matrix[][V] = {

            {A,4, 3, 2,A,A,A,A,A,A,A },
            { 4,A,A,A,2,A,7,A,A,A,A },
            { 3,A,A,A,6,7,A,A,A,A },
            {  2,A,A,A,A,2,4,A,A,A,A },
            { A,2,6,A,A,A,A,7,5,A,A },
            { A,A,7,2,A,A,A,4,A,3,A},
            {A,7,A,4,A,A,A,A,4,5,A },
            {A,A,A,A,7,4,A,A,A,A,7  },
            { A,A,A,A,5,A,4,A,A,A,1 },
            { A,A,A,A,A,3,5,A,A,A,3 },
```

```
            { A,A,A,A,A,3,5,A,A,A,3 },
            { A,A,A,A,A,A,A,7,1,3,A }
    };

    kraskal(matrix);

    return 0;
}
```

Результат виконання програми :

```
            { A,A,A,A,A,3,5,A,A,A,3 },
            { A,A,A,A,A,A,A,7,1,3,A }
    };

    kraskal(matrix);

    return 0;
}
```

Програмна реалізація Прима :

Start here | 4laba.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int n,m,v1,v2,w,sum = 0;
cout<< "Enter the  number of vertices of the graph"<<endl;
cin>> n;
cout<< "Enter the number of edges of the graph"<<endl;
cin >> m;
cout<< "Enter two vertices and weight between them"<<endl;

int **arr = new int* [n];
for (int i = 0;i< n;i++){
    arr[i]= new int [n];
    for (int j=0;j<n;j++){
        arr[i][j]=0;
    }
}
for (int i=0; i<m;i++){
    cin>> v1>>v2>>w;
    arr[v1-1][v2-1]=w;
    arr[v2-1][v1-1]=w;
}
int *arr2 = new int [n];
for (int i=0;i<n;i++){
    arr2[i]=0;
}
arr2[0]=1;
```

```
31  for( int r=0,t = 0; r != (n-1); ){
32      t = 0;
33      for (int i=0;i<n;i++){
34          if (arr2[i] != 0 ){
35              for (int j=0; j<n ; j++){
36                  if( arr[i][j] != 0 ){
37                      if (t==0){
38                          v1 = i;
39                          v2 = j;
40                          w = arr[v1][v2];
41                          t++;
42                      }
43                      else {
44                          if(arr[i][j]< w){
45                              v1=i;
46                              v2 =j;
47                              w = arr[i][j];
48
49                          }
50                      }
51                  }
52              }
53          }
54      }
55
56  sum+=w;
57  cout<<" V("<<v1+1<<") - "<<" V(" << v2+1 <<") = "<<w<<endl;
58  arr2[v2]= v2+1;
59  arr[v1][v2]=0;
60  arr[v2][v1]=0;
```

```
61  r = 0;
62  for (int q = 0; q < n; q ++ ){
63      if( arr2[q] != 0 ){ r++;}
64  }
65
66  }
67
68  for (int i = 0; i<n ; i++){
69      if (arr2[i] == 0){v2 = i;}
70  }
71  for (int i=0, t=0; i<n; i++){
72      if (arr[i][v2] != 0 ){
73          if(t==0){
74              v1 = i;
75              w = arr[i][v2];
76              t++;
77          }
78          else {
79              if (arr[i][v2]<w){
80                  v1 = i;
81                  w = arr[i][v2];
82              }
83          }
84      }
85  }
86  sum +=w;
87  cout << " V("<< v1 + 1<<") - V("<< v2+1<<") = "<<w<<endl;
88  cout << " The minimal weight :"<< sum;
89
90      return 0;
```

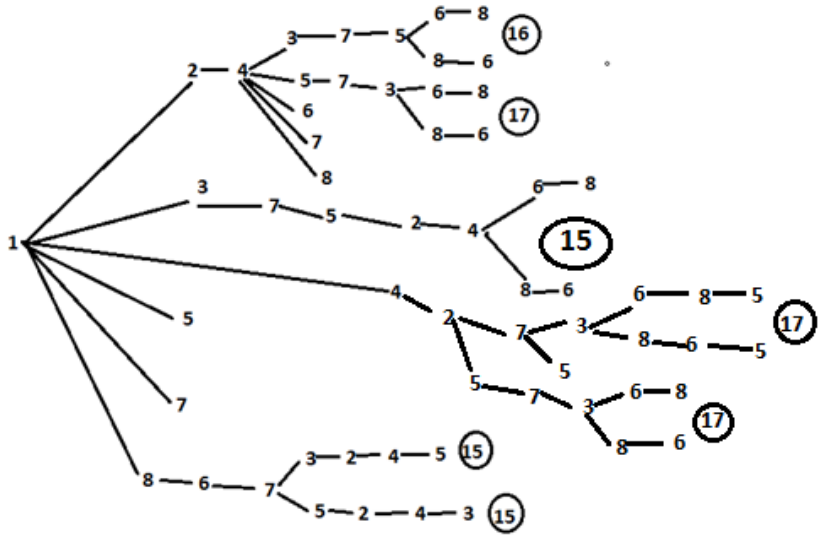
Результат виконання програми :

```
Enter the  number of vertices of the graph
11
Enter the number of edges of the graph
18
Enter two vertices and weight between them
1 2 4
1 3 3
1 4 2
2 5 2
2 7 1
3 5 6
3 6 7
4 6 2
4 7 4
5 8 7
5 9 5
6 8 4
6 10 3
7 9 4
7 10 5
8 11 7
9 11 1
10 11 3
 V(1) -  V(4) = 2
 V(4) -  V(6) = 2
 V(1) -  V(3) = 3
 V(6) -  V(10) = 3
 V(10) -  V(11) = 3
 V(11) -  V(9) = 1
 V(1) -  V(2) = 4
 V(2) -  V(7) = 1
 V(2) -  V(5) = 2
 V(6) - V(8) = 4
 The minimal weight :25
Process returned 0 (0x0)   execution time : 114.387 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:

19)

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



Програмна реалізація :

```cpp
#include <bits/stdc++.h>
using namespace std;
#define V 8

int komivoy(int graph[][V], int s)
{
    vector<int> vertex;
    for (int i = 0; i < V; i++)
        if (i != s)
            vertex.push_back(i);

    int min_path = INT_MAX;
    do {
        int current_pathweight = 0;
        int k = s;
        for (int i = 0; i < vertex.size(); i++) {
            current_pathweight += graph[k][vertex[i]];
            k = vertex[i];
        }
        current_pathweight += graph[k][s];
        min_path = min(min_path, current_pathweight);

    } while (next_permutation(vertex.begin(), vertex.end()));

    return min_path;
}

int main()
{
```

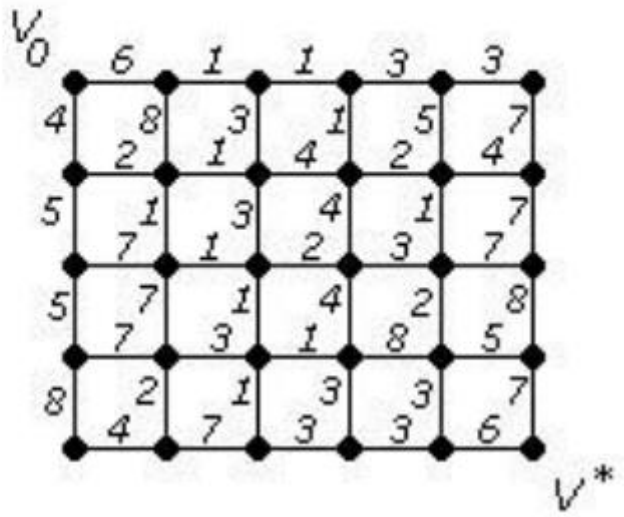
```cpp
int main()
{
    int graph[][V] = { {0, 2, 2, 2, 2, 3, 2, 2},
                       {2, 0, 5, 1, 2, 3, 2, 4},
                       {2, 5, 0, 6, 6, 5, 1, 5},
                       {2, 1, 6, 0, 6, 6, 6, 6},
                       {2, 2, 6, 6, 0, 5, 1, 5},
                       {3, 3, 5, 6, 5, 0, 2, 1},
                       {2, 2, 1, 6, 1, 2, 0, 5},
                       {2, 4, 5, 6, 5, 1, 5, 0}
                     };
    int s = 0;
    cout << komivoy(graph, s) << endl;
    return 0;
}
```

Результати виконання програми :

```
The minimal length between cities is 15

Process returned 0 (0x0)   execution time : 0.024 s
Press any key to continue.
```

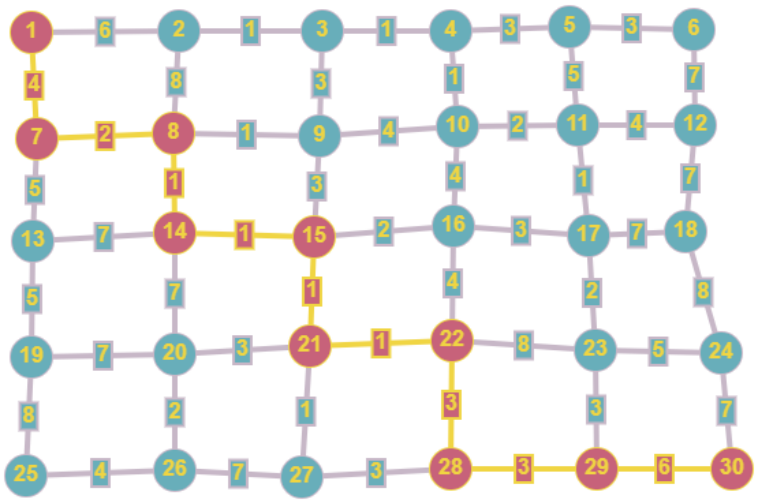
Завдання № 7 За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V і V*

19)



Вершина	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
1	6					4																							
7	6						6					9																	
2		7					6					9																	
8		7						7				9	7																
3			8					7				9	7																
9			8						13			9	7	10															
14			8						13			9		8					14										
4				11					9			9		8					14										

15				11				9			9			10				14	9									
10				11					11		9			10				14	9									
13				11					11					10			14	14	9									
21				11					11					10			14	12		10					10			
16				11					11						13		14	12		10					10			
22				11					11						13		14	12			18				10	13		
27				11					11						13		14	12			18			17		13		
5					14				11						13		14	12			18			17		13		
11					14					15					12		14	12			18			17		13		
17					14					15						19	14	12			14			17		13		
20					14					15						19	14				14			14		13		
28					14					15						19	14				14			14			16	
6										15						19	14				14			14			16	
19										15						19					14		22	14			16	
23										15						19						19	22	14			16	
26										15						19						19	22				16	
12																19						19	22				16	
29																19						19	22					22



Найкоротший шлях : V1-V7-V8-V14-V15-V21-V22 – V28 – V29- V30 = 22 або

V1- V7 – V8 – V14 – V15-V21 – V27 – V28 – V29 – V30

Програмна реалізація :

```cpp
#include<iostream>
using namespace std;
int n,i, j,dist[40],pred[40],c[40][40];
bool visited[40];
int minDistance()
{
    int minimum = 1000, minD;
    for (int v = 0; v < n; v++)
        if (visited[v]==false && dist[v] <= minimum)
        {
            minimum = dist[v];
            minD = v;
        }
    return minD;
}
void printPath(int j)
{
    if (pred[j] == -1)
        return;
    printPath(pred[j]);
    cout << "V" << j+1 << " ";
}
void dijkstra(int c[40][40])
{
    int point;
    cout << "Enter start point : ";
    cin >> point;
    for (int i = 0; i < n; i++)
    {
        pred[0] = -1;
        dist[i] = 1000;
        visited[i] = false;
    }
    dist[point-1] = 0;
    for (int i = 0; i < n - 1; i++)
    {
        int u = minDistance();
        visited[u] = true;
        for (int v = 0; v < n; v++)
            if (visited[v]==false && c[u][v] && dist[u] + c[u][v] < dist[v])
            {
                pred[v] = u;
                dist[v] = dist[u] + c[u][v];
            }
    }
    cout << "The least way is: ";
    cout << dist[29] << endl;
    cout << "The way is: ";
    cout << "V1 ";
    printPath(29);
    cout << endl;
}
int main()
{

    int g1, g2;
    cout << "Enter the number of vertices: ";
    cin >> n;
```

```
57      cout << "Enter the number of vertices: ";
58      cin >> n;
59      for (int i = 0; i < n; i++) {
60
61          for (int j = 0; j < n; j++)
62          {
63              c[i][j] = 0;
64          }
65      }
66      cout<<" Enter number of columns ";
67      cin >> g1;
68      cout<<" Enter number of rows ";
69      cin >> g2;
70
71      for (i = 0; i < n; i++) {
72          for (j = i + 1; j < n; j++)
73          {
74              if (j == i + 1 || j == i + g1) {
75                  cout << "From " << i+1 << " to  " << j+1 << ": ";
76                  cin >> c[i][j];
77              }
78              else {
79                  c[i][j] = 0;
80              }
81          }
82      }
83      dijkstra(c);
84      return 0;
85  }
86
```

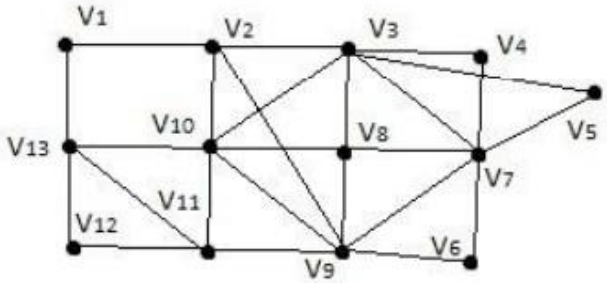
Результат виконання програми :

```
Enter the number of vertices: 30
 Enter number of columns 6
 Enter number of rows 5
From 1 to  2: 6
From 1 to  7: 4
From 2 to  3: 1
From 2 to  8: 8
From 3 to  4: 1
From 3 to  9: 3
From 4 to  5: 3
From 4 to  10: 1
From 5 to  6: 3
From 5 to  11: 5
From 6 to  7: 0
From 6 to  12: 7
From 7 to  8: 2
From 7 to  13: 5
From 8 to  9: 1
From 8 to  14: 1
From 9 to  10: 4
From 9 to  15: 3
From 10 to  11: 2
From 10 to  16: 4
From 11 to  12: 4
From 11 to  17: 1
From 12 to  13: 0
From 12 to  18: 7
From 13 to  14: 7
From 13 to  19: 5
From 14 to  15: 1
From 14 to  20:
7
From 15 to  16: 2
From 15 to  21: 1
From 16 to  17: 3
From 16 to  22: 4
From 17 to  18: 7
From 17 to  23: 2
From 18 to  19: 0
From 18 to  24: 8
From 19 to  20: 7
From 19 to  25: 8
From 20 to  21: 3
From 20 to  26: 2
From 21 to  22: 1
From 21 to  27: 1
From 22 to  23: 8
From 22 to  28: 3
From 23 to  24: 5
From 23 to  29: 3
From 24 to  25: 0
From 24 to  30: 7
From 25 to  26: 4
From 26 to  27: 7
From 27 to  28: 3
From 28 to  29: 3
From 29 to  30: 6
From 27 to  28: 3
From 28 to  29: 3
From 29 to  30: 6
Enter start point : 1
The least way is: 22
The way is: V1 V7 V8 V14 V15 V21 V27 V28 V29 V30

Process returned 0 (0x0)   execution time : 203.147 s
Press any key to continue.
```

Завдання № 8 Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері; б) елементарних циклів.

19)



Розв’язок :

а) Алгоритм Флері :

V1V2; V2V9; V9V6; V6V7; V7V4;V4V3; V3V5; V5V7; V7V9; V9V8; V8V7; V7V3; V3V10; V10V2; V2V3; V3V8; V8V10; V10V9; V9V11; V11V13; V13V10; V10V11; V11V12; V12V13; V13V1

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

#define N 13
#define STACK_SIZE 100
using namespace std;

int G[N][N] {
    {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
    {1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0},
    {0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0},
    {0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0},
    {0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0},
    {0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0},
    {0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1},
    {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 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])
```

```
31          {
32              G[v][i] = G[i][v] = 0;
33              Search(i);
34          }
35      Stack[++k] = v+1;
36  }
37  int main()
38  {
39      int T, p, q, s;
40      int j, vv;
41      T = 1;
42      for(p = 0; p < N; p++)
43      {   s = 0;
44          for(q = 0; q < N; q++)
45          {
46              s += G[p][q];
47          }
48          if(s%2) T = 0;
49      }
50      k = -1;
51      cout << "Start vertice : ";
52      cin >> vv;
53      vv-=1;
54      if(T)
55      {
56          Search (vv);
57          for(j = 0; j <= k; j++)
58              cout << Stack[j] << " ";
59      }
60      else
```

```
60      else
61          cout << "it is not Eulerian graph\n";
62      return 0;
63  }
64
```

Результати виконання програми :

```
Start vertice : 1
1 13 12 11 13 10 11 9 10 8 9 7 8 3 10 2 9 6 7 5 3 7 4 3 2 1
Process returned 0 (0x0)   execution time : 1.475 s
Press any key to continue.
```

б) Алгоритм на основі циклів

Елементарні цикли :

V1 – V2 – V3 – V4 – V7 – V6 – V9 – V11 – V12 – V13 – V1

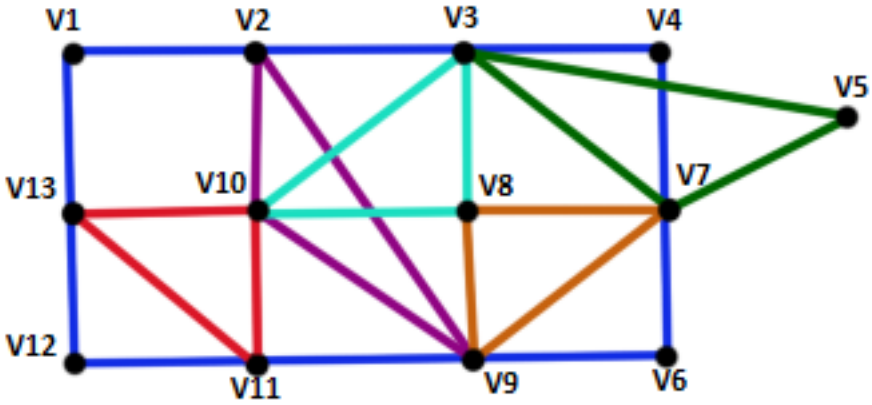
V13 – V10 – V11 – V13

V2 – V10 – V9 – V2

V10 – V3 – V8- V10

V9 – V8 – V7 –V9

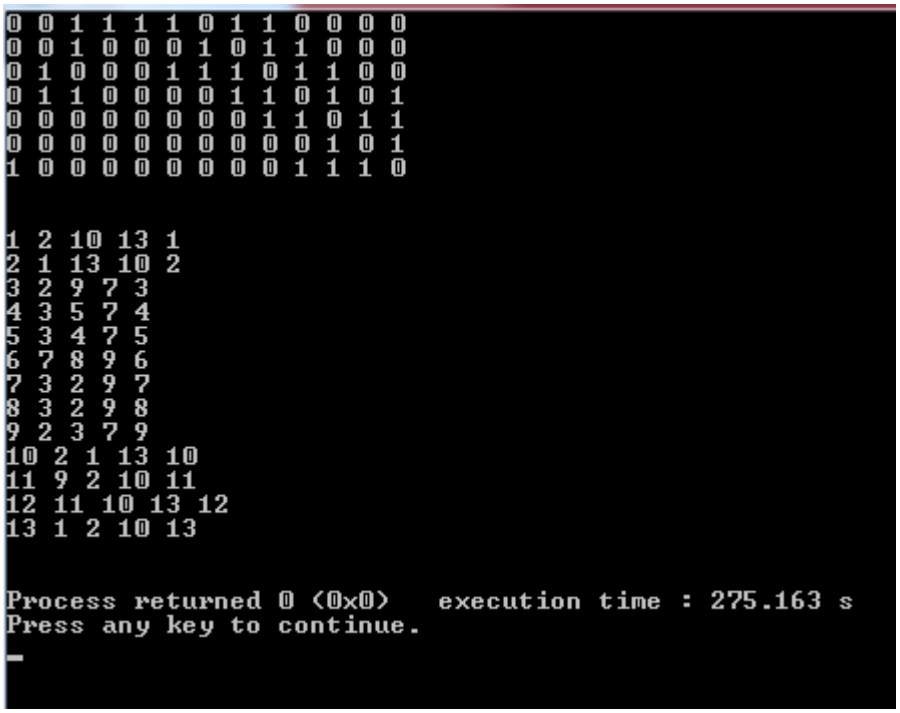
V3 – V5 – V7- V3




```cpp
#include <iostream>
#include <vector>
using namespace std;
vector<int> R;
int MAXS = 1000;
bool check(vector<int> V, int vertices) {
  for (auto i = V.begin(); i != V.end(); i++) {
    if (*i == vertices)return false;
  }
  return true;
}
void Find(vector<int>* V, int** arr, int n, int pos, int start_vertices) {
  for (int i = pos, k = 0; k < 1; i++, k++) {
    for (int j = 0; j < n; j++)
      if (arr[i][j] == 1 && check((*V), j)) {
        if (j == start_vertices && (*V).size() > 2) {
          if (MAXS > V->size()) {
            R.clear();
            R.push_back(start_vertices + 1);
            for (auto it = (*V).begin(); it != (*V).end();
              it++)
              R.push_back(*it + 1);
            R.push_back(start_vertices + 1);
            MAXS = V->size();
            break;
          }
        }
        else {
          (*V).push_back(j);
          Find(V, arr, n, j, start_vertices);
```

```
            Find(V, arr, n, j, start_vertices);
          }
        }
    }
    if (V->size() != 0)
      V->pop_back();
}
int main() {
    int n;
    cout << "Enter number of vertices: ";
    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> V;
    vector<int> WAS;
    cout << endl;
    int count, p, q, sum;
    count = 1;
    for (p = 0; p < n; p++)
    {
      sum = 0;
      for (q = 0; q < n; q++)
      {
        sum += arr[p][q];
      }
      if (sum % 2) count = 0;
    }
    cout << endl;
    if (count) {
      for (int j = 0; j < n; j++) {
        MAXS = 1000;
        Find(&V, arr, n, j, j);
        for (int i = 1; i <= R.size(); i++) {
          cout << R[i - 1] << " ";
        }
        cout << endl;
        R.clear();
      }
    }
    else
      cout << "Graf is not correct \n";
    cout << endl;
    return 0;
}
```

Результат виконання програми :



Завдання №9 Спростити формули (привести їх до скороченої ДНФ)

19. $\overline{\overline{xy}(x\bar{y}z \vee \bar{x}y)}$

Розв’язок :

$(xy) \vee \overline{(x\bar{y}z \vee \bar{x}y)}$

$(xy) \vee (x\bar{y}z \wedge \overline{\bar{x}y})$

$(xy) \vee((\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y}))$

$(xy) \vee(((\bar{x} \vee y \vee \bar{z}) \wedge x) \vee((\bar{x} \vee y \vee \bar{z}) \wedge \bar{y}))$

$(xy) \vee((0 \vee xy \vee x\bar{z}) \vee (\bar{x}\bar{y} \vee 0 \vee \bar{y}\bar{z}))$

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

Відповідь : $xy \vee x\bar{z} \vee \bar{x}\bar{y}$