

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
“ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Розрахунково-графічна робота**

з дисципліни

«Дискретна математика»

**Виконав:**

студент групи КН-112

Весна Ігор

**Викладач:**

Мельникова Н.І.

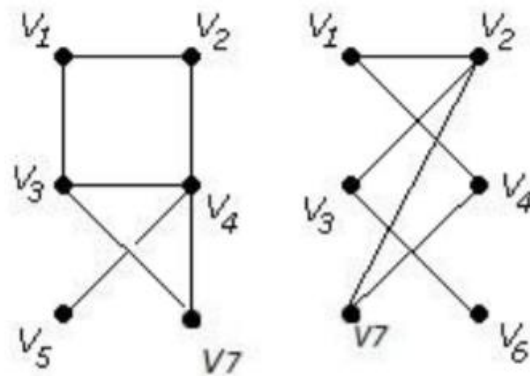
Львів – 2019 р.

# ІНДИВІДУАЛЬНІ ЗАВДАННЯ

## Завдання № 1

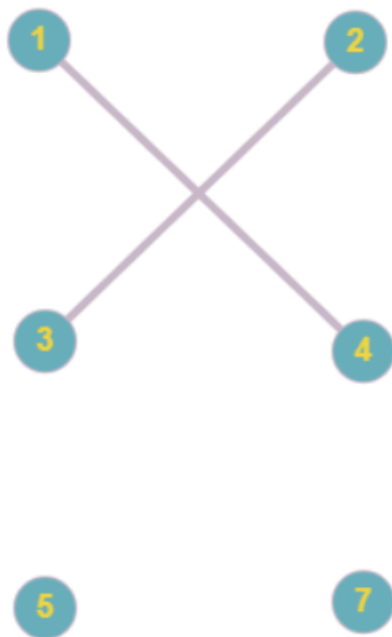
Виконати наступні операції над графами: 1) знайти доповнення до першого графу, 2) об'єднання графів, 3) кільцеву сумму  $G1$  та  $G2$  ( $G1+G2$ ), 4) розмножити вершину у другому графі, 5) виділити підграф  $A$  - що складається з 3-х вершин в  $G1$  6) добуток графів.

8)

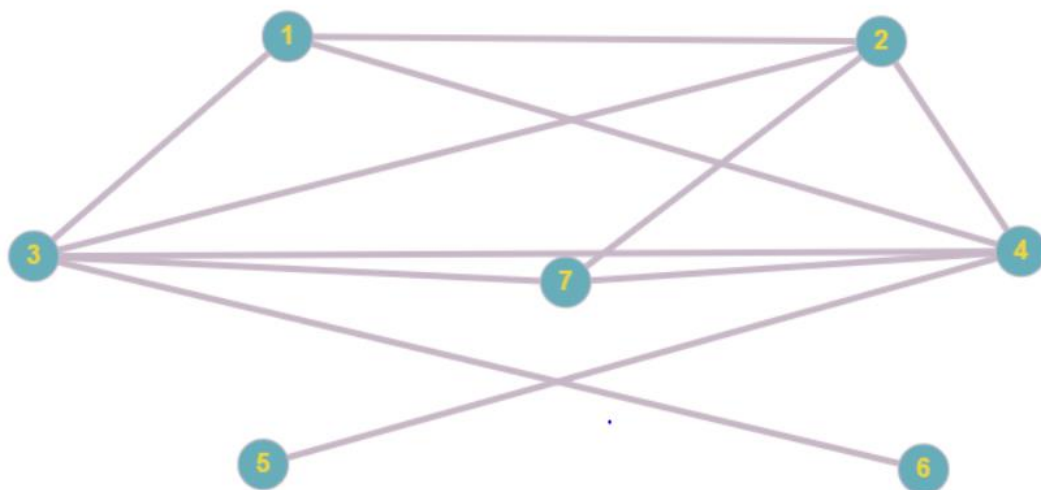


1)Доповнення

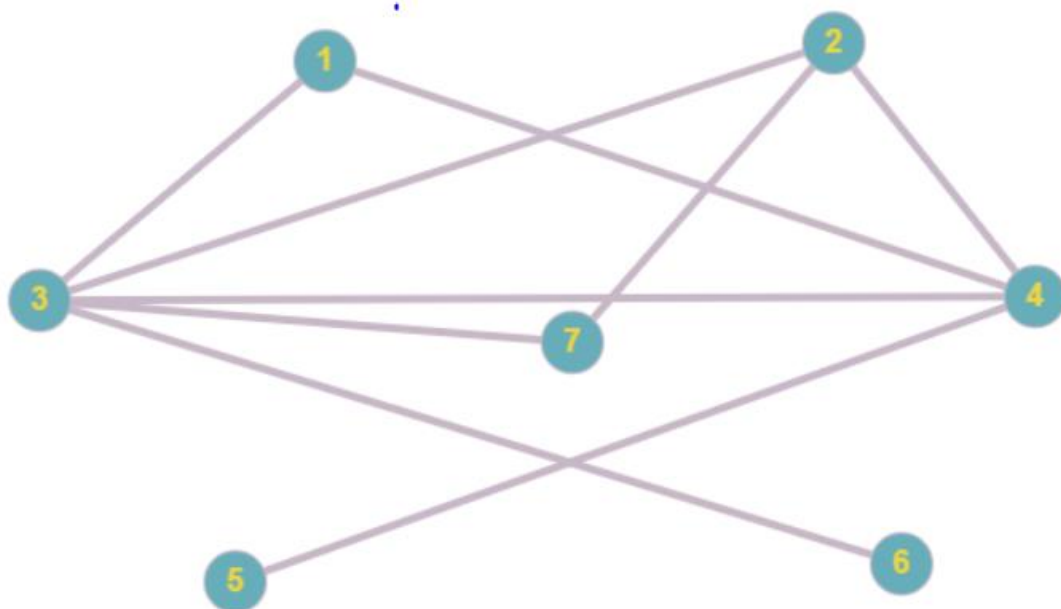
$G1 \setminus G2$ :



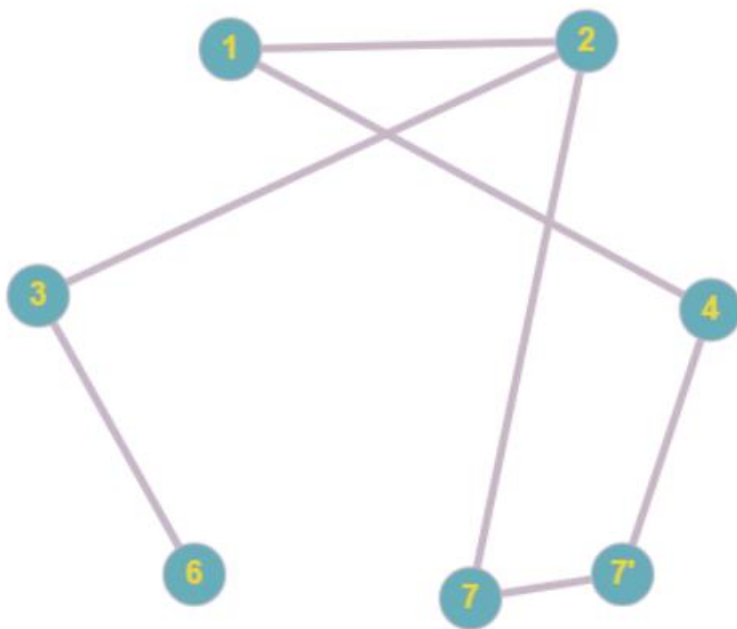
2)об'єднання графів:



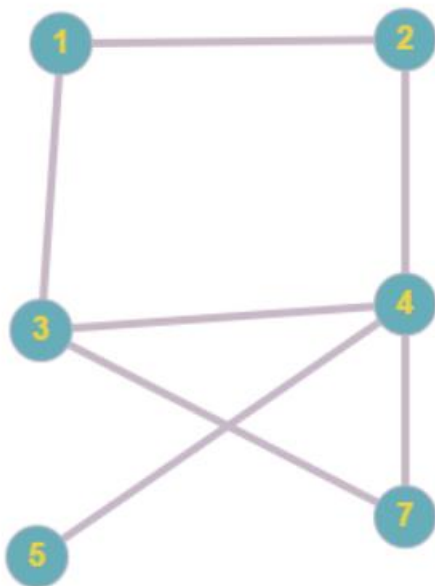
3)Кільцева сума:



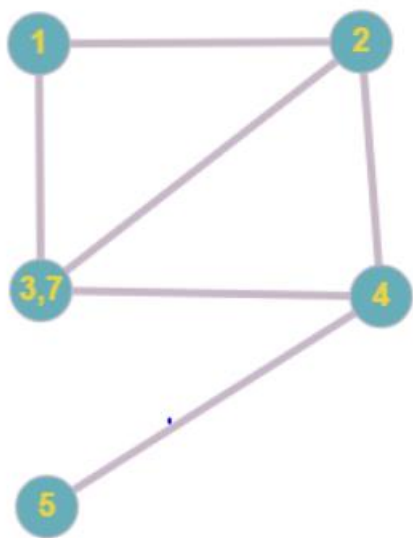
4)Розщепити вершину у другому графі:  
Розщепимо вершину 7



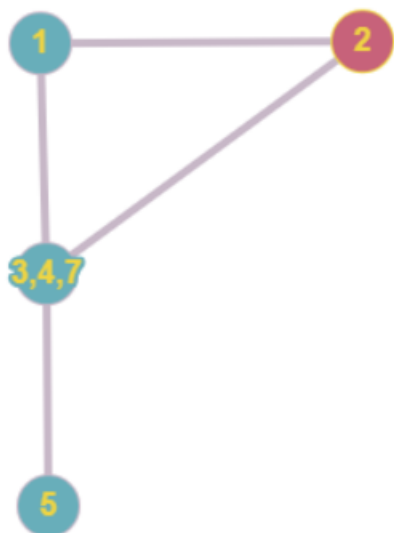
5) Виділити підграф A, що складається з  $V = \{3, 4, 7\}$  в  $G_1$  і знайти стягнення A в  $G_1$



Стягуємо 7 в 3



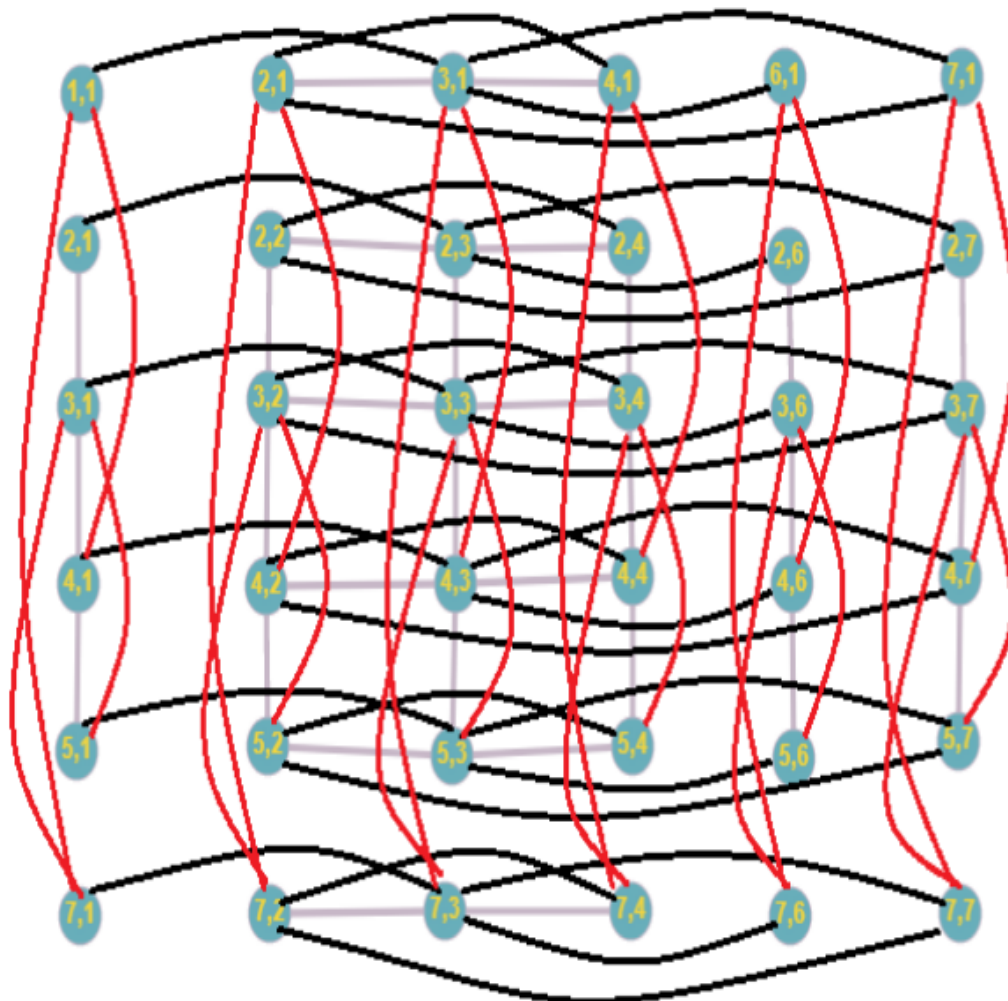
Стягуємо 3,7 в 4



Стягуємо 3,7,4 в 2



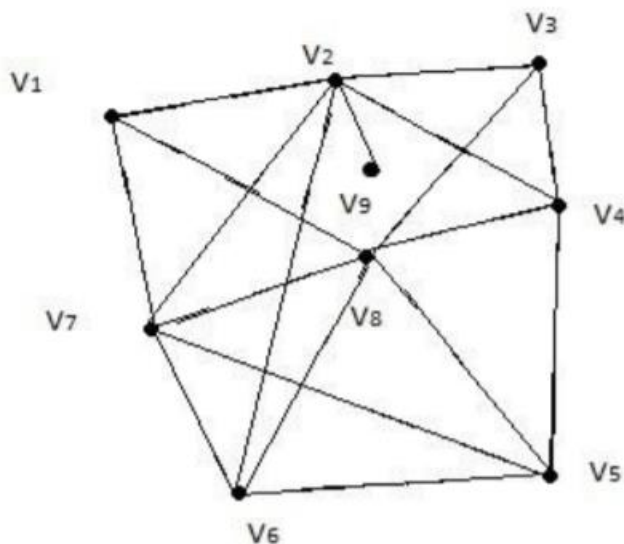
6) Добуток графів



### Завдання № 2

Скласти таблицю суміжності для графа.

8)



Таблиця суміжності:

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

### Завдання № 3

Для графа з другого завдання знайти діаметр.

Найдовший шлях від V5 до V9 і V8 до V9

Діаметр = 3

### Завдання № 4

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

Обхід вшир:

Номер	Номер вершини	Черга
0	-	-
1	1	1
2	2	12
3	8	128
4	7	1287
5	-	287
6	3	2873
7	4	28734
8	9	287349
9	6	2873496
10	-	873496
11	5	8734965
12	-	734965
13	-	34965
14	-	4965
15	-	965
16	-	65
17	-	5
18	-	-



```

1  #include<iostream>
2  #include<queue>
3  #define v 8
4  using namespace std;
5  class n {
6  public:
7      int val;
8      int st;
9  };
10 int matrix[v][v] = {
11     {0, 1, 1, 1, 0, 1, 1, 0},
12     {1, 0, 1, 1, 0, 0, 1, 1},
13     {1, 1, 0, 1, 0, 1, 1, 0},
14     {1, 1, 1, 0, 0, 0, 1, 0},
15     {0, 0, 0, 0, 0, 1, 7, 0},
16     {1, 0, 0, 0, 1, 0, 1, 0},
17     {1, 1, 1, 1, 1, 1, 0, 0},
18     {0, 1, 0, 0, 0, 0, 0, 0}
19 };
20 void bfs(n* verh, n s) {
21     n u;
22     int i, j;
23     queue<n> que;
24     for (i = 0; i < v; i++) {
25         verh[i].st = 0;
26     }
27     verh[s.val].st = 1;
28     que.push(s);
29     while (!que.empty()) {
30         u = que.front();
31         que.pop();
32         cout << u.val+1 << " ";
33         for (i = 0; i < v; i++) {
34             if (matrix[i][u.val]) {

```

```

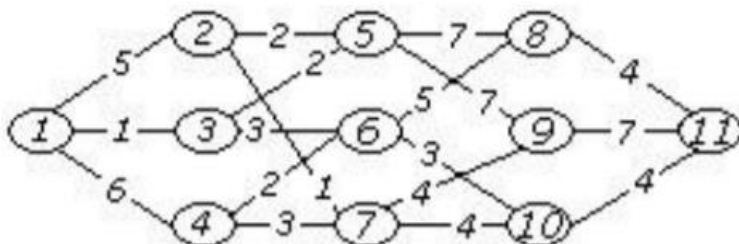
25         verh[i].st = 0;
26     }
27     verh[s.val].st = 1;
28     que.push(s);
29     while (!que.empty()) {
30         u = que.front();
31         que.pop();
32         cout << u.val+1 << " ";
33         for (i = 0; i < v; i++) {
34             if (matrix[i][u.val]) {
35                 if (verh[i].st == 0) {
36                     verh[i].st = 1;
37                     que.push(verh[i]);
38                 }
39             }
40         }
41         u.st = 2;
42     }
43 }
44
45 int main() {
46     n verh[v];
47     n start;
48     int s;
49     for (int i = 0; i < v; i++) {
50         verh[i].val = i;
51     }
52     s = 65;
53     start.val = s - 65;
54     cout << "bfs: ";
55     bfs(verh, start);
56     cout << endl;
57 }

```

### Завдання № 5

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

8)



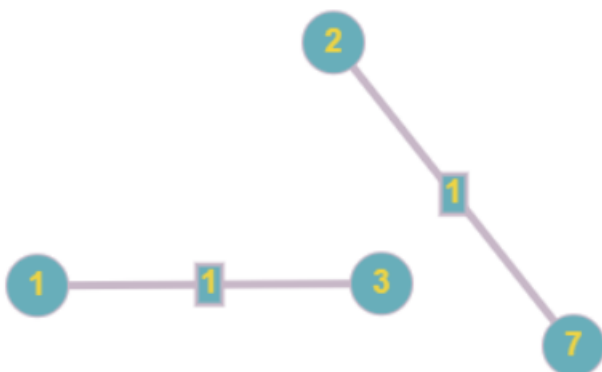
Метод Краскала  $V = \{1, 3\}$   $E = \{(1, 3)\}$




---

$V = \{1, 2, 3, 7, \}$

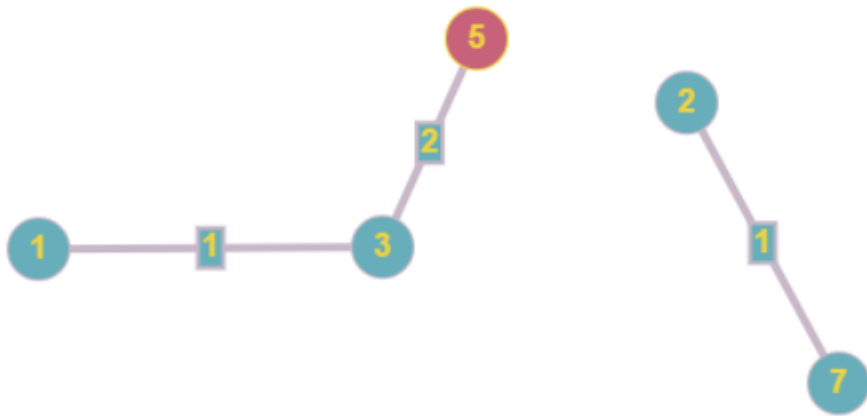
$E = \{(1, 3), (2, 7)\}$



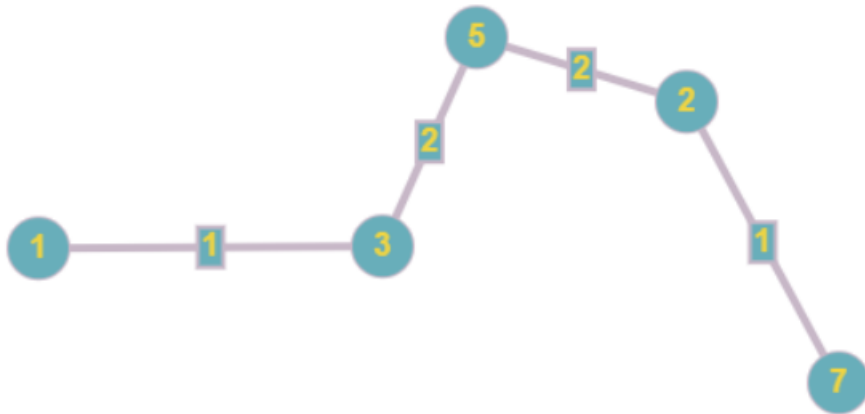

---

$V = \{1, 2, 5, 7, 9\}$

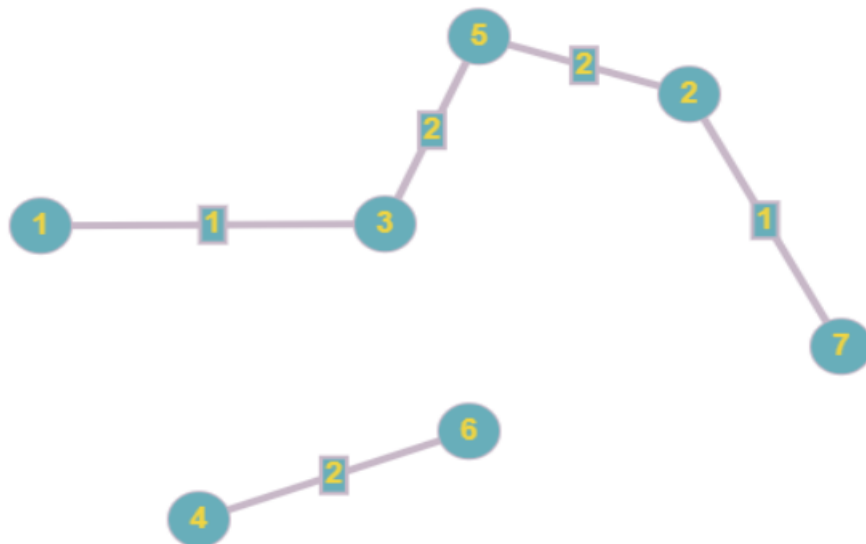
$E = \{(1, 3), (2, 7), (3, 5)\}$



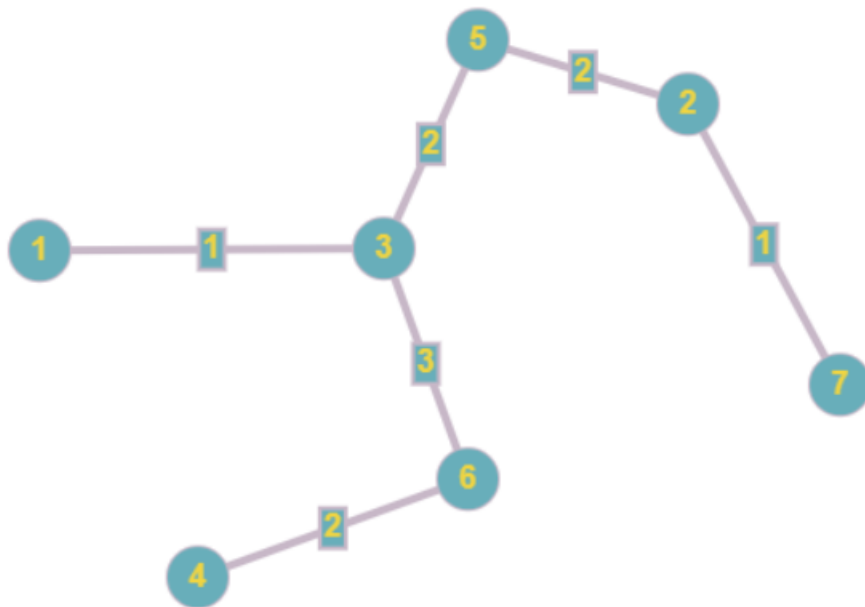
$V=\{1,3,5,2,7\}$   $E=\{(1,3),(3,5),(5,2),(2,7)\}$



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



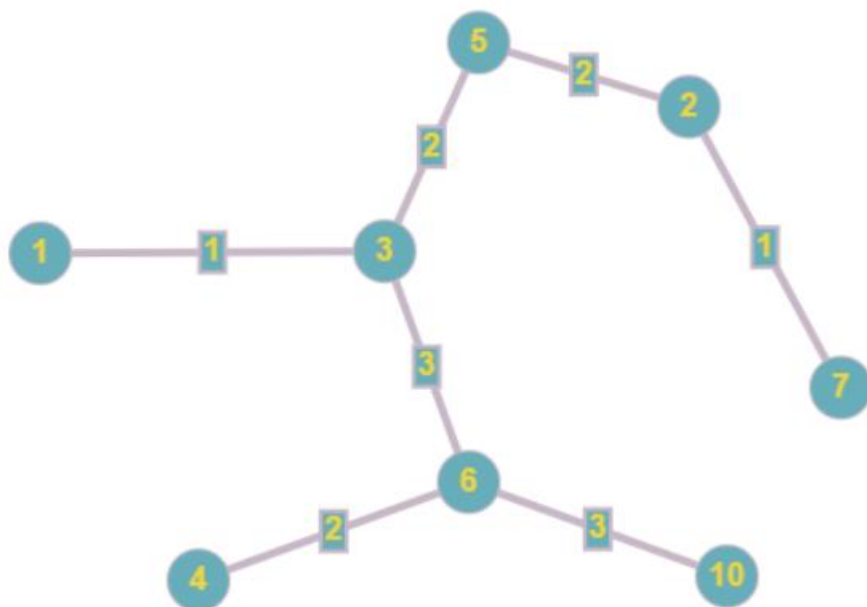
$V = \{1, 2, 7, 9, 5, 4, 6\}$   $E = \{(1, 3), (3, 5), (5, 2), (2, 7), (4, 6), (3, 6)\}$



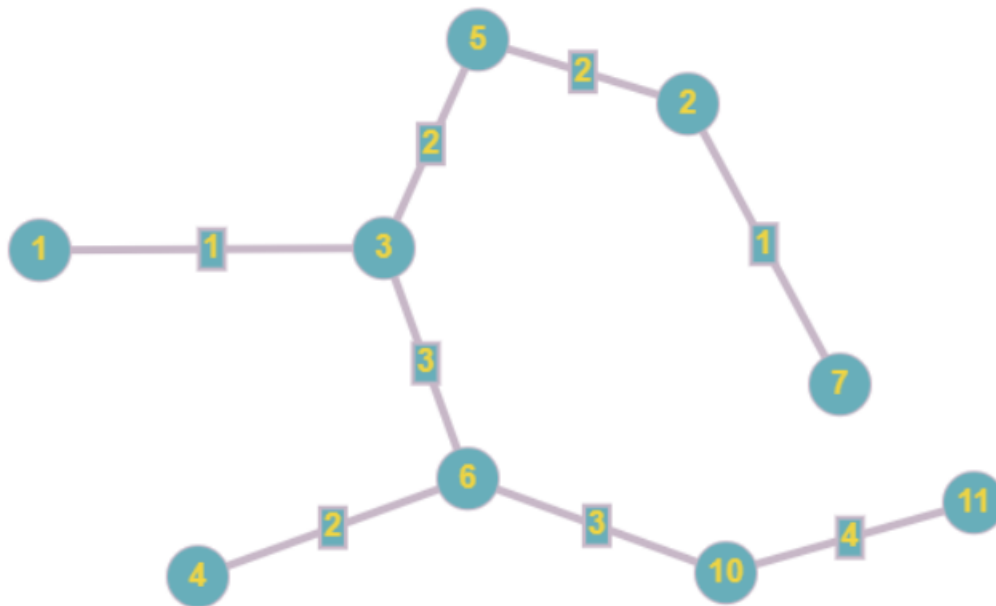
$\{(4, 7)\}$  - ЦИКЛ

---

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

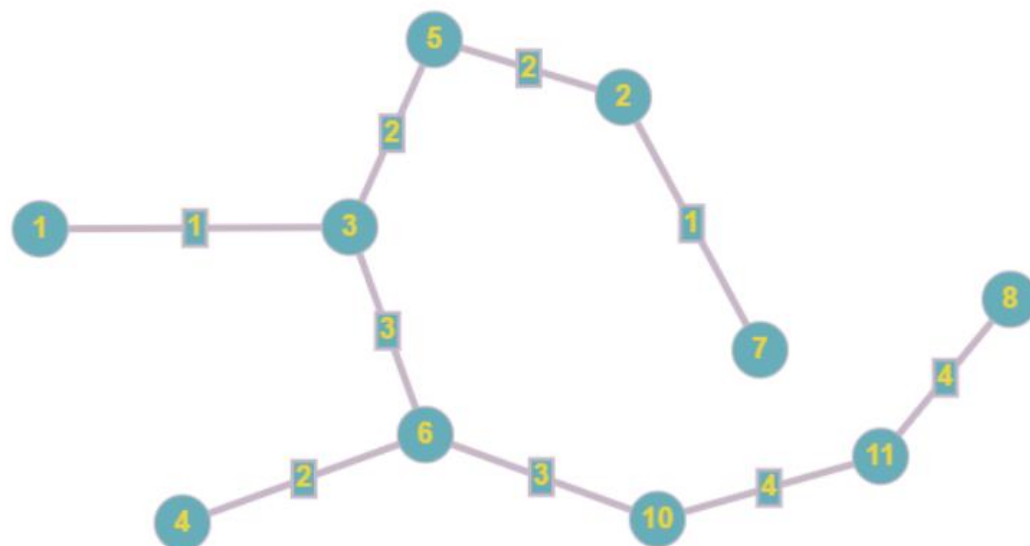


$V=\{1,2,7,9,5,4,6,10,11\}$   $E=\{(1,3),(3,5),(5,2),(2,7),(4,6),(3,6),(6,10),(10,11)\}$



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

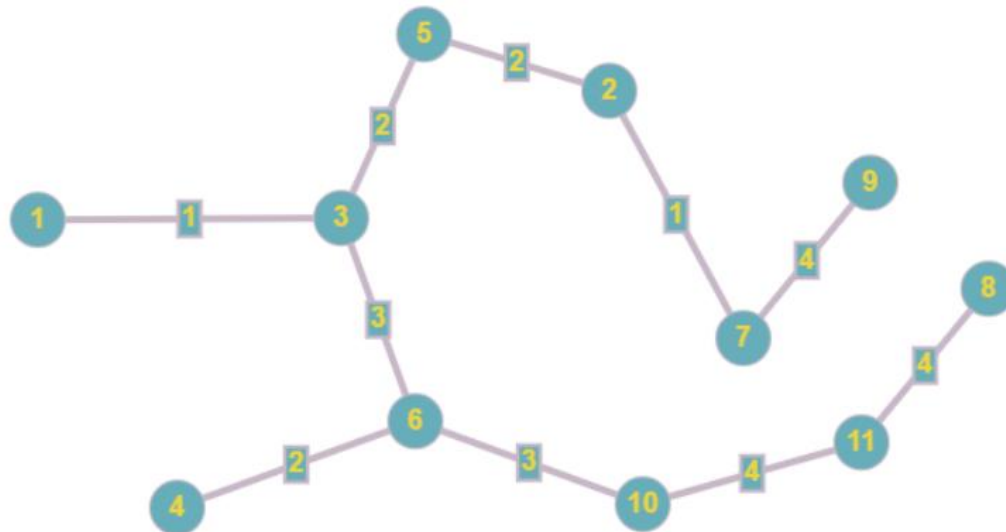
$E=\{(1,3),(3,5),(5,2),(2,7),(4,6),(3,6),(6,10),(10,11),(11,8)\}$




---

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

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



Bara=26

```
#include <iostream>
#define INT_MAX 2147483647

using namespace std;

#define V 11
int parent[V];

int find(int i) {
    while (parent[i] != i)
        i = parent[i];
    return i;
}

void union1(int i, int j) {
    int a = find(i);
    int b = find(j);
    parent[a] = b;
}

void Kruskal(int cost[][V]) {
    cout << "The Kruskal Method" << endl;
    int min_cost = 0;

    for (int i = 0; i < V; i++)
        parent[i] = i;

    int edge_count = 0;
    while (edge_count < V - 1) {
        int min = INT_MAX, a = -1, b = -1;
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                if (find(i) != find(j) && cost[i][j] < min) {
                    min = cost[i][j];
                    a = i;
                    b = j;
                }
            }
        }

        if (min == INT_MAX) {
            cout << "There is no minimum spanning tree." << endl;
            exit(0);
        }

        union1(a, b);
        printf("Edge %d-%d \t cost: %d \n",
            edge_count++ + 1, a + 1, b + 1, min);
        min_cost += min;
    }
    printf("\n Minimum cost= %d \n", min_cost);
}

int main() {
    int cost[][V] = {
        {INT_MAX, 7, 3, 2, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX,
          INT_MAX, INT_MAX},
        {7, INT_MAX, INT_MAX, INT_MAX, 2, INT_MAX, 1, INT_MAX, INT_MAX,
          INT_MAX, INT_MAX},
        {3, INT_MAX, INT_MAX, INT_MAX, 7, 4, INT_MAX, INT_MAX, INT_MAX,
          INT_MAX, INT_MAX},
        {2, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 5, 5, INT_MAX, INT_MAX,
          INT_MAX, INT_MAX},
        {INT_MAX, 2, 7, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 2, 4,
          INT_MAX, INT_MAX},
        {INT_MAX, INT_MAX, 4, 5, INT_MAX, INT_MAX, INT_MAX, 4, INT_MAX, 2,
          INT_MAX},
        {INT_MAX, 1, INT_MAX, 5, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 3, 4,
          INT_MAX},
        {INT_MAX, INT_MAX, INT_MAX, INT_MAX, 2, 4, INT_MAX, INT_MAX, INT_MAX,
          INT_MAX, 3},
        {INT_MAX, INT_MAX, INT_MAX, INT_MAX, 4, INT_MAX, 3, INT_MAX, INT_MAX,
          INT_MAX, 1},
        {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 2, 4, INT_MAX, INT_MAX,
          INT_MAX, 6},
        {INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, INT_MAX, 3, 1, 6,
          INT_MAX}
    };

    Kruskal(cost);
    return 0;
}
```

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

Метод Прима:

V={1,3} E={(1,3)}




---

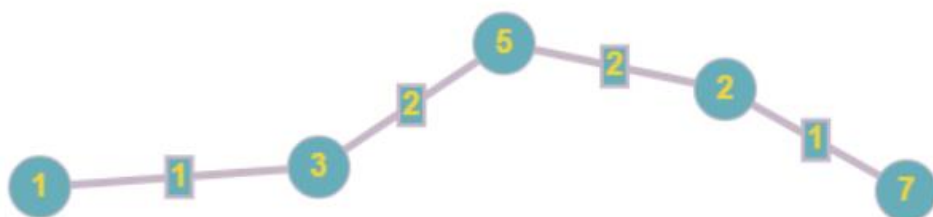
$V=\{1,3,5\}$   $E=\{(1,3),(3,5)\}$



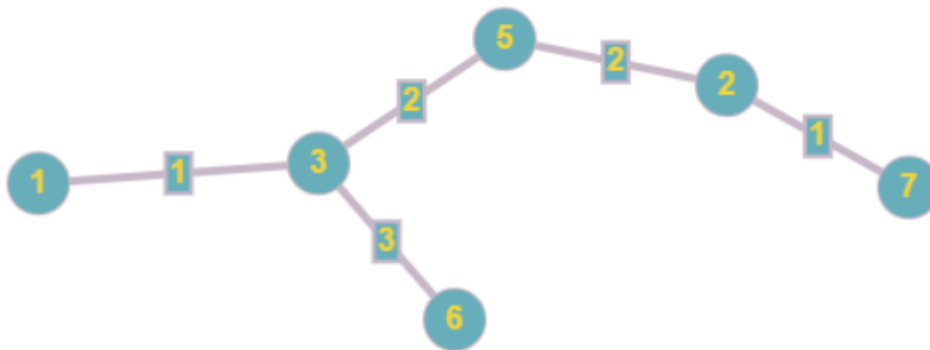
$V=\{1,3,5,2\}$   $E=\{(1,3),(3,5),(5,2)\}$



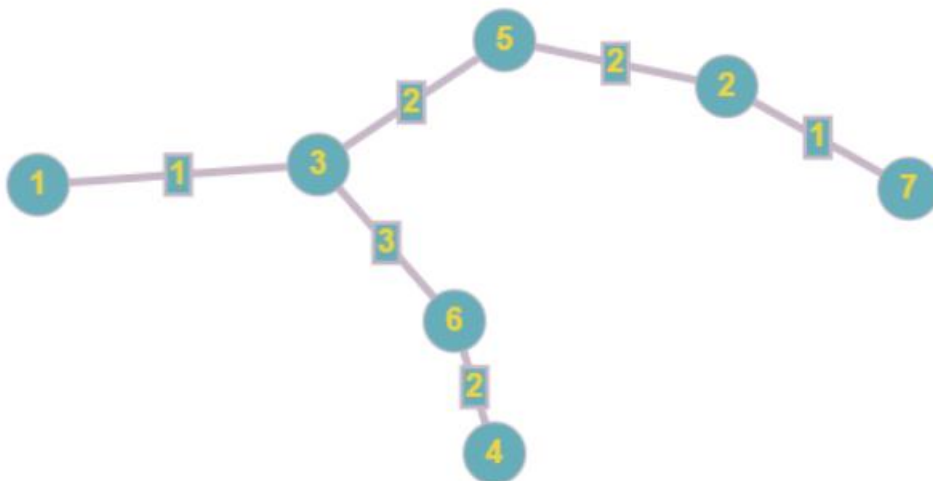
$V=\{1,3,5,2,7\}$   $E=\{(1,3),(3,5),(5,2),(2,7)\}$



$V=\{1,3,5,2,7,6\}$   $E=\{(1,3),(3,5),(5,2),(2,7),(3,6)\}$



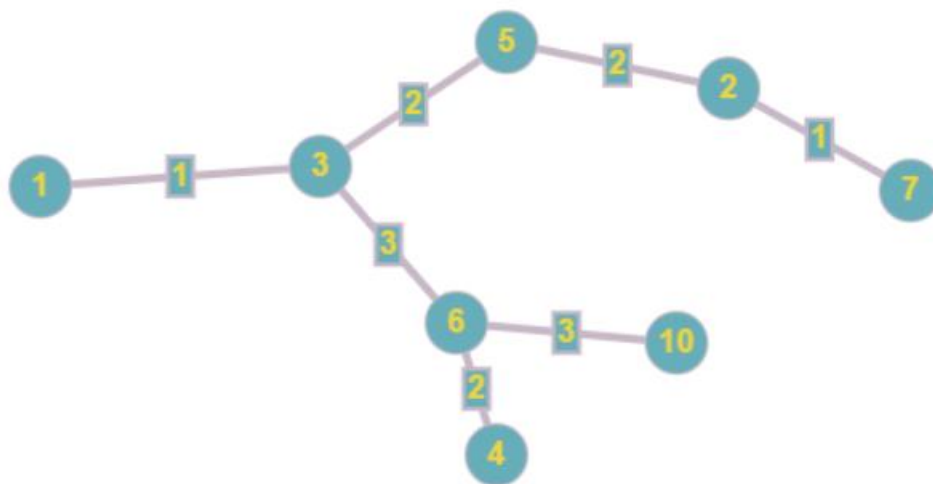
$V=\{1,3,5,2,7,6,4\}$   $E=\{(1,3),(3,5),(5,2),(2,7),(3,6),(6,4)\}$



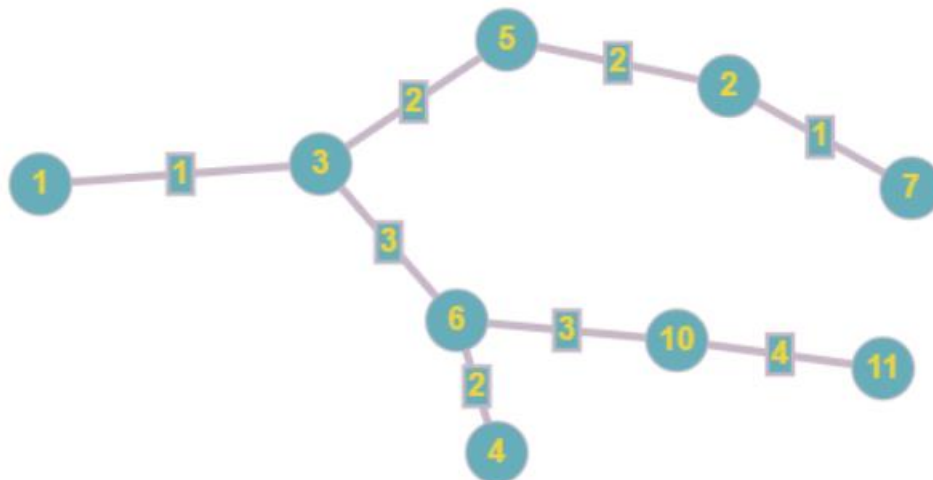

---

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



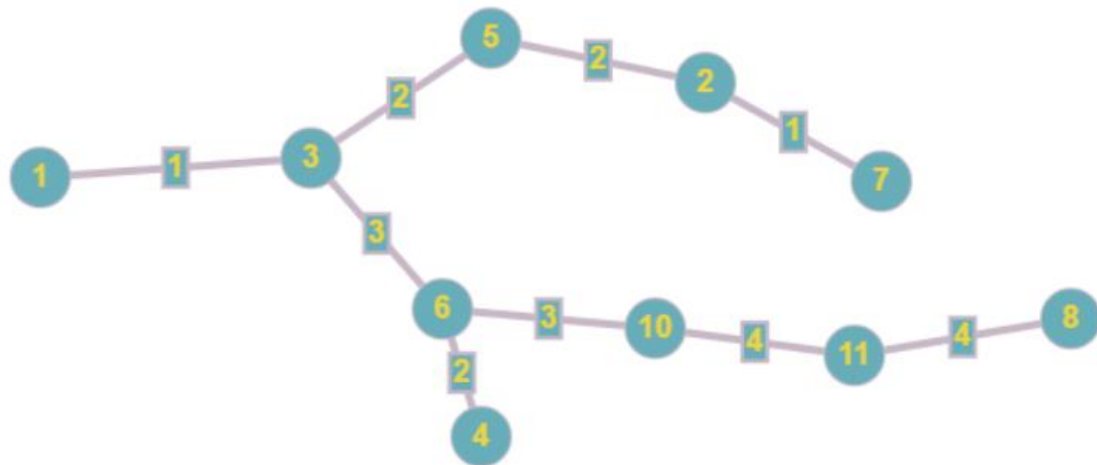


$V = \{1, 3, 5, 2, 7, 6, 4, 10\}$   $E = \{(1, 3), (3, 5), (5, 2), (2, 7), (3, 6), (6, 4), (6, 10)\}$



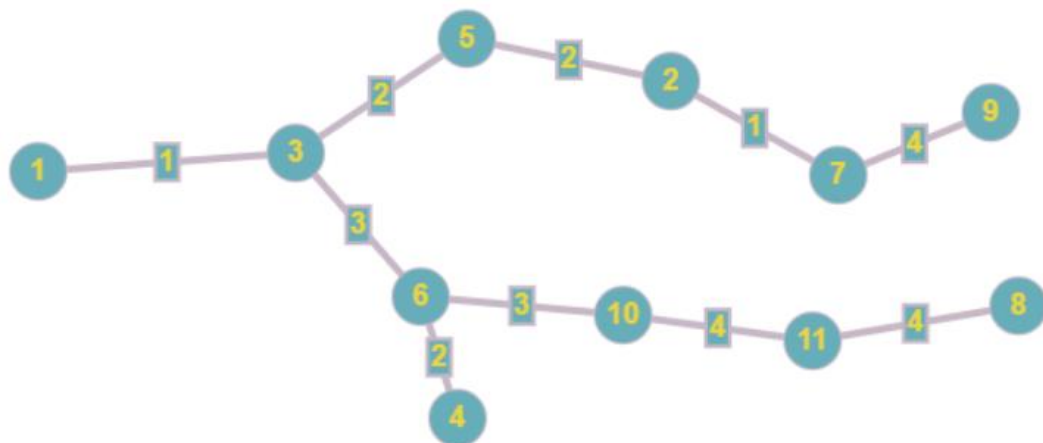
$V = \{1, 3, 5, 2, 7, 6, 4, 10, 11, 8\}$

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



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

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



Weight=26.

```

1  #include <iostream>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include "Source.h"
5
6  using namespace std;
7
8  int main(void)
9  {
10     int versh, cnt = 0, min_ = 0, n, m;
11     bool c = false;
12
13     cout << "Vershini : "; cin >> versh; cout << "\n";
14
15     int** graph = new int* [versh];
16     for (int i = 0; i < versh; ++i)
17         graph[i] = new int[versh];
18
19     int** rebr = new int* [versh - 1];
20     for (int i = 0; i < versh - 1; ++i)
21         rebr[i] = new int[2];
22
23     for (int i = 0; i < versh; ++i)
24         for (int j = 0; j < versh; ++j)
25             cin >> graph[i][j];
26
27     int* tops = new int[versh];
28     tops[cnt] = 1;
29     ++cnt;
30
31     for (int i = 0; cnt < versh; ++i) {
32         for (int j = 0; j < cnt; ++j) {
33             for (int x = 0; x < versh; ++x) {
34                 for (int y = 0; y < cnt; ++y)
35                     if (tops[y] == x + 1)
36                         c = true;
37                 if (c == true)

```

```

38                     if (tops[y] == x + 1)
39                         c = true;
40                     continue;
41                 }
42             }
43             if (min_ == 0 && graph[tops[j] - 1][x] > 0)
44             {
45                 min_ = graph[tops[j] - 1][x];
46                 n = rebr[cnt - 1][0] = tops[j];
47                 m = rebr[cnt - 1][1] = x + 1;
48                 continue;
49             }
50
51             if (graph[tops[j] - 1][x] > 0 && graph[tops[j] - 1][x] < min_)
52             {
53                 min_ = graph[tops[j] - 1][x];
54                 n = rebr[cnt - 1][0] = tops[j];
55                 m = rebr[cnt - 1][1] = x + 1;
56             }
57         }
58     }
59
60     graph[n - 1][m - 1] = 0;
61     graph[m - 1][n - 1] = 0;
62     tops[cnt] = m;
63     ++cnt;
64     min_ = 0;
65 }
66
67 cout << endl << "Rebra: ";
68 for (int i = 0; i < versh - 1; ++i)
69     cout << "(" << rebr[i][0] << ", " << rebr[i][1] << ") ";
70 }

```

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

## Завдання № 6

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

8)

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

Почнемо з 1-ої вершини:

1

Найближча до 1-ої вершина - 7

1->7

Довжина шляху: 2

найближча до 7-ої вершини - 2

1->7->2

Довжина шляху: 2+1

найближча до 2-ої вершини - 4

1->7->2->4

Довжина шляху: 2+1+1

найближча до 4-ої вершини - 5

1->7->2->4->5

Довжина шляху: 2+1+1+3

найближча до 5-ої вершини - 3

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

Довжина шляху: 2+1+1+3+1

найближча до 3-ої вершини - 8

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

Довжина шляху: 2+1+1+3+1+5

найближча до 8-ої вершини - 6

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

Довжина шляху: 2+1+1+3+1+5+7

Всі вершини пройдені, повертаємось у початкову

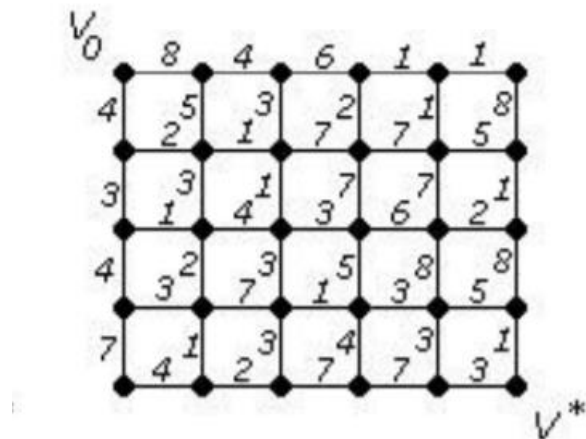
1->7->2->4->5->3->8->6->1

Довжина шляху:  $2+1+1+3+1+5+7+6=26$ .

### Завдання № 7

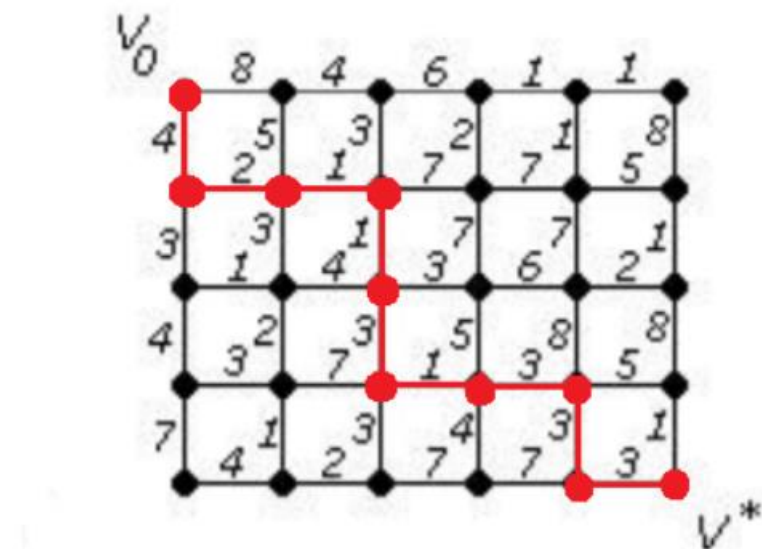
За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин  $V_0$  і  $V^*$ .

8)



Найкоротший шлях:

8)



```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #define SIZE 30
4  #define _CRT_SECURE_NO_WARNINGS
5  int main()
6  {
7      int a[SIZE][SIZE];
8      int d[SIZE];
9      int v[SIZE];
10     int temp, minindex, min;
11     int begin_index = 0;
12     system("chcp 1251");
13     system("cls");
14     printf("Enter Matrix:\n");
15     for (int i = 0; i < SIZE; i++)
16     {
17         for (int j = 0; j < SIZE; j++) {
18             scanf_s("%d", &temp);
19             a[i][j] = temp;
20         }
21     }
22
23     for (int i = 0; i < SIZE; i++)
24     {
25         d[i] = 10000;
26         v[i] = 1;
27     }
28     d[begin_index] = 0;
29     do {
30         minindex = 10000;
31         min = 10000;
32         for (int i = 0; i < SIZE; i++)
33         {
34             if ((v[i] == 1) && (d[i] < min))
35             {
36                 min = d[i];
37                 minindex = i;
38             }
39         }
40         if (minindex != 10000)
41         {

```

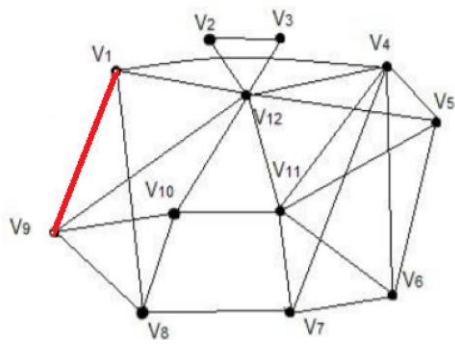
```

42             for (int i = 0; i < SIZE; i++)
43             {
44                 if (a[minindex][i] > 0)
45                 {
46                     temp = min + a[minindex][i];
47                     if (temp < d[i])
48                     {
49                         d[i] = temp;
50                     }
51                 }
52             }
53             v[minindex] = 0;
54         } while (minindex < 10000);
55     printf("\nShortest distants: \n");
56     for (int i = 0; i < SIZE; i++)
57         printf("%3d", d[i]);
58     int ver[SIZE];
59     int end = 29;
60     ver[0] = end + 1;
61     int k = 1;
62     int weight = d[end];
63
64     while (end != begin_index)
65     {
66         for (int i = 0; i < SIZE; i++)
67         {
68             if (a[end][i] != 0)
69             {
70                 int temp = weight - a[end][i];
71                 if (temp == d[i])
72                 {
73                     weight = temp;
74                     end = i;
75                     ver[k] = i + 1;
76                     k++;
77                 }
78             }
79         }
80     }
81     printf("\noutput of the shortest way:");

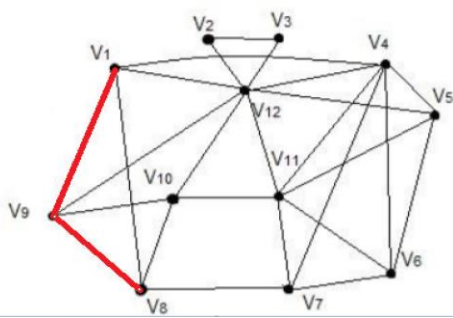
```



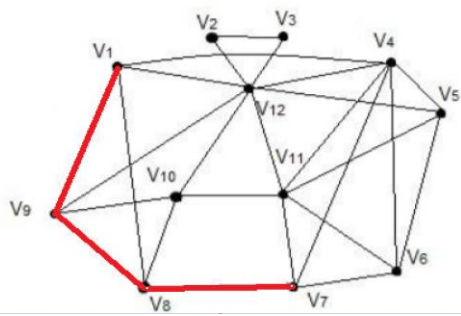
8)



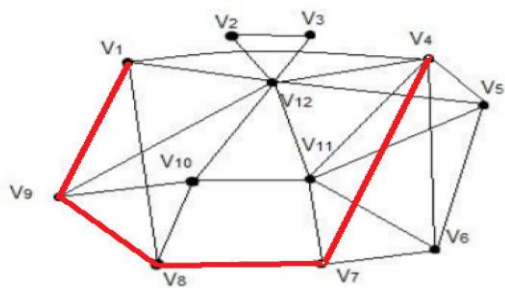
2



3

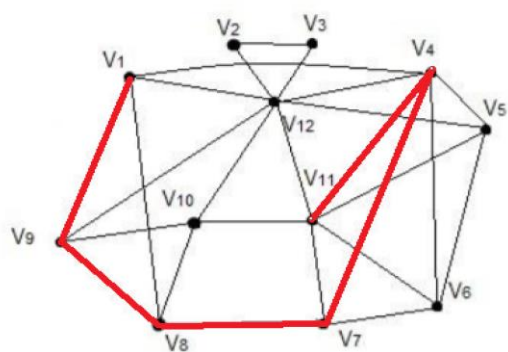


4

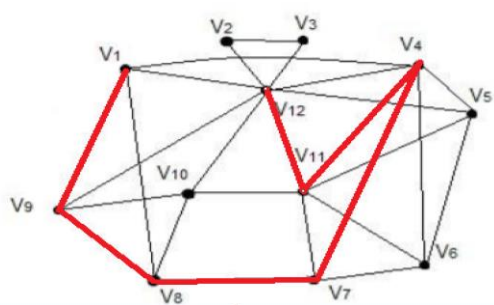


5

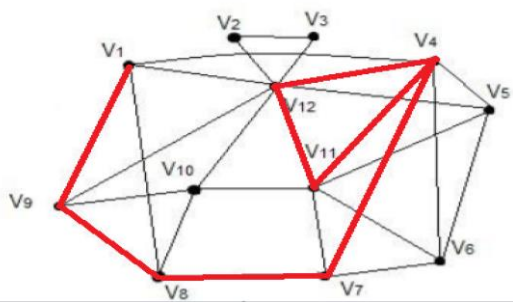




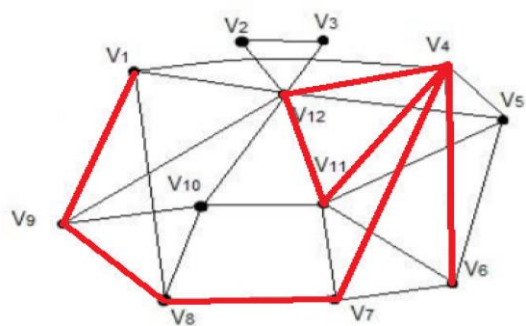
6



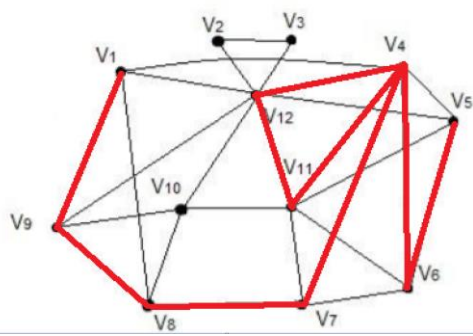
7



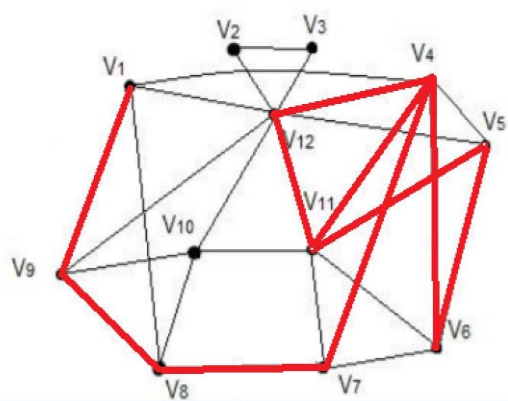
8



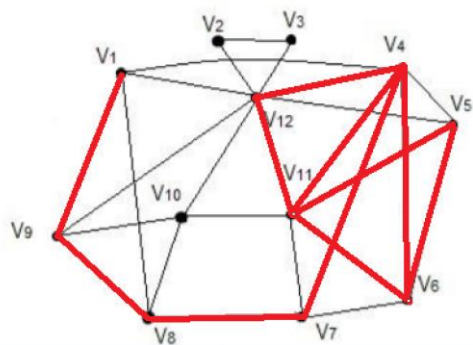
9



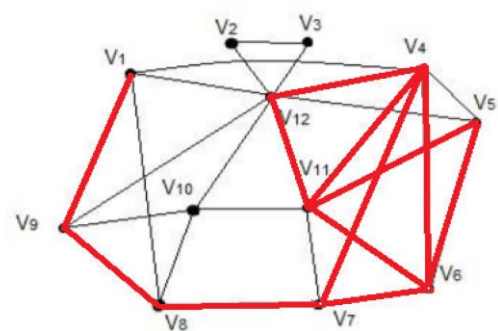
10



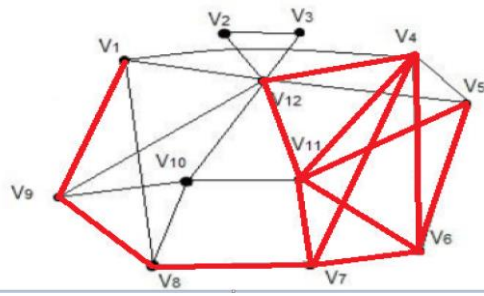
11



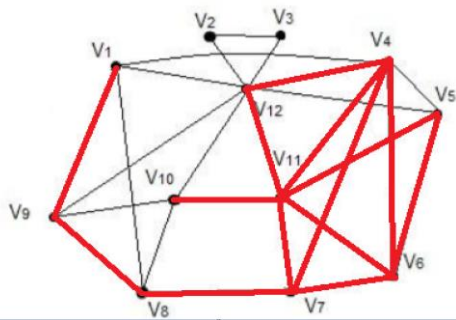
12



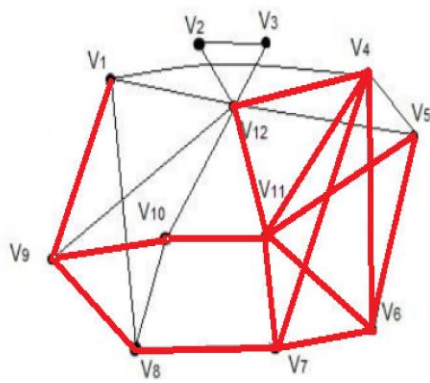
13



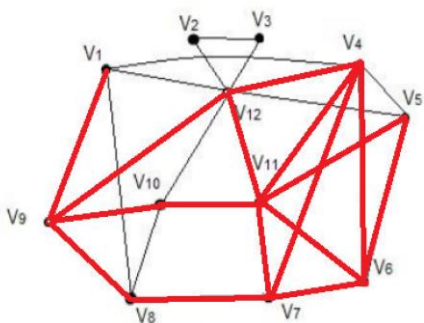
14



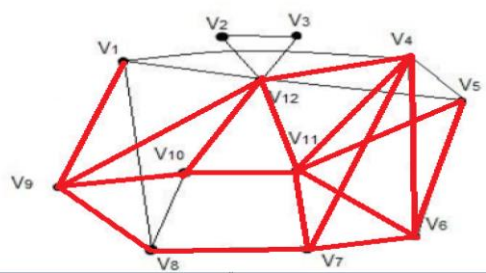
15



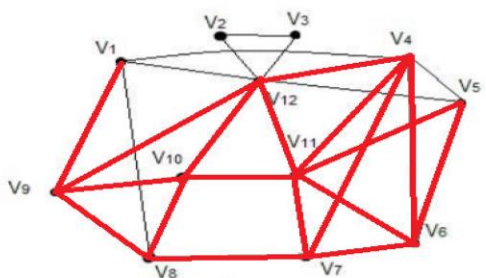
16



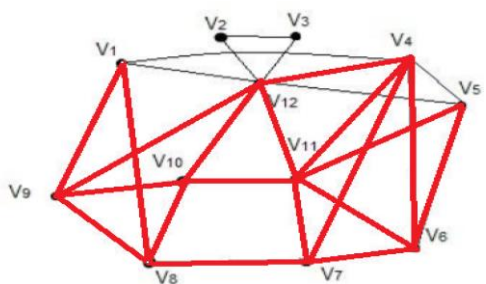
17



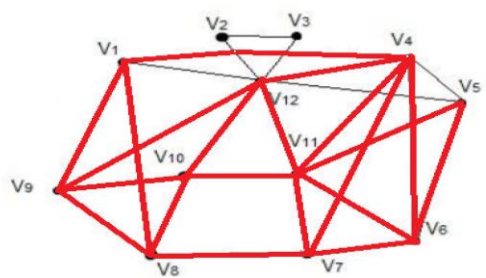
18



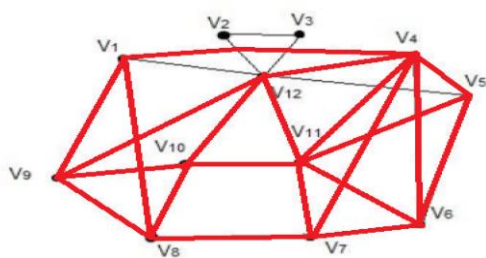
19



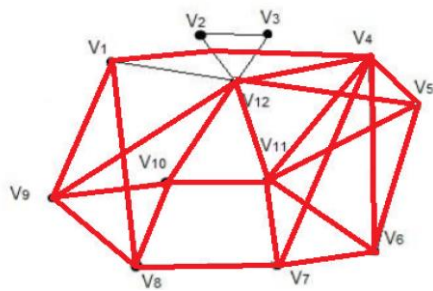
20



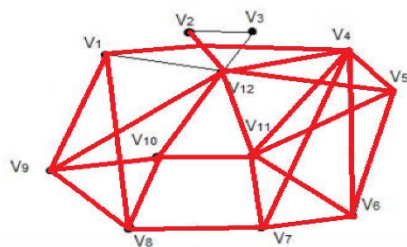
21



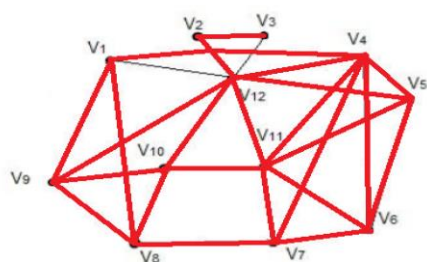
22



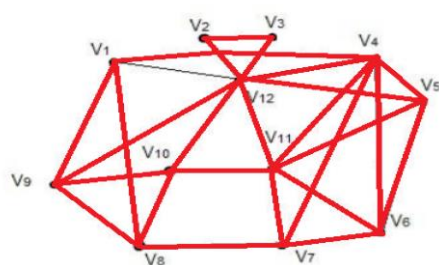
23



24



25



26

```

1  #include<iostream>
2  #include<vector>
3  #define v 12
4  using namespace std;
5  int matrix[v][v] = {
6      {0,0,0,0,0,0,0,1,0,1,0,0},
7      {0,0,1,0,0,0,0,1,0,1,0,0},
8      {0,1,0,1,0,1,0,1,0,1,0,1},
9      {0,0,1,0,0,0,0,1,0,0,0,0},
10     {0,0,0,0,0,1,0,0,0,0,0,1},
11     {0,0,1,0,1,0,0,0,0,0,0,0},
12     {0,1,0,1,0,0,0,1,0,0,0,1},
13     {1,0,1,0,0,0,0,1,0,0,0,1,0},
14     {0,1,0,0,0,0,0,0,0,1,0,0},
15     {1,0,1,0,0,0,0,0,0,1,0,1},
16     {0,1,0,0,0,0,0,1,0,1,0,1},
17     {0,0,1,0,1,0,1,0,0,0,1,0}
18 };
19 int temp[v][v];
20 int findstartvr() {
21     for (int i = 0; i < v; i++) {
22         int stp = 0;
23         for (int j = 0; j < v; j++) {
24             if (temp[i][j])
25                 stp++;
26         }
27         if (stp % 2 != 0)
28             return i;
29     }
30     return 0;
31 }
32 bool most(int u, int vr) {
33     int stp = 0;
34     for (int i = 0; i < v; i++)
35         if (temp[vr][i])

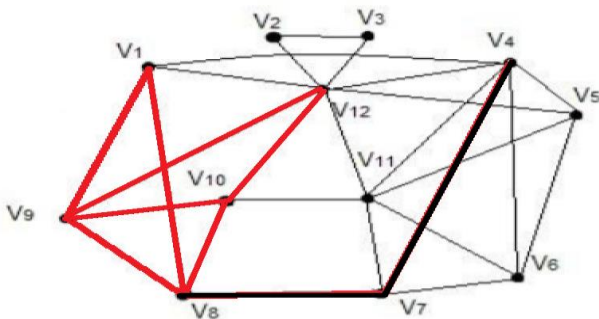
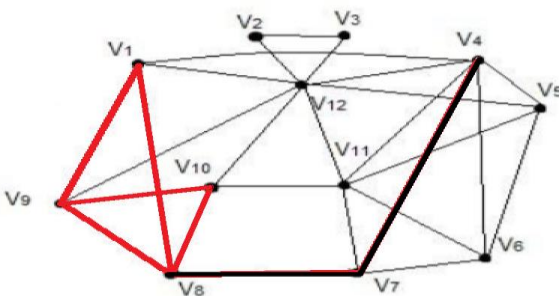
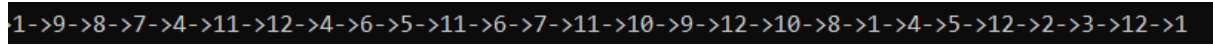
```

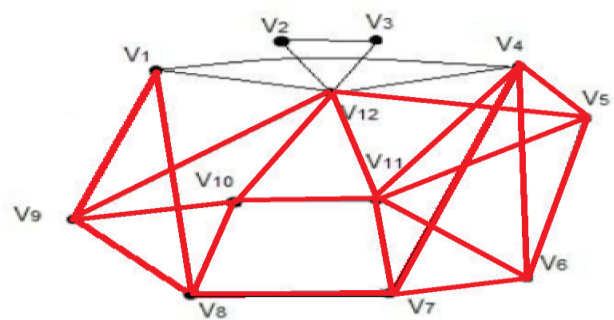
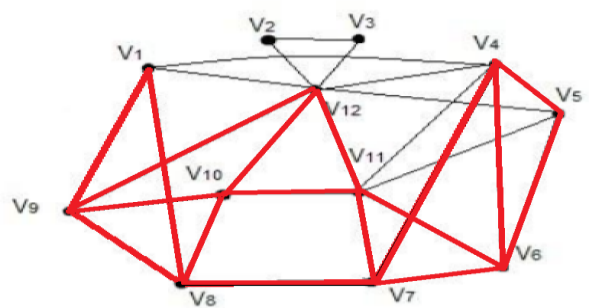
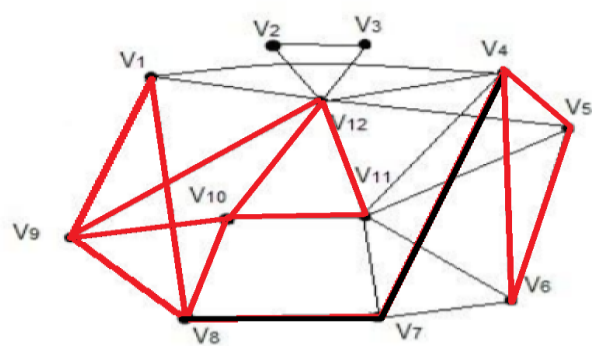
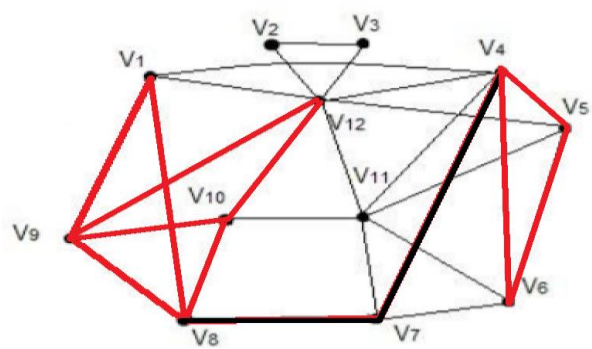
```

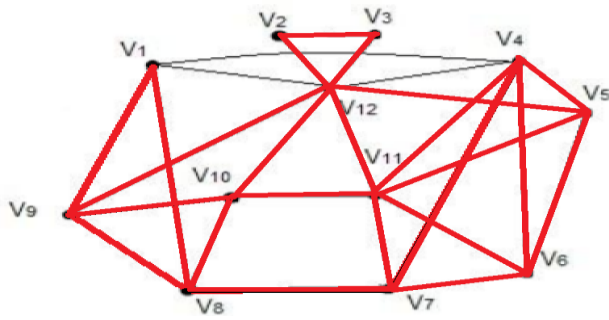
34     for (int i = 0; i < v; i++)
35         if (temp[vr][i])
36             stp++;
37     if (stp > 1) {
38         return false;
39     }
40     return true;
41 }
42 int edgecount() {
43     int counter = 0;
44     for (int i = 0; i < v; i++)
45         for (int j = i; j < v; j++)
46             if (temp[i][j])
47                 counter++;
48     return counter;
49 }
50 void fleri(int start) {
51     static int edge = edgecount();
52     for (int i = 0; i < v; i++) {
53         if (temp[start][i]) {
54             if (edge <= 1 || !most(start, i)) {
55                 cout << start+1 << "-" << i+1 << " ";
56                 temp[start][i] = temp[i][start] = 0;
57                 edge--;
58                 fleri(i);
59             }
60         }
61     }
62 }
63 int main() {
64     for (int i = 0; i < v; i++)
65         for (int j = 0; j < v; j++)
66             temp[i][j] = matrix[i][j];
67     cout << "Euler Path Or Circuit: ";
68     fleri(findstartvr());

```

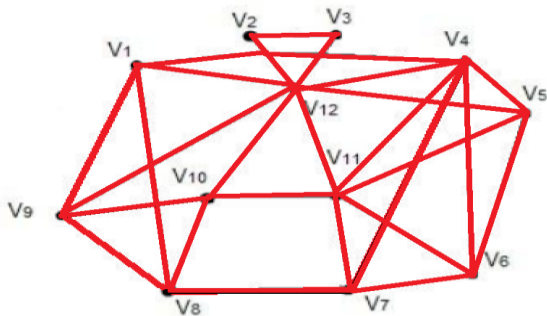








Знайдено всі елементарні цикли, ось результат їх поєднання:



### Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

$$8. (y \cdot x \cdot \bar{y}) \vee x \vee (y \cdot x \cdot \bar{x})$$

Скорочена ДНФ (англ. Reduced disjunctive normal form) - форма запису функції, що володіє наступними властивостями:

- 1) будь-які два доданки відрізняються як мінімум в двох позиціях,
- 2) жоден з Кон'юнктів не міститься в іншому.

Отже,

- 1.  $y * (-y) = 0$ ;
- 2.  $-(x * (-x)) = 1$ ;
- 3.  $y * 1 = y$ ;
- 4.  $x * 0 = 0$ ;
- 5.  $0 \vee x \vee y = x \vee y$ ;

Відповідь :  $x \vee y$ ;