

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ
УКРАЇНИ НАЦІОНАЛЬНОМУ
УНІВЕРСИТЕТУ “ЛЬВІВСЬКА
ПОЛІТЕХНІКА”**

Кафедра систем штучного інтелекту

Розрахункова робота
з дисципліни
«Дискретна математика»

Виконав:

студент групи КН-115

Кагуй Андрій

Викладач:

Мельникова Н.І.

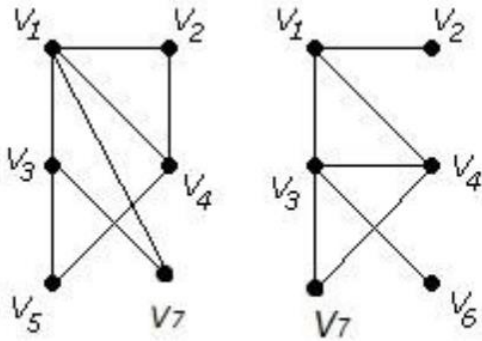
Львів-2019

Варіант 9

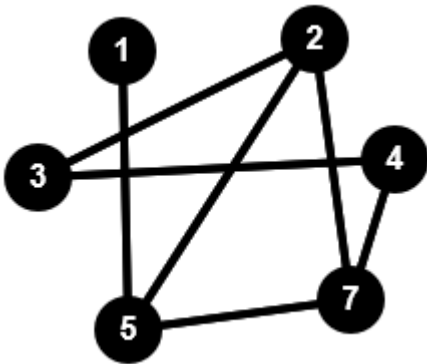
Завдання № 1

Виконати наступні операції над графами:

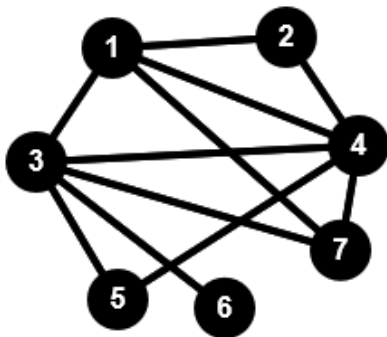
9)



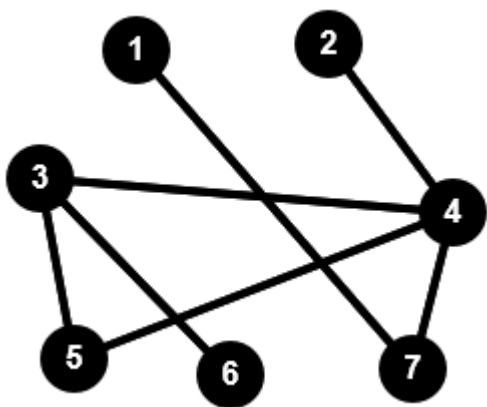
1) знайти доповнення до першого графу,



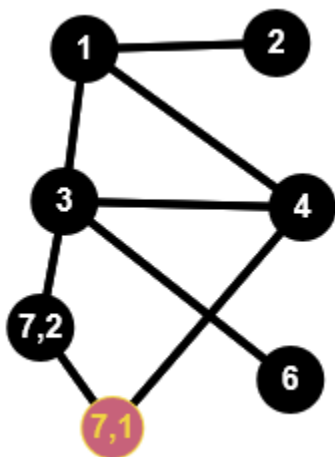
2) об'єднання графів,



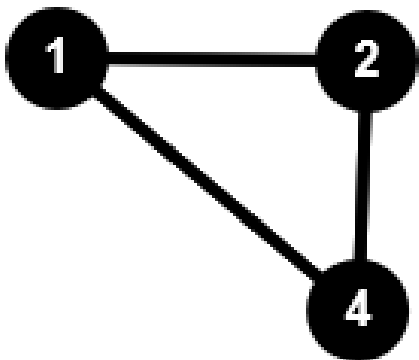
3) кільцеву суму $G1$ та $G2$ ($G1+G2$)



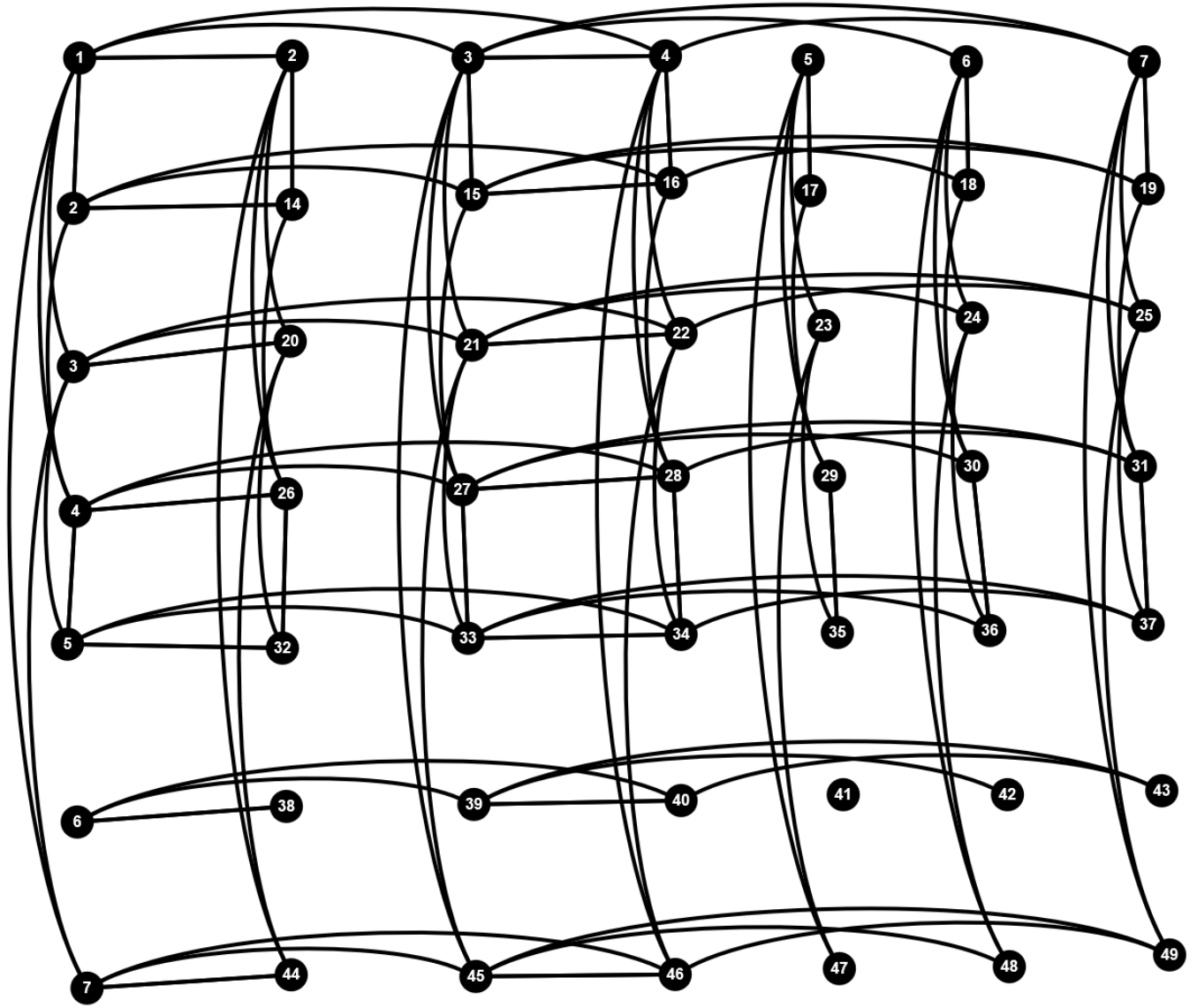
4) розмножити вершину у другому графі,



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



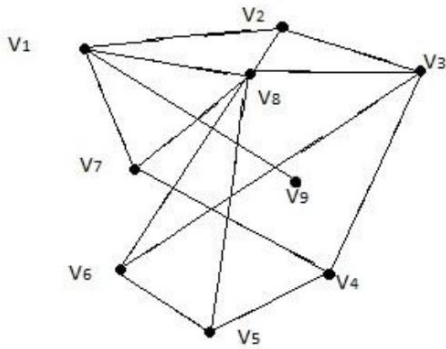
6) добуток графів



Завдання № 2

Скласти таблицю суміжності для орграфа.

9)



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

Завдання № 3

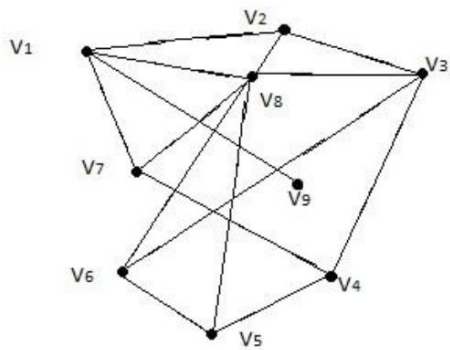
Для графа з другого завдання знайти діаметр.

Діаметр = 3

Завдання № 4

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

9)



Обхід дерева вглиб:

Вершина	Стек
V1	V1
V2	V1V2
V3	V1V2V3
V4	V1V2V3V4
V5	V1V2V3V4V5
V6	V1V2V3V4V5V6
V8	V1V2V3V4V5V6V8
V7	V1V2V3V4V5V6V8V7
-	V1V2V3V4V5V6V8
-	V1V2V3V4V5V6
-	V1V2V3V4V5
-	V1V2V3V4
-	V1V2V3
-	V1V2
-	V1
V9	V1V9
-	V1
-	Ø

Програмна реалізація:

```
1  #include <iostream>
2  using namespace std;
3  const int n = 9;
4  int i, j;
5  bool* visited = new bool[n];
6  int graph[n][n] =
7  {
8      {0,1,0,0,0,0,1,1,1},
9      {1,0,1,0,0,0,0,1,0},
10     {0,1,0,1,0,1,0,1,0},
11     {0,0,1,0,1,0,1,0,0},
12     {0,0,0,1,0,1,0,1,0},
13     {0,0,1,0,1,0,0,1,0},
14     {1,0,0,1,0,0,0,1,0},
15     {1,1,1,0,1,1,1,0,0},
16     {1,0,0,0,0,0,0,0,0}
17 };
18 void DFS(int st)
19 {
20     int r;
21     cout << st + 1 << " ";
22     visited[st] = true;
23     for (r = 0; r <= n; r++)
24         if ((graph[st][r] != 0) && (!visited[r]))
25             DFS(r);
26 }
27 void main()
28 {
29     int start;
30     cout << "Matrix: " << endl;
31     for (i = 0; i < n; i++)
32     {
33         visited[i] = false;
34         for (j = 0; j < n; j++)
35             cout << " " << graph[i][j];
36         cout << endl;
37     }
38     cout << "Start edge >> "; cin >> start;
39     bool* vis = new bool[n];
40     cout << "Path: ";
41     DFS(start - 1);
42     delete[] visited;
43 }
```

Апробація програми:

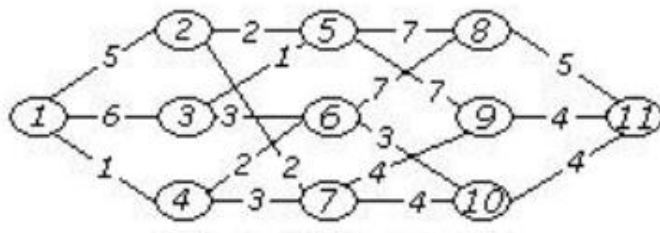
Microsoft Visual Studio Debug (

```
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 >> 1
Path: 1 2 3 4 5 6 8 7 9
```

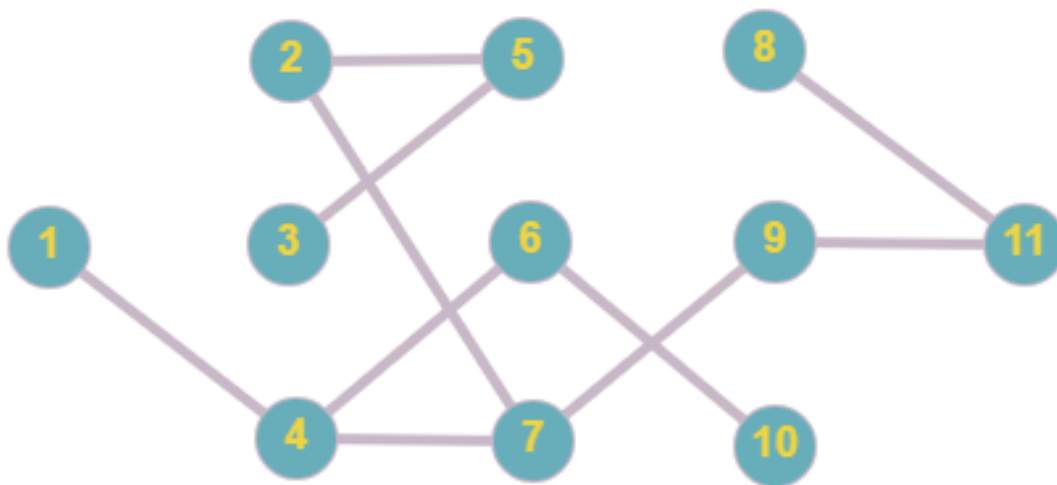
Завдання № 5

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

9)



Прима:



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

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

Програмна реалізація:


```

1   using namespace std;
2   #include <stdio.h>
3   #include <iostream>
4
5   #define V 11
6
7   int graph[V][V] = {
8       {0, 5, 6, 1, 0, 0, 0, 0, 0, 0, 0},
9       {5, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0},
10      {6, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0},
11      {1, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0},
12      {0, 2, 1, 0, 0, 0, 0, 7, 7, 0, 0},
13      {0, 0, 3, 2, 0, 0, 0, 7, 0, 3, 0},
14      {0, 2, 0, 3, 0, 0, 0, 0, 4, 4, 0},
15      {0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 5},
16      {0, 0, 0, 0, 7, 0, 4, 0, 0, 0, 4},
17      {0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 4},
18      {0, 0, 0, 0, 0, 0, 0, 0, 5, 4, 4}
19  };
20
21  int main() {
22
23      int no_edge;
24      int selected[V];
25      memset(selected, false, sizeof(selected));
26
27      no_edge = 0;
28

```

```

28      selected[0] = true;
29
30      int x;
31      int y;
32
33      cout << "Edge" << " : " << "Weight";
34      cout << endl;
35      while (no_edge < V - 1) {
36
37          int min = INT_MAX;
38          x = 0;
39          y = 0;
40
41          for (int i = 0; i < V; i++) {
42              if (selected[i]) {
43                  for (int j = 0; j < V; j++) {
44                      if (!selected[j] && graph[i][j]) {
45                          if (min > graph[i][j]) {
46                              min = graph[i][j];
47                              x = i;
48                              y = j;
49                          }
50                      }
51                  }
52              }
53          }
54
55          cout << x + 1 << " - " << y + 1 << " : " << graph[x][y];
56          cout << endl;
57          selected[y] = true;
58          no_edge++;
59      }
60
61      return 0;
62
63  }
64

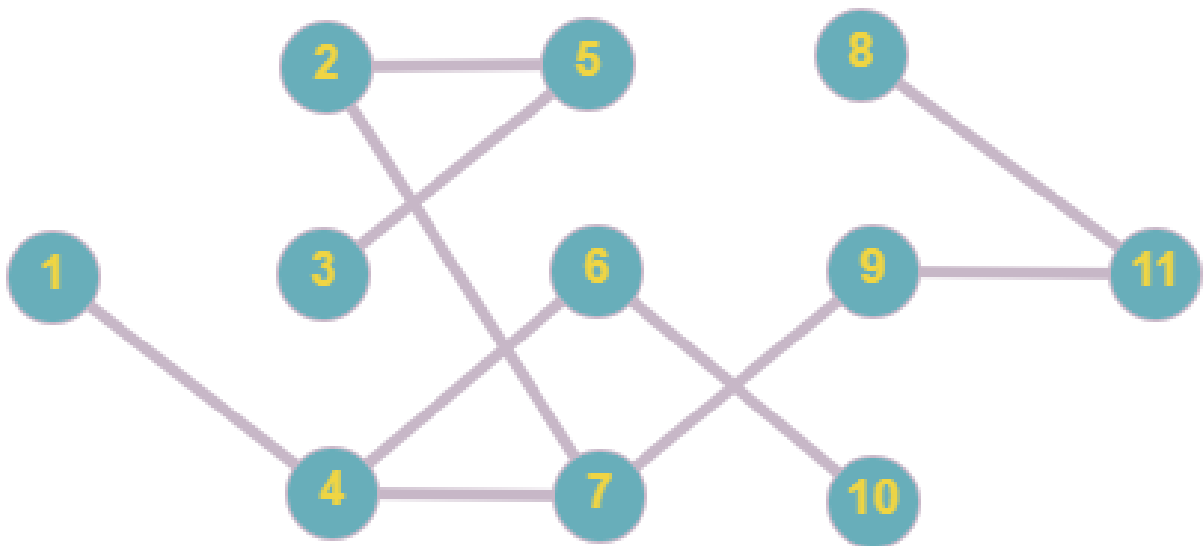
```

Апробація програми :

Microsoft Visual S

```
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
```

Краскала:



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

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

Програмна реалізація:

```
1  #include <iostream>
2  #define INT_MAX 2147483647
3
4  using namespace std;
5
6  #define V 11
7  int parent[V];
8
9  int find(int i) {
10     while (parent[i] != i)
11         i = parent[i];
12     return i;
13 }
14 void union1(int i, int j) {
15     int a = find(i);
16     int b = find(j);
17     parent[a] = b;
18 }
19 void Kruskal(int cost[][V]) {
20     cout << "The Kruskal Method" << endl;
21     int min_cost = 0;
22
23     for (int i = 0; i < V; i++)
24         parent[i] = i;
25
26     int edge_count = 0;
27     while (edge_count < V - 1) {
28         int min = INT_MAX, a = -1, b = -1;
29         for (int i = 0; i < V; i++) {
30             for (int j = 0; j < V; j++) {
31                 if (find(i) != find(j) && cost[i][j] < min) {
32                     min = cost[i][j];
33                     a = i;
34                     b = j;
35                 }
36             }
37         }
38         if (min == INT_MAX) {
39             cout << "There is no minimum spanning tree." << endl;
40             exit(0);
41         }
42         union1(a, b);
43         cout << "Edge" << edge_count++ + 1 << ":" << "(" << a + 1 << '-' << b + 1 << ")" << " \t cost:" << min << endl;
44         min_cost += min;
45     }
46     cout << "\n Minimum cost=" << min_cost << endl;
47 }
48
49 int main() {
50     int cost[][V] = {
51         {INT_MAX, 5, 6, 1, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX}, //1
52         {5, INT_MAX, INT_MAX, INT_MAX, 2, INT_MAX, 2, INT_MAX, INT_MAX, INT_MAX, INT_MAX}, //2
53         {6, INT_MAX, INT_MAX, INT_MAX, 1, 3, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX}, //3
54         {1, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 2, 3, INT_MAX, INT_MAX, INT_MAX, INT_MAX}, //4
55         {INT_MAX, 2, 1, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 7, 7, INT_MAX, INT_MAX}, //5
56         {INT_MAX, INT_MAX, 3, 2, INT_MAX, INT_MAX, INT_MAX, 7, INT_MAX, 3, INT_MAX}, //6
57         {INT_MAX, 2, INT_MAX, 3, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 4, 4, INT_MAX}, //7
58         {INT_MAX, INT_MAX, INT_MAX, INT_MAX, 7, 7, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 5}, //8
59         {INT_MAX, INT_MAX, INT_MAX, INT_MAX, 7, INT_MAX, 4, INT_MAX, INT_MAX, INT_MAX, 4}, //9
60         {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 3, 4, INT_MAX, INT_MAX, INT_MAX, 4}, //10
61         {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 5, 4, 4, INT_MAX}, //11
62     };
63
64     Kruskal(cost);
65     return 0;
66 }
```

Апробація програми:

```
The Kruskal Method
Edge1:(1-4)      cost:1
Edge2:(3-5)      cost:1
Edge3:(2-5)      cost:2
Edge4:(2-7)      cost:2
Edge5:(4-6)      cost:2
Edge6:(3-6)      cost:3
Edge7:(6-10)     cost:3
Edge8:(7-9)      cost:4
Edge9:(9-11)     cost:4
Edge10:(8-11)    cost:5

Minimum cost=27
```

Завдання № 6

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

9)

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

Вершина	Маршрут	Вага
-	1	0
8	1-8	1
5	1-8-7	3
4	1-8-7-4	4
7	1-8-7-4-5	5
2	1-8-7-4-5-2	7
6	1-8-7-4-5-2-6	8
3	1-8-7-4-5-2-6-3	14

Програмна реалізація:

```

1  #include<iostream>
2  #include<vector>
3
4  using namespace std;
5  #define V 8
6
7  int counter = 0, Minimal_way = INT_MAX;
8
9  bool check(vector<int> v, int Node) {
10     for (auto i = v.begin(); i != v.end(); i++)
11         if (*i == Node)
12             return false;
13     return true;
14 }
15
16 int minimum(vector<int>* v, int graph[][8], int i) {
17     int min = INT_MAX;
18     for (int j = 0; j < V; j++) {
19         if (graph[i][j] < min && graph[i][j] != 0 && check(*v, j)) min = graph[i][j];
20     }
21     return min;
22 }
23
24 void search(vector<int>* v, int graph[][8], int pos, vector<int>* res) {
25     int min, i = pos;
26     min = minimum(v, graph, i);
27     for (int j = 0; j < V; j++)
28         if (graph[i][j] == min && check(*v, j)) {
29             (*v).push_back(j);
30             search(v, graph, j, res);
31         }
32     if (v->size() == V) {
33         (*v).push_back((*v)[0]);
34         counter = 0;
35         for (int l = 1; l <= V; l++)
36             counter += graph[(v->size() - 1)][(v->size() - 1)];
37         if (Minimal_way == counter) {
38             for (int op = 0; op <= V; op++)
39                 (*res).push_back((*v)[op]);
40             (*res).push_back(counter);
41         }
42         else if (Minimal_way > counter) {
43             (*res).clear();
44             for (int op = 0; op <= V; op++)
45                 (*res).push_back((*v)[op]);
46             (*res).push_back(counter);
47             Minimal_way = counter;
48         }
49         v->pop_back();
50     }
51     v->pop_back();
52 }
53
54 int main()
55 {
56     vector<int> v;
57     vector<int> result;
58
59     int graph[][V] = {
60         {0, 5, 5, 3, 3, 4, 4, 1},
61         {5, 0, 4, 3, 2, 1, 4, 6},
62         {5, 4, 0, 4, 5, 6, 5, 5},
63         {3, 3, 4, 0, 1, 5, 1, 7},
64         {3, 2, 5, 1, 0, 5, 5, 2},
65         {4, 1, 6, 5, 5, 0, 7, 3},
66         {4, 4, 5, 1, 5, 7, 0, 2},
67         {1, 6, 5, 7, 2, 3, 2, 0},
68     };
69
70     for (int i = 0; i < V; i++) {
71         v.clear();
72         v.push_back(i);
73         search(&v, graph, i, &result);
74     }
75     cout << "The shortest way is: " << endl;
76     for (int i = 1; i <= 10; i++) {
77         if (i != 0 && i % (V + 2) == 0)
78             cout << "\n" << "The weight of path is: " << result[i - 1] << endl;
79         else
80             cout << result[i - 1] + 1 << " ";
81     }
82 }

```

Апробація програми:

Microsoft Visual Studio Debug Console

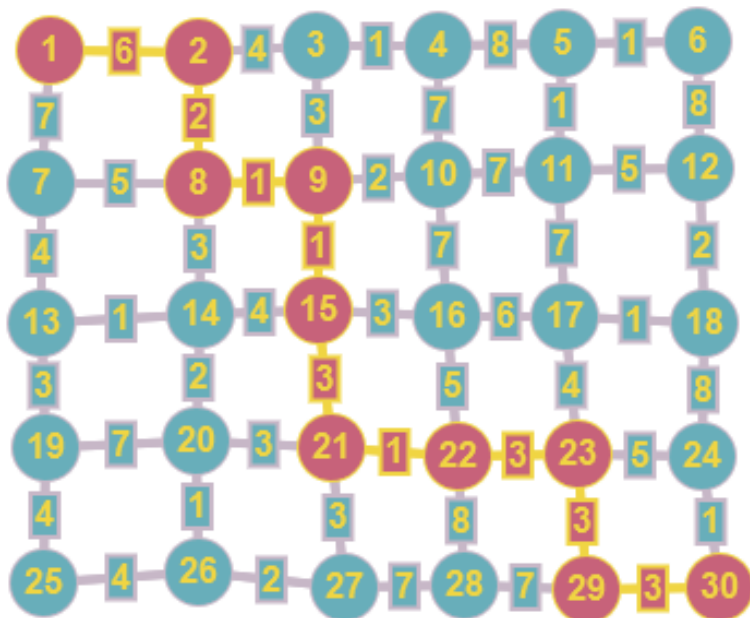
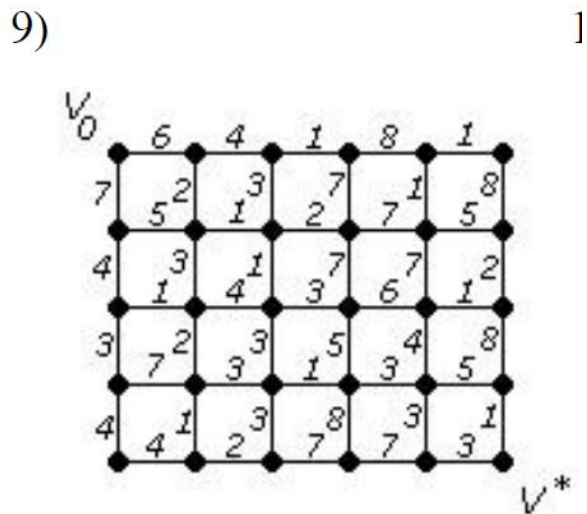
The shortest way is:

1 8 7 4 5 2 6 3 1

The weight of path is: 19

Завдання № 7

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



Програмна реалізація:

```
1  #include <iostream>
2  #include <stdio.h>
3  using namespace std;
4  #define inf INT_MAX
5  #define nodes 30
6
7  void dijkstra(double Graph[nodes][nodes], int n, int startNode) {
8      double cost[nodes][nodes], dist[nodes], pred[nodes], minDist, visited[nodes];
9      int counter, nextNode, i, j;
10     for ( i = 0; i < n; i++) {
11         for ( j = 0; j < n; j++) {
12             if (Graph[i][j] == 0) {
13                 cost[i][j] = inf;
14             }
15             else {
16                 cost[i][j] = Graph[i][j];
17             }
18         }
19     }
20     for ( i = 0; i < n; i++) {
21         dist[i] = cost[startNode][i];
22         pred[i] = startNode;
23         visited[i] = 0;
24     }
25     dist[startNode] = 0;
26     visited[startNode] = 1;
27     counter = 1;
28
29     while (counter < n-1)
30     {
31         minDist = inf;
32         for (i = 0; i < n; i++) {
33             if (dist[i] < minDist && !visited[i]) {
34                 minDist = dist[i];
35                 nextNode = i;
36             }
37         }
38         visited[nextNode] = 1;
39         for ( i = 0; i < n; i++) {
40             if (!visited[i]) {
41                 if (minDist + cost[nextNode][i] < dist[i]) {
42                     dist[i] = minDist + cost[nextNode][i];
43                     pred[i] = nextNode;
44                 }
45             }
46         }
47         counter++;
48     }
49     for (i = 0; i < n; i++) {
50         if (i != startNode) {
51             cout << "Distance of node" << i + 1 << " = " << dist[i];
52             cout << "Path=" << i + 1;
53             j = i;
54             do {
55                 j = pred[j];
56                 cout << "<- " << j + 1;
57             } while (j != startNode);
58         }
59     }
60 }
61
62 int main()
63 {
64     double G[nodes][nodes];
65     for (int i = 0; i < nodes; i++) {
66         for (int j = 0; j < nodes; j++) {
67             G[i][j] = 0;
68         }
69     }
70
71     int i = 0, j = 1;
72     do {
73         cout << "Enter weight from " << i + 1 << " to " << j + 1 << endl;
74         cin >> G[i][j];
75         i++;
76         j++;
77     } while (j < nodes);
78     i = nodes - 1;
79     j = i - 6;
80
81     do {
82         cout << "Enter weight from " << i + 1 << " to " << j + 1 << endl;
83         cin >> G[j][i];
84         i--;
85         j = i - 6;
86     } while (j >= 0);
87
88     for (i = 0; i < nodes; i++) {
89         for (j = 0; j < nodes; j++) {
90             G[j][i] = G[i][j];
91             G[j][i] = G[i][j];
92         }
93     }
94
95
96     int n = 30;
97     int u = 0;
98     dijkstra(G, n, u);
99 }
```

Апробація програми:

```
Microsoft Visual Studio Debug Console
Enter weight from 1 to 2
6
Enter weight from 2 to 3
1
Enter weight from 3 to 4
3
Enter weight from 4 to 5
1
Enter weight from 5 to 6
3
Enter weight from 6 to 7
0
Enter weight from 7 to 8
4
Enter weight from 8 to 9
1
Enter weight from 9 to 10
1
Enter weight from 10 to 11
2
Enter weight from 11 to 12
7
Enter weight from 12 to 13
0
Enter weight from 13 to 14
4
Enter weight from 14 to 15
1
Enter weight from 15 to 16
2
Enter weight from 16 to 17
3
Enter weight from 17 to 18
7
Enter weight from 18 to 19
0
Enter weight from 19 to 20
7
Enter weight from 20 to 21
3
Enter weight from 21 to 22
8
Enter weight from 22 to 23
1
Enter weight from 23 to 24
5

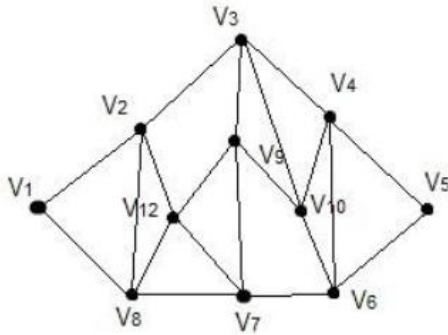
Microsoft Visual Studio Debug Console
Enter weight from 24 to 25
0
Enter weight from 25 to 26
4
Enter weight from 26 to 27
7
Enter weight from 27 to 28
3
Enter weight from 28 to 29
3
Enter weight from 29 to 30
6
Enter weight from 30 to 24
8
Enter weight from 29 to 23
3
Enter weight from 28 to 22
1
Enter weight from 27 to 21
3
Enter weight from 26 to 20
2
Enter weight from 25 to 19
7
Enter weight from 24 to 18
8
Enter weight from 23 to 17
2
Enter weight from 22 to 16
7
Enter weight from 21 to 15
7
Enter weight from 20 to 14
1
Enter weight from 19 to 13
5
Enter weight from 18 to 12
3
Enter weight from 17 to 11
1
Enter weight from 16 to 10
7
Enter weight from 15 to 9
5
Enter weight from 14 to 8
1

Microsoft Visual Studio Debug Console
Enter weight from 13 to 7
4
Enter weight from 12 to 6
8
Enter weight from 11 to 5
5
Enter weight from 10 to 4
4
Enter weight from 9 to 3
3
Enter weight from 8 to 2
2
Enter weight from 7 to 1
4

Distance of node2=6
Path=2<-1
Distance of node3=7
Path=3<-2<-1
Distance of node4=10
Path=4<-3<-2<-1
Distance of node5=11
Path=5<-4<-3<-2<-1
Distance of node6=14
Path=6<-5<-4<-3<-2<-1
Distance of node7=4
Path=7<-1
Distance of node8=8
Path=8<-7<-1
Distance of node9=9
Path=9<-8<-7<-1
Distance of node10=10
Path=10<-9<-8<-7<-1
Distance of node11=12
Path=11<-10<-9<-8<-7<-1
Distance of node12=19
Path=12<-11<-10<-9<-8<-7<-1
Distance of node13=8
Path=13<-7<-1
Distance of node14=9
Path=14<-8<-7<-1
Distance of node15=10
Path=15<-14<-8<-7<-1
Distance of node16=12
Path=16<-15<-14<-8<-7<-1
Distance of node17=13
Path=17<-11<-10<-9<-8<-7<-1
Distance of node18=20
Path=18<-17<-11<-10<-9<-8<-7<-1
Distance of node19=13
Path=19<-13<-7<-1
Distance of node20=10
Path=20<-14<-8<-7<-1
Distance of node21=13
Path=21<-20<-14<-8<-7<-1
Distance of node22=16
Path=22<-23<-17<-11<-10<-9<-8<-7<-1
Distance of node23=15
Path=23<-17<-11<-10<-9<-8<-7<-1
Distance of node24=20
Path=24<-23<-17<-11<-10<-9<-8<-7<-1
Distance of node25=16
Path=25<-26<-20<-14<-8<-7<-1
Distance of node26=12
Path=26<-20<-14<-8<-7<-1
Distance of node27=16
Path=27<-21<-20<-14<-8<-7<-1
Distance of node28=17
Path=28<-22<-23<-17<-11<-10<-9<-8<-7<-1
Distance of node29=18
Path=29<-23<-17<-11<-10<-9<-8<-7<-1
Distance of node30=24
Path=30<-29<-23<-17<-11<-10<-9<-8<-7<-1
```


Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.



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

Програмна реалізація:

```
1  #include<iostream>
2  #define NODE 11
3  using namespace std;
4  int graph[NODE][NODE] = {
5      {0,1,0,0,0,0,0,1,0,0,0},
6      {1,0,1,0,0,0,0,1,0,0,1},
7      {0,1,0,1,0,0,0,0,1,1,0},
8      {0,0,1,0,1,1,0,0,0,1,0},
9      {0,0,0,1,0,1,0,0,0,0,0},
10     {0,0,0,1,1,0,1,0,0,1,0},
11     {0,0,0,0,0,1,0,1,1,0,1},
12     {1,1,0,0,0,0,1,0,0,0,1},
13     {0,0,1,0,0,0,1,0,0,1,1},
14     {0,0,1,1,0,1,0,0,1,0,0},
15     {0,1,0,0,0,0,1,1,1,0,0}
16 };
17 int tempGraph[NODE][NODE];
18 int findStartVert() {
19     for (int i = 1; i < NODE; i++) {
20         int deg = 0;
21         for (int j = 0; j < NODE; j++) {
22             if (tempGraph[i][j])
23                 deg++;
24         }
25         if (deg % 2 != 0)
26             return i;
27     }
28     return 0;
29 }
30 bool isBridge(int u, int v) {
31     int deg = 0;
32     for (int i = 0; i < NODE; i++)
33         if (tempGraph[v][i])
34             deg++;
35     if (deg > 1) {
36         return false;
37     }
38     return true;
39 }
40 int edgeCount() {
41     int count = 0;
42     for (int i = 0; i < NODE; i++)
43         for (int j = i; j < NODE; j++)
44             if (tempGraph[i][j])
45                 count++;
46     return count;
47 }
48 void fleuryAlgorithm(int start) {
49     static int edge = edgeCount();
50     for (int v = 0; v < NODE; v++) {
51         if (tempGraph[start][v]) {
52             if (edge <= 1 || !isBridge(start, v)) {
53                 cout << start + 1 << "--" << v + 1 << " " << endl;
54                 tempGraph[start][v] = tempGraph[v][start] = 0;
55                 edge--;
56                 fleuryAlgorithm(v);
57             }
58         }
59     }
60 }
61 int main() {
62     for (int i = 0; i < NODE; i++)
63         for (int j = 0; j < NODE; j++)
64             tempGraph[i][j] = graph[i][j];
65     cout << "Euler Path Or Circuit: \n";
66     fleuryAlgorithm(findStartVert());
67 }
```

Апробація програми:

```
Microsoft Visual Studio Debug
Euler Path Or Circuit:
1--2
2--3
3--4
4--5
5--6
6--4
4--10
10--3
3--9
9--7
7--6
6--10
10--9
9--11
11--2
2--8
8--7
7--11
11--8
8--1
```

б)

1) $1 - 2 - 8 - 1 +$

2) $2 - 3 - 9 - 12 - 2 +$

3) $12 - 8 - 7 - 12$

4) $7 - 9 - 10 - 6 - 7$

5) $3 - 10 - 4 - 3$

5) $6 - 4 - 5 - 6$

- $1 - 2 - 8 - 1$
- $1 - 2 - 3 - 9 - 12 - 2 - 8 - 1$
- $1 - 2 - 3 - 9 - 12 - 2 - 8 - 12 - 7 - 9 - 10 - 6 - 7 - 8 - 1$
- $1 - 2 - 3 - 9 - 12 - 2 - 8 - 12 - 7 - 9 - 10 - 6 - 4 - 10 - 3 - 4 - 5 - 6 - 7 - 8 - 1$

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

$$9. (x \rightarrow y) \cdot (y \rightarrow z) \rightarrow (x \rightarrow z).$$

$$(\neg x \vee y)(\neg y \vee z) \rightarrow (x \rightarrow z)$$

$$(\neg x \neg y \vee \neg x z \vee y z) \rightarrow (x \rightarrow z)$$

$$(\neg x \neg y \vee y z) \rightarrow (\neg x \vee z)$$

$$\neg(\neg x \neg y \vee y z) \vee \neg x z$$

$$((x \vee y)(\neg y \vee \neg z)) \vee \neg x \vee z$$

$$x \neg y \vee x \neg z \vee y \neg y \vee y \neg z \vee \neg x \vee z$$

$$x \neg y \vee x \neg z \vee y \neg z \vee \neg x \vee z$$

$$y \neg z \vee x \neg y \vee \neg x \vee z$$

$$\neg x \vee \neg y \vee \neg z \vee z$$

$$\neg x \vee \neg y \vee 1$$