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:

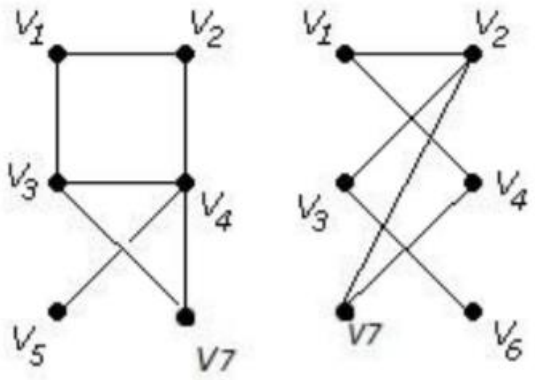
Мельникова Н.І.

Львів – 2019 р.

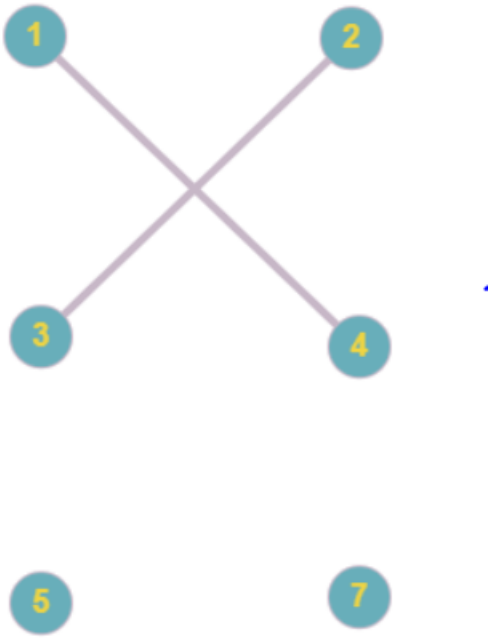
Варіант 8
ІНДИВІДУАЛЬНІ ЗАВДАННЯ
Завдання № 1

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 А - що скадається з 3-х вершин в G1 6) добуток графів.

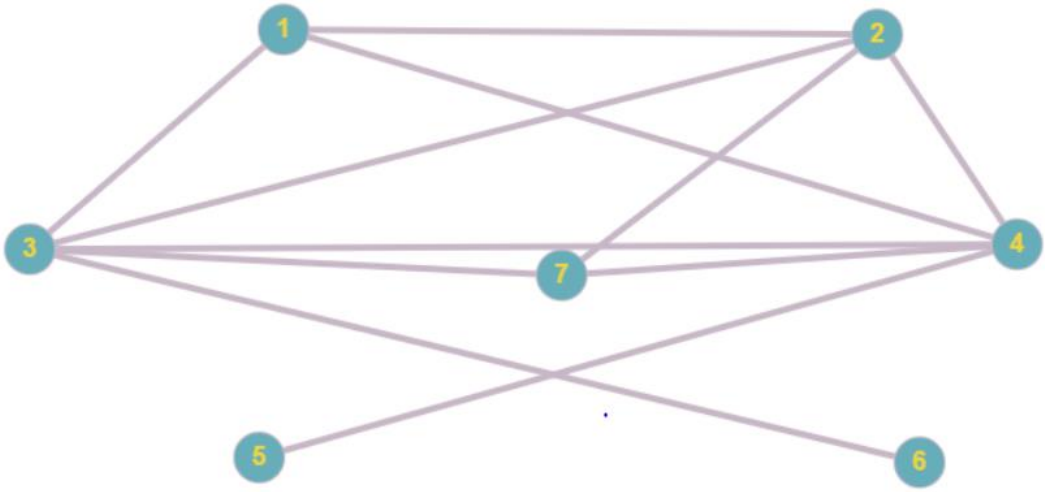
8)



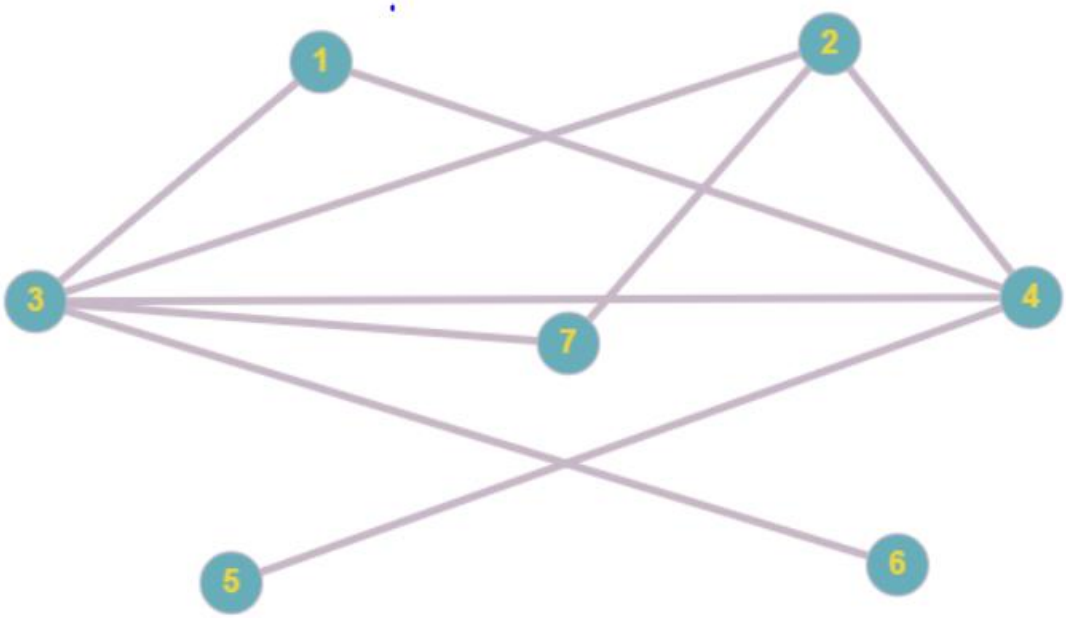
1)Доповнення
G1\G2:



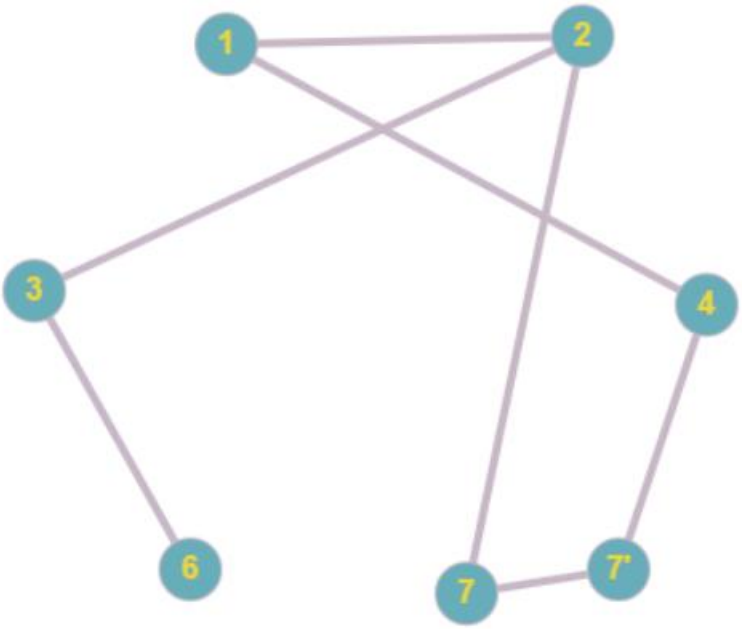
2)об’єднання графів:



3)Кільцева сума:

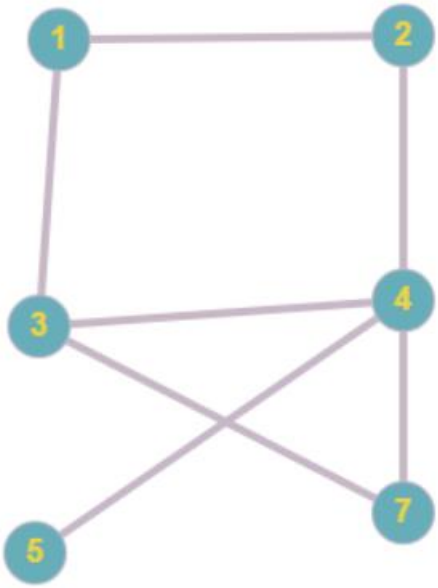


4)Розщепити вершину у другому графі:
Розщепимо вершину 7

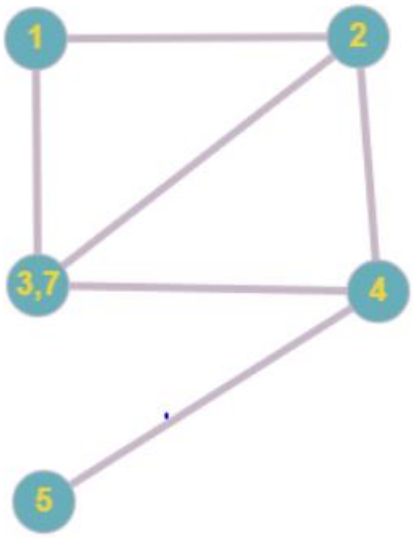


5)Виділити підграф A,що складається з V={3,4,7} в G1 і знайти стягнення A в G1

G1



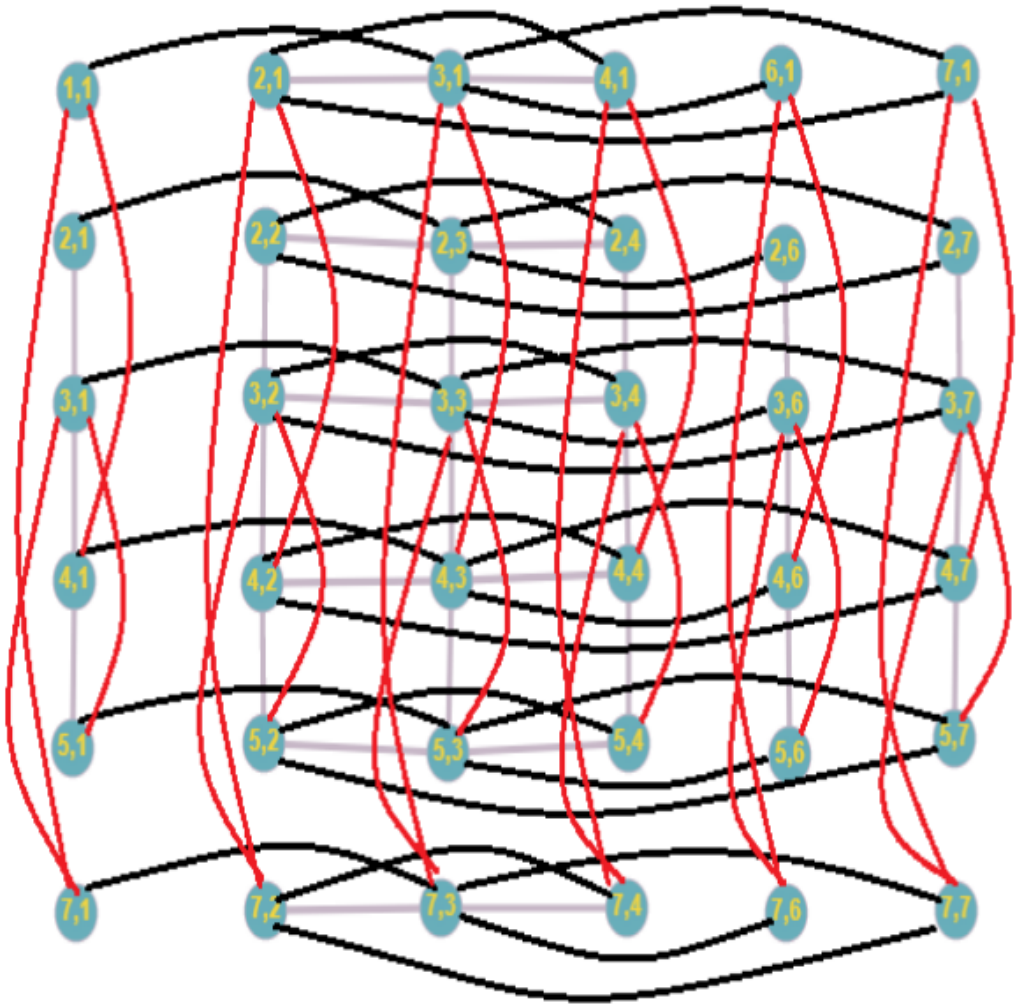
Стягуємо 7 в 3



Стягуємо 3,7 в 4

Стягуємо 3,7,4 в 2

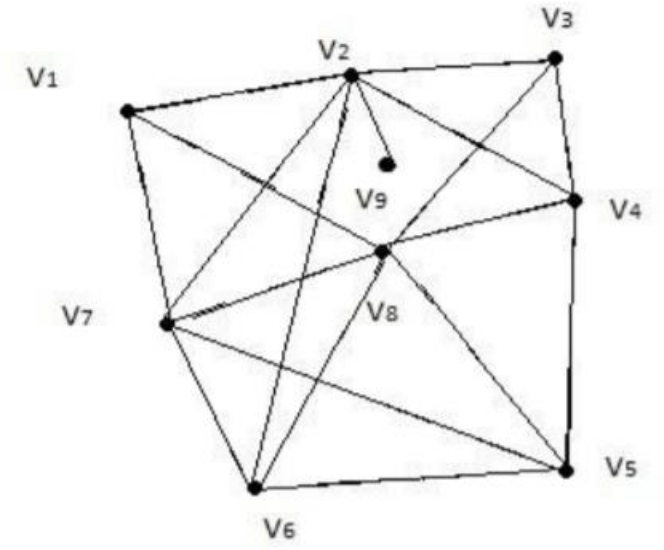
6)Добуток графів



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	-	-

```cpp
#include<iostream>
#include<queue>
#define v 8
using namespace std;
class n {
public:
    int val;
    int st;
};
int matrix[v][v] = {
   {0, 1, 1, 1, 0, 1, 1, 0},
   {1, 0, 1, 1, 0, 0, 1, 1},
   {1, 1, 0, 1, 0, 1, 1, 0 },
   {1, 1, 1, 0, 0, 0, 1 ,0},
   {0, 0, 0, 0, 0, 1, 7, 0},
   {1, 0, 0, 0, 1, 0, 1, 0},
    {1, 1, 1, 1 ,1, 1,0,0},
    {0,1,0,0,0,0,0,0}
};
void bfs(n* verh, n s) {
    n u;
    int i, j;
    queue<n> que;
    for (i = 0; i < v; i++) {
        verh[i].st = 0;
    }
    verh[s.val].st = 1;
    que.push(s);
    while (!que.empty()) {
        u = que.front();
        que.pop();
        cout << u.val+1 << " ";
        for (i = 0; i < v; i++) {
            if (matrix[i][u.val]) {
```

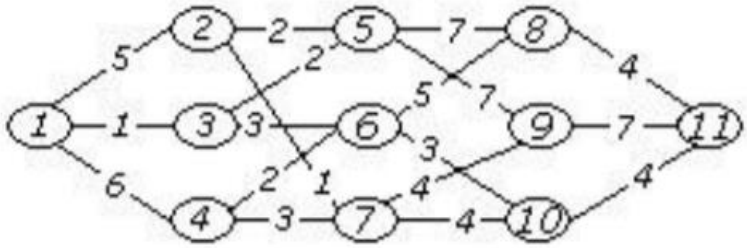
```cpp
        verh[i].st = 0;
    }
    verh[s.val].st = 1;
    que.push(s);
    while (!que.empty()) {
        u = que.front();
        que.pop();
        cout << u.val+1 << " ";
        for (i = 0; i < v; i++) {
            if (matrix[i][u.val]) {

                if (verh[i].st == 0) {
                    verh[i].st = 1;
                    que.push(verh[i]);
                }
            }
        }
        u.st = 2;
    }
}
int main() {
    n verh[v];
    n start;
    int s;
    for (int i = 0; i < v; i++) {
        verh[i].val = i;
    }
    s = 65;
    start.val = s - 65;
    cout << "bfs: ";
    bfs(verh, start);
    cout << endl;
}
```

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

Завдання № 5

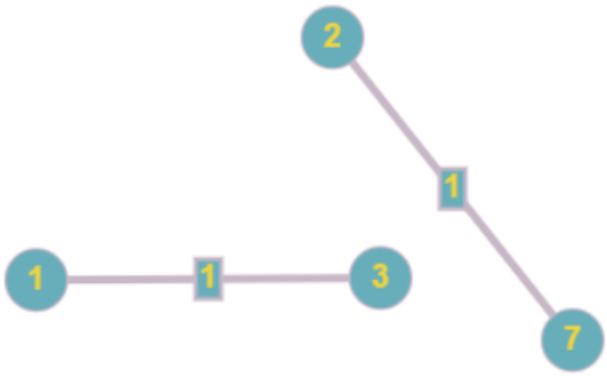
Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

8)

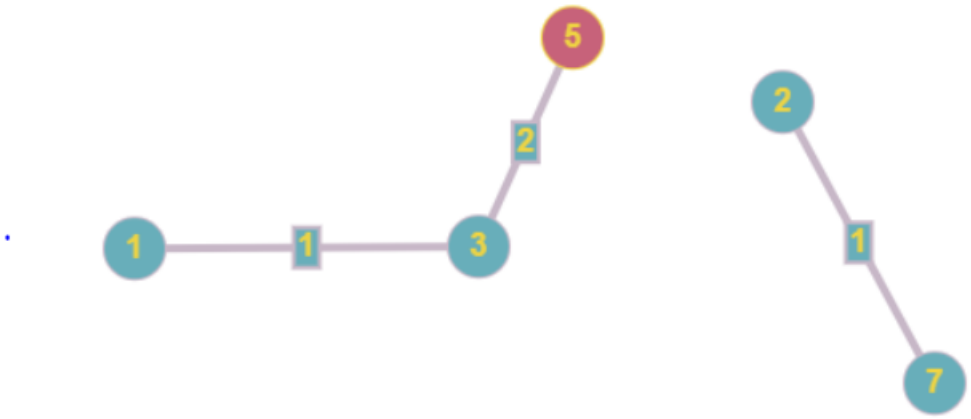


Метод Краскала V={1,3} E={(1,3)}

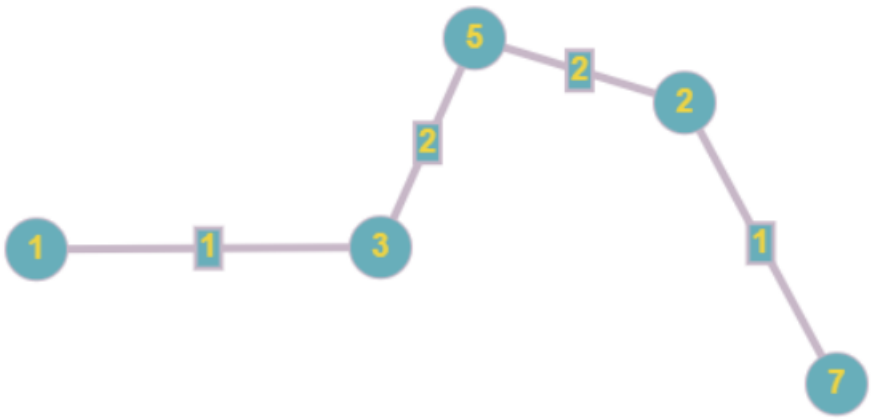
V={1,2,3,7,}
E={(1,3),(2,7)}



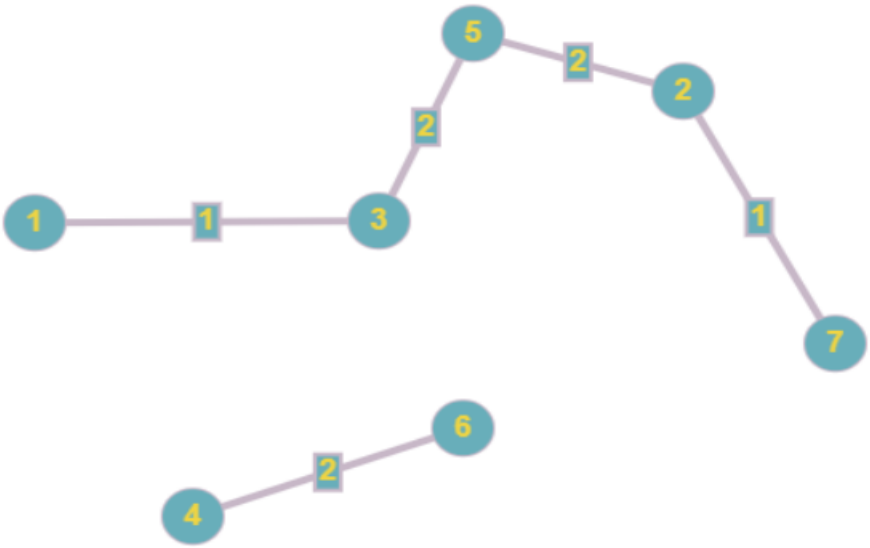
V={1,2,5,7,9}
E={(1,3),(2,7),(3,5)}



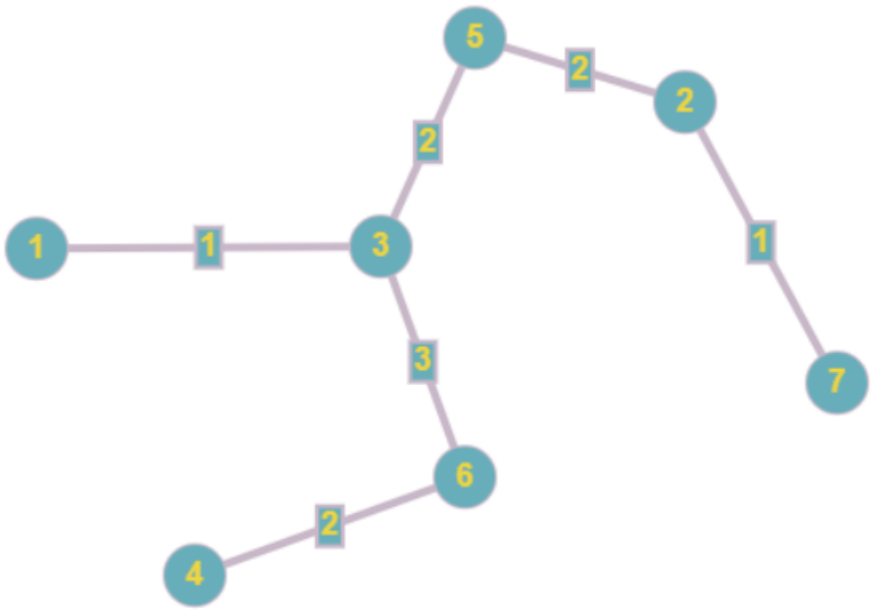
V={1,3,5,2,7} E={(1,3),(3,5),(5,2),(2,7)}



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

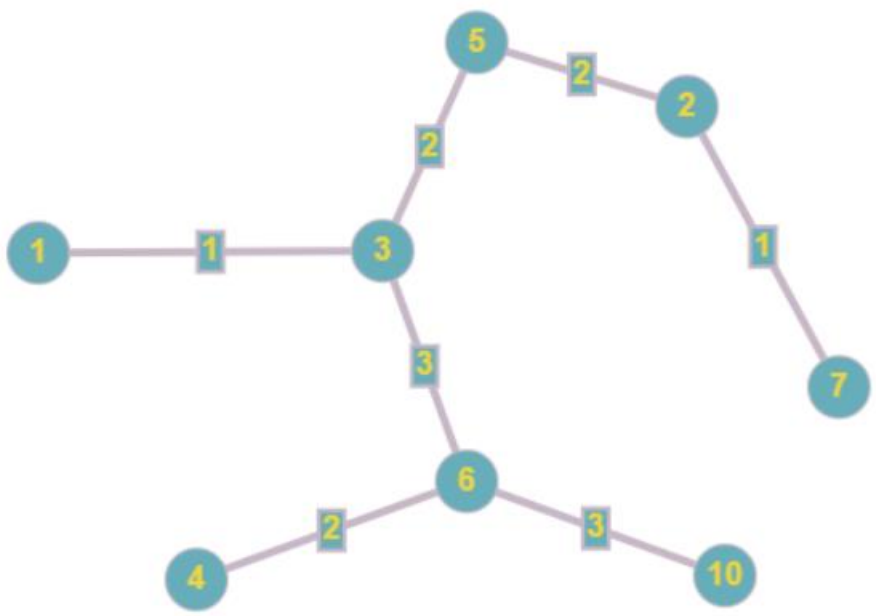


V={1,2,7,9,5,4,6} E={(1,3),(3,5),(5,2),(2,7),(4,6),(3,6)}

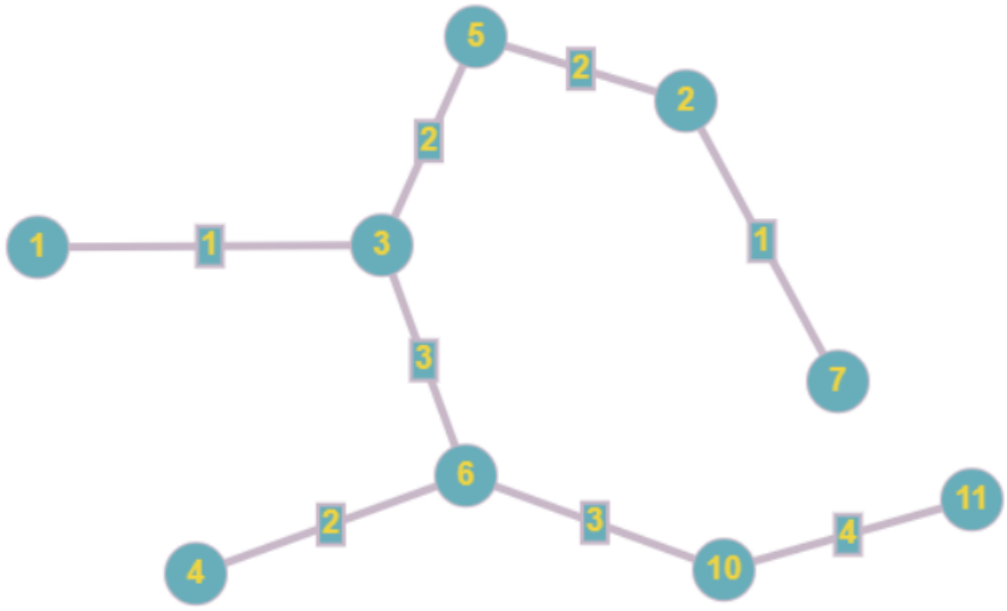


{(4,7)} - цикл

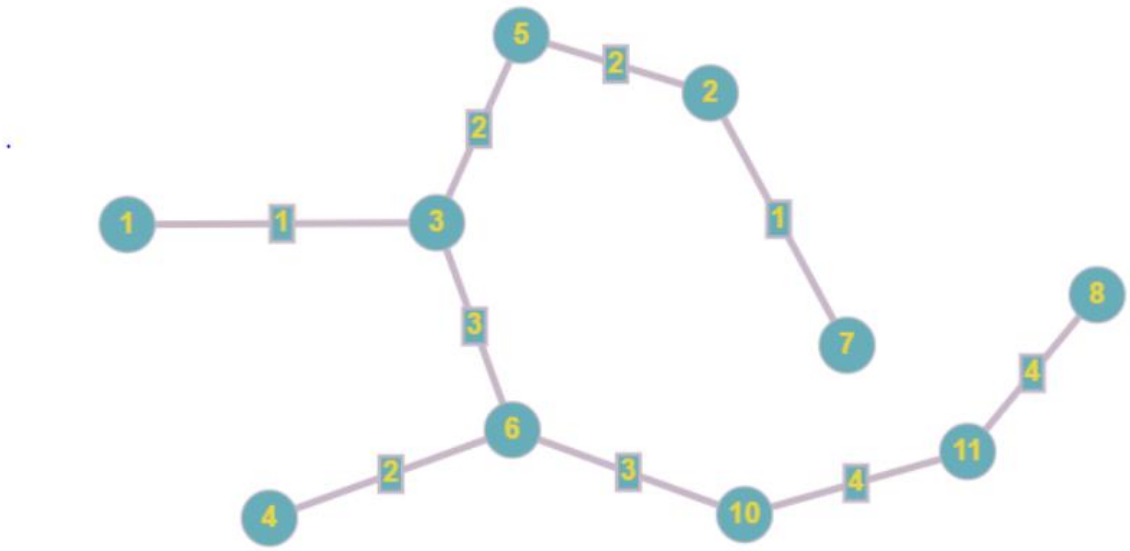
V={1,2,7,9,5,4,6,10} E={(1,3),(3,5),(5,2),(2,7),(4,6),(3,6),(6,10)}



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)}

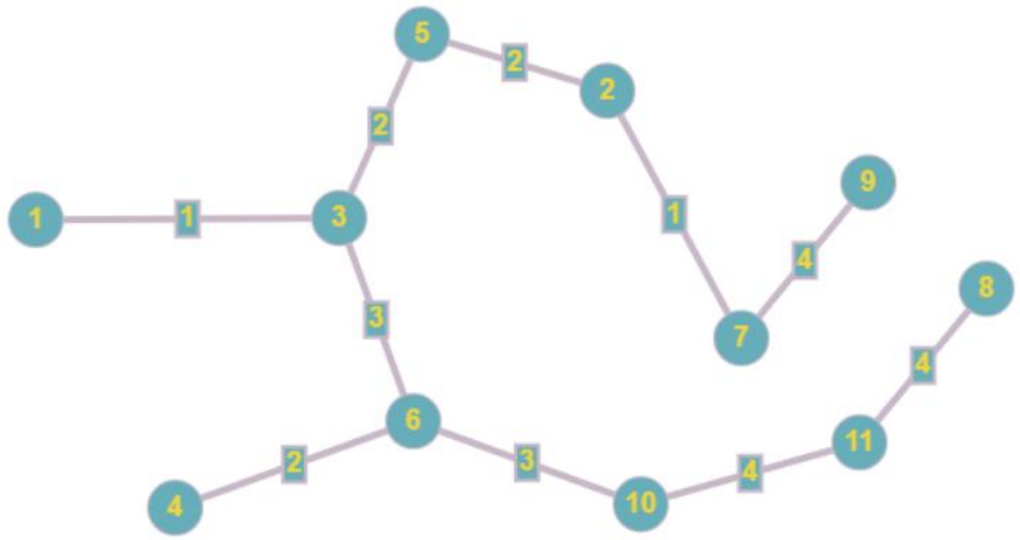


V={1,2,7,9,5,4,6,10,11,8}
E={(1,3),(3,5),(5,2),(2,7),(4,6),(3,6),(6,10),(10,11),(11,8)}



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

        union1(a, b);
        printf("Edge %d:(%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},
```

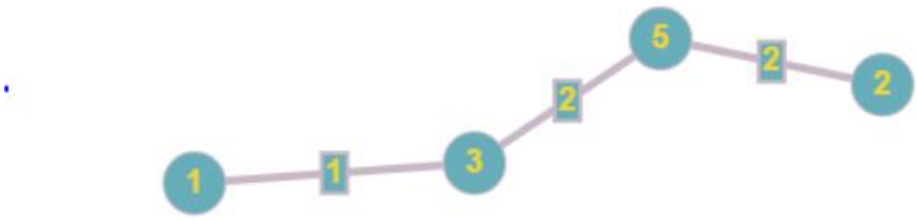
```
V={1,2,7,9,5,4,6,10,11,8,3}
E={(1,2),(7,9),(2,7),(5,9),(4,6),(1,4),(6,10),(10,11),(5,8),(6,3)}
```

Метод Прима:

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

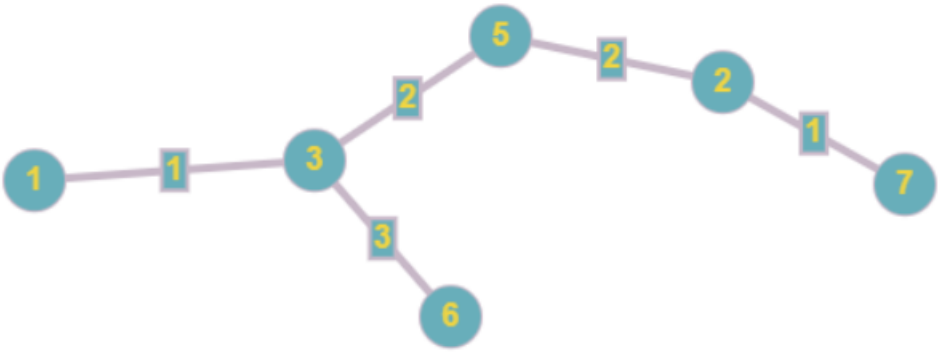
V={1,3,5} E={(1,3),(3,5)}

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

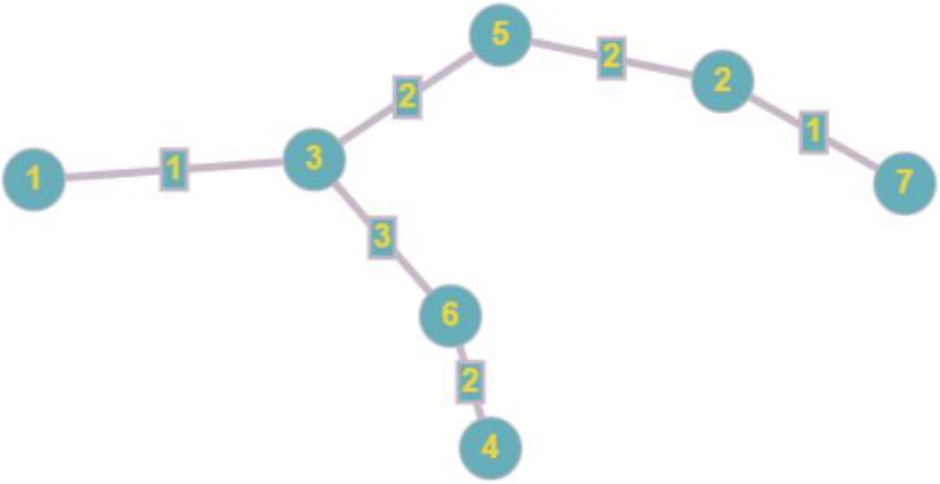


V={1,3,5,2,7} E={(1,3),(3,5),(5,2),(2,7)}

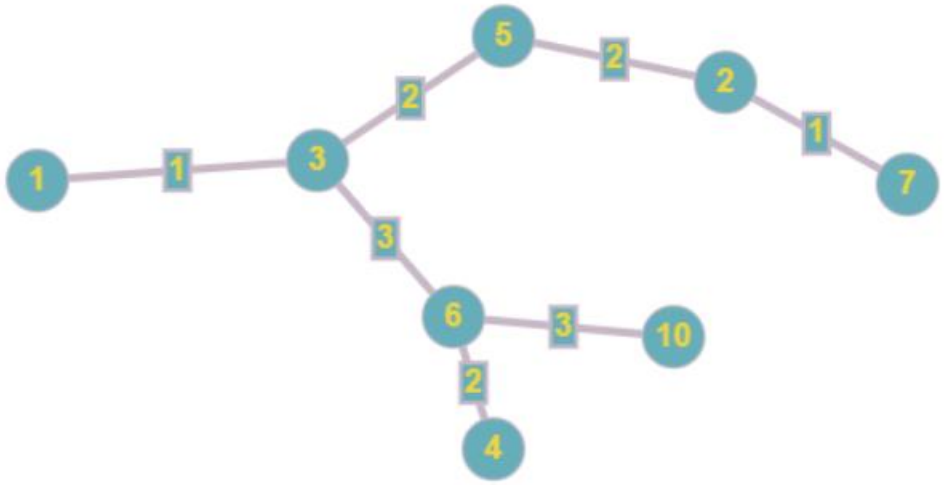
V={1,3,5,2,7,6} E={(1,3),(3,5),(5,2),(2,7),(3,6}



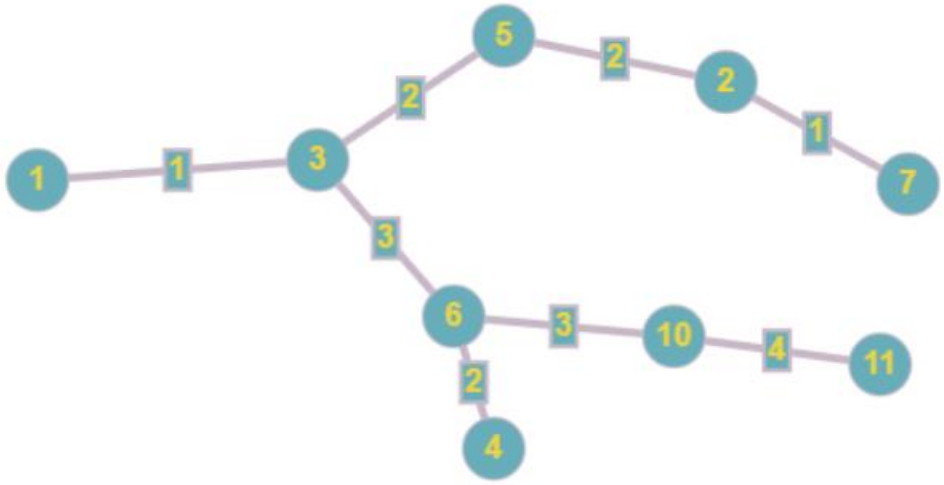
V={1,3,5,2,7,6,4} E={(1,3),(3,5),(5,2),(2,7),(3,6),(6,4)}



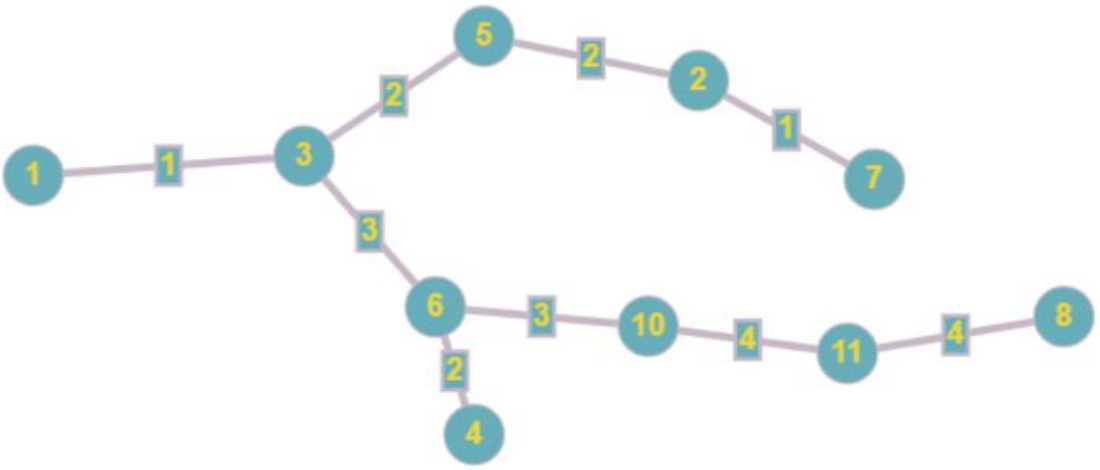
V={1,3,5,2,7,6,4,10} E={(1,3),(3,5),(5,2),(2,7),(3,6),(6,4),(6,10



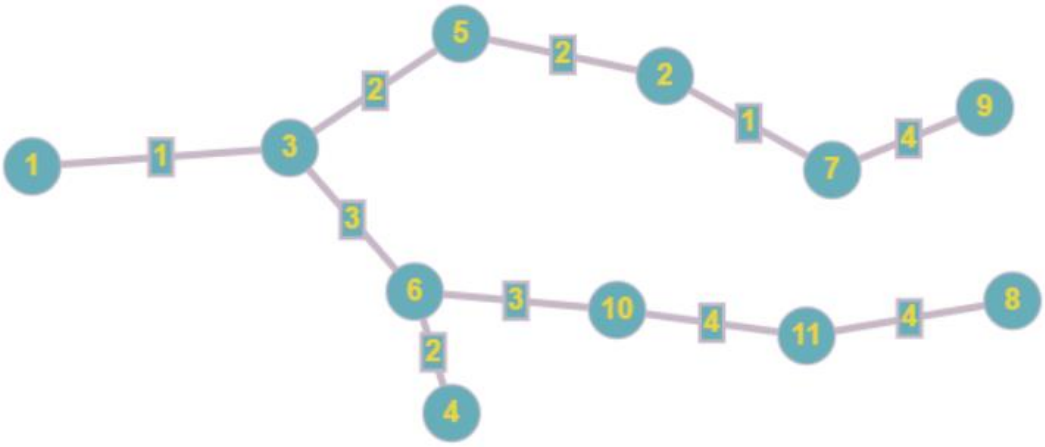
V={1,3,5,2,7,6,4,10} E={(1,3),(3,5),(5,2),(2,7),(3,6),(6,4),(6,10)}



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.

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

int main()
{

    int v, count = 0, min = 0, k, t;
    bool check = false;
    cin >> v;
    int* tops = new int[v];

    int** matrix = new int* [v];
    for (int i = 0; i < v; i++) {
        matrix[i] = new int[v];
    }

    int** rebra = new int* [v - 1];

    for (int i = 0; i < v - 1; i++) {
        rebra[i] = new int[2];
    }

    for (int i = 0; i < v; i++) {
        for (int j = 0; j < v; j++) {
            cin >> matrix[i][j];
        }
    }


    tops[count] = 1;
    count++;

    for (int i = 0; count < v; i++) {
        for (int j = 0; j < count; j++) {
            for (int a = 0; a < v; a++) {
                for (int m = 0; m < count; m++) {

                    if (tops[m] == a + 1) {
                        check = true;
                    }
                }
```

```cpp
                }
            }

            if (check) { check = false; continue; }

            if (min == 0 && matrix[tops[j] - 1][a] > 0) {
                min = matrix[tops[j] - 1][a];
                k = rebra[count - 1][0] = tops[j]; t = rebra[count - 1][1] = a + 1;
                continue;
            }

            if (matrix[tops[j] - 1][a] > 0 && matrix[tops[j] - 1][a] < min) {
                min = matrix[tops[j] - 1][a];

                k = rebra[count - 1][0] = tops[j]; t = rebra[count - 1][1] = a + 1;
            }
        }
    }

    matrix[k - 1][t - 1] = 0; matrix[t - 1][k - 1] = 0;

    tops[count] = t;
    count++;
    min = 0;
}




for (int j = 0; j < v - 1; j++) {
    cout << rebra[j][0] << " " << rebra[j][1] << endl;
}

return 0;

}
```

```
11
0 5 1 6 0 0 0 0 0 0 0
5 0 0 0 2 0 1 0 0 0 0
1 0 0 0 2 3 0 0 0 0 0
6 0 0 0 0 2 3 0 0 0 0
0 2 2 0 0 0 0 7 7 0 0
0 0 3 2 0 0 0 5 0 3 0
0 1 0 3 0 0 0 0 4 4 0
0 0 0 0 7 5 0 0 0 0 4
0 0 0 0 7 0 4 0 0 0 7
0 0 0 0 0 3 4 0 0 0 4
0 0 0 0 0 0 0 4 7 4 0
1 3
3 5
5 2
2 7
3 6
6 4
6 10
7 9
10 11
11 8
```

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

Почнемо з 1-ої вершини:

1

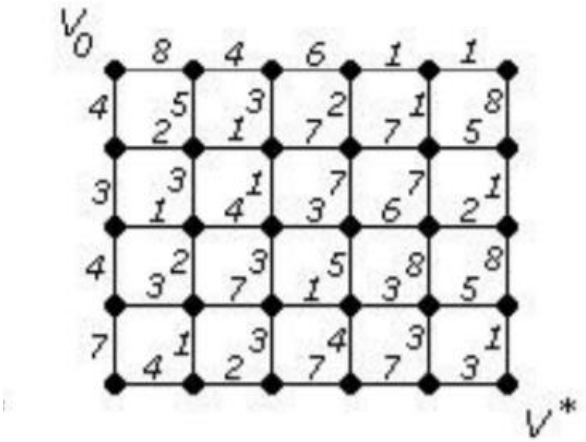
Найближча до 1-ої вершина - 7

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

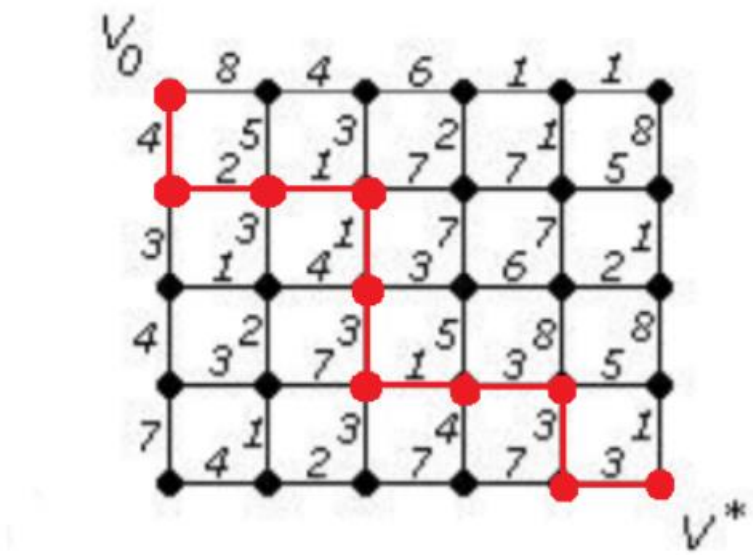
За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V * .

8)



Найкоротший шлях:

8)




```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 30
#define _CRT_SECURE_NO_WARNINGS
int main()
{
    int a[SIZE][SIZE];
    int d[SIZE];
    int v[SIZE];
    int temp, minindex, min;
    int begin_index = 0;
    system("chcp 1251");
    system("cls");
    printf("Enter Matrix:\n");
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++) {
            scanf_s("%d", &temp);
            a[i][j] = temp;
        }
    }

    for (int i = 0; i < SIZE; i++)
    {
        d[i] = 10000;
        v[i] = 1;
    }
    d[begin_index] = 0;
    do {
        minindex = 10000;
        min = 10000;
        for (int i = 0; i < SIZE; i++)
        {
            if ((v[i] == 1) && (d[i] < min))
            {
                min = d[i];
                minindex = i;
            }
        }
        if (minindex != 10000)
        {
```

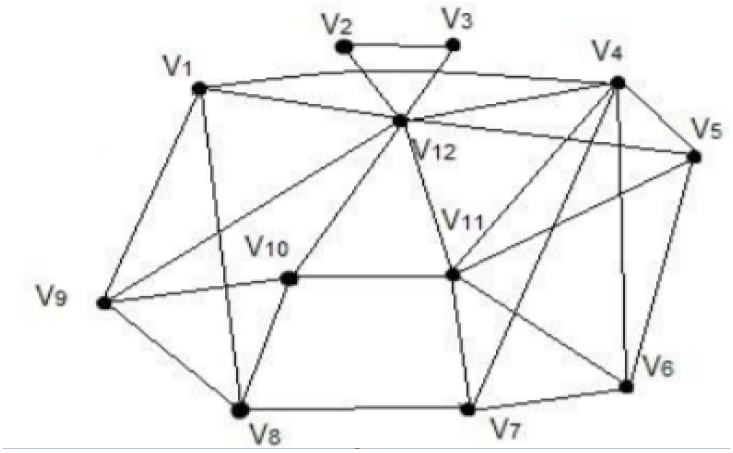
```
    {
      for (int i = 0; i < SIZE; i++)
      {
        if (a[minindex][i] > 0)
        {
          temp = min + a[minindex][i];
          if (temp < d[i])
          {
            d[i] = temp;
          }
        }
      }
      v[minindex] = 0;
    }
  } while (minindex < 10000);
  printf("\nShortest distants: \n");
  for (int i = 0; i < SIZE; i++)
    printf("%3d", d[i]);
  int ver[SIZE];
  int end = 29;
  ver[0] = end + 1;
  int k = 1;
  int weight = d[end];

  while (end != begin_index)
  {
    for (int i = 0; i < SIZE; i++)
      if (a[end][i] != 0)
      {
        int temp = weight - a[end][i];
        if (temp == d[i])
        {
          weight = temp;
          end = i;
          ver[k] = i + 1;
          k++;
        }
      }
  }

  printf("\nOutput of the shortest way:");
```

```
  printf("\nOutput of the shortest way:");
  for (int i = k - 1; i >= 0; i--)
    printf("%3d ", ver[i]);
  return 0;
}

/*
0 7 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7 0 5 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 5 0 3 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 3 0 8 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 8 0 5 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 5 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 7 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 0 0 0 0 7 0 3 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 3 0 0 0 0 3 0 1 0 0 0 0 7 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 4 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 4 0 0 0 0 2 0 2 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0 0 0 0 3 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 3 0 8 0 0 0 0 5 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 7 0 0 0 0 8 0 8 0 0 0 0 6 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 4 0 0 0 0 8 0 7 0 0 0 0 6 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 7 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 7 0 0 0 0 0 0 7 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 4 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 1 0 3 0 0 0 0 2 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 3 0 1 0 0 0 0 3 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 1 0 1 0 0 0 0 8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 1 0 5 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 5 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 0 0 0 0 3 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 3 0 3 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 3 0 4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 4 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 3
0 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
*/
```

Завдання № 8

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

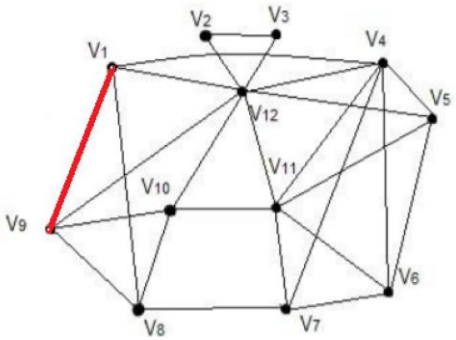
8)



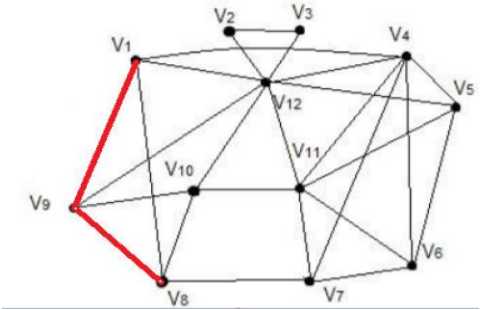
Метод флері:

1

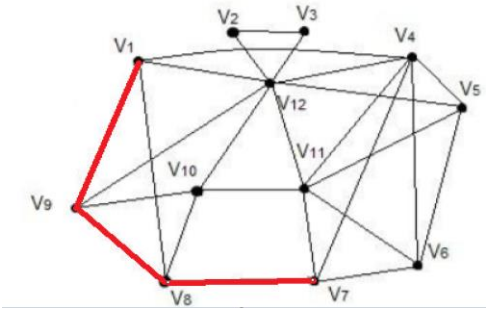
8)



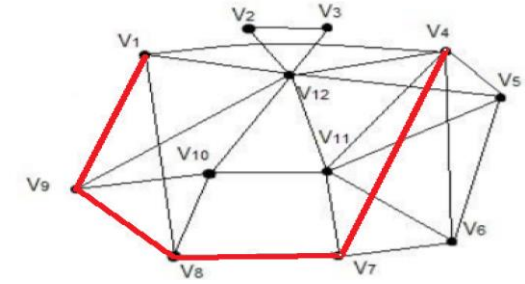
2



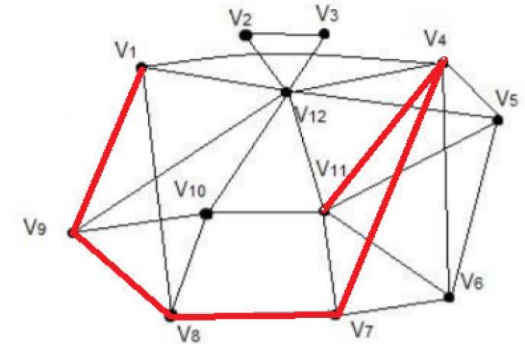
3


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

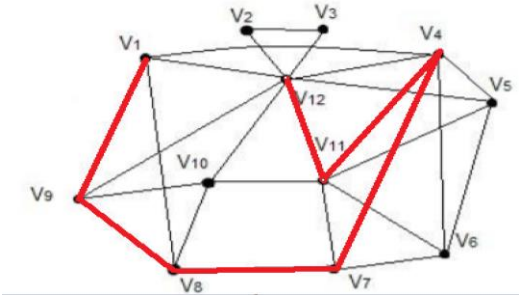
4


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

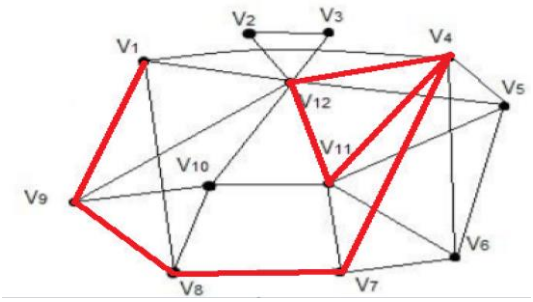
5


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

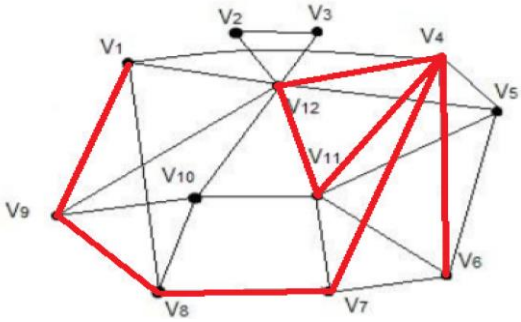
6


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

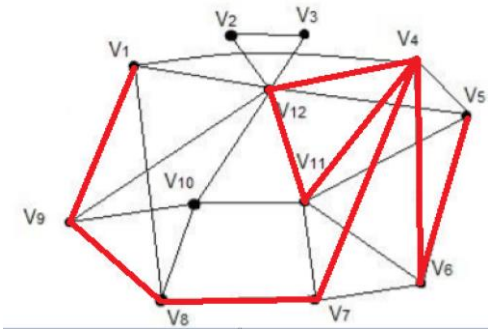
7


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

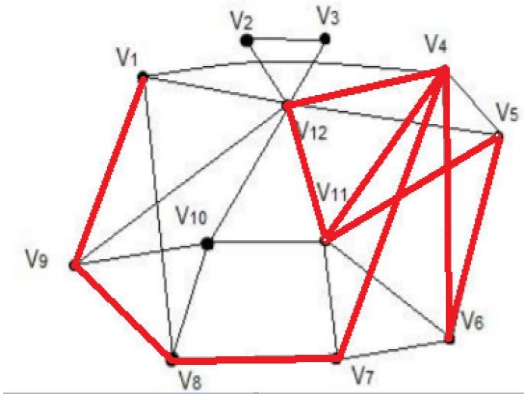
8


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

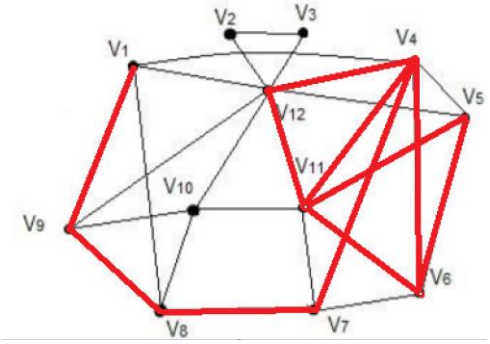
9


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

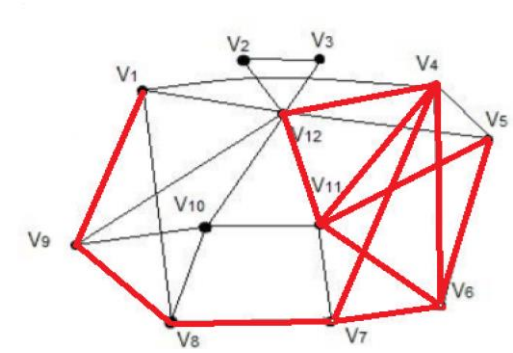
10


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

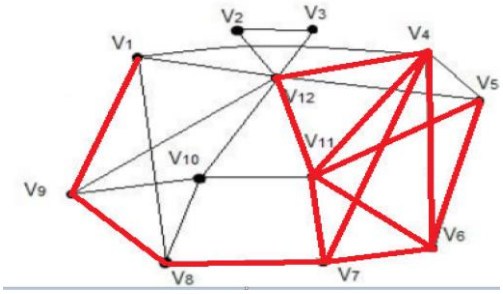
11


V2
V3
V1
V4
V5
V12
V11
V10
V9
V6
V8
V7

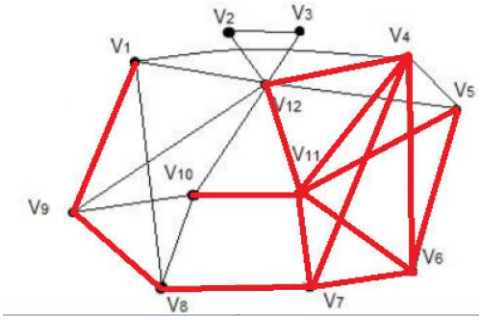
12


V2
V3
V1
V4
V5
V12
V11
V10
V9
V6
V8
V7

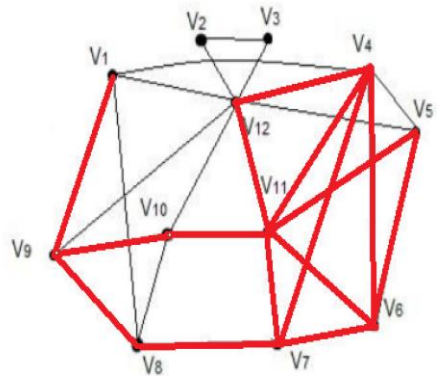
13


V2
V3
V1
V4
V5
V12
V11
V10
V9
V6
V8
V7

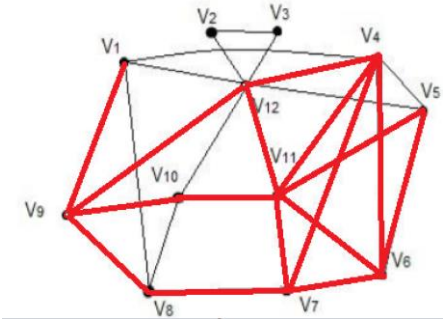
14


V2
V3
V1
V4
V5
V12
V11
V10
V9
V6
V8
V7

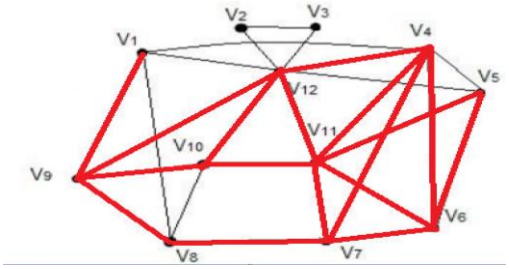
15


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

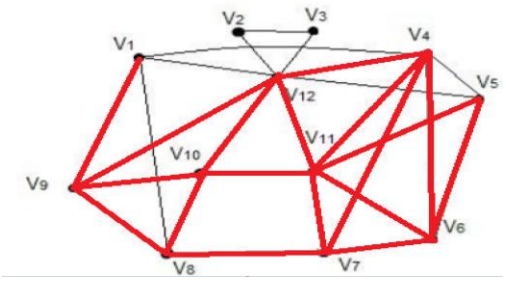
16


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

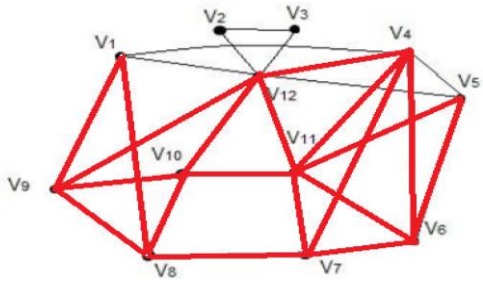
17


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

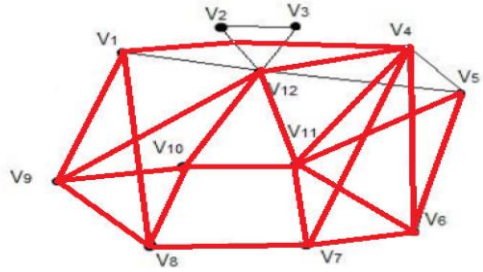
18


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

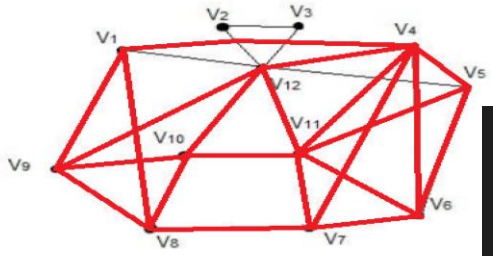
19



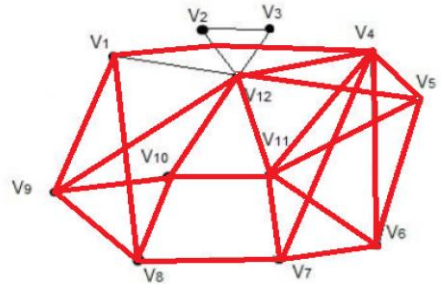
20



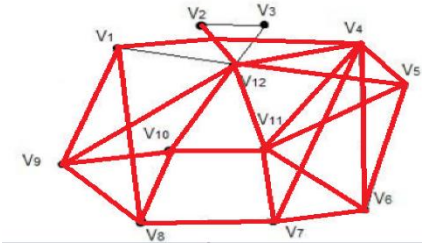
21



22



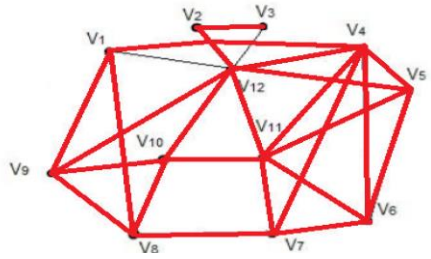
23



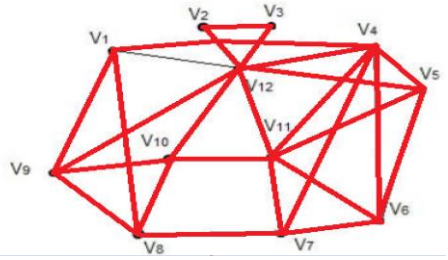

```cpp
#include<iostream>
#include<vector>
#define v 12
using namespace std;
int matrix[v][v] = {
    {0,0,0,0,0,0,0,1,0,1,0,0},
    {0,0,1,0,0,0,1,0,1,0,1,0},
    {0,1,0,1,0,1,0,1,0,1,0,1},
    {0,0,1,0,0,0,1,0,0,0,0,0},
    {0,0,0,0,0,1,0,0,0,0,0,1},
    {0,0,1,0,1,0,0,0,0,0,0,0},
    {0,1,0,1,0,0,0,1,0,0,0,1},
    {1,0,1,0,0,0,1,0,0,0,1,0},
    {0,1,0,0,0,0,0,0,0,1,0,0},
    {1,0,1,0,0,0,0,0,1,0,1,0},
    {0,1,0,0,0,0,0,1,0,1,0,1},
    {0,0,1,0,1,0,1,0,0,0,1,0}
};
int temp[v][v];
int findstartvr() {
    for (int i = 0; i < v; i++) {
        int stp = 0;
        for (int j = 0; j < v; j++) {
            if (temp[i][j])
                stp++;
        }
        if (stp % 2 != 0)
            return i;
    }
    return 0;
}
bool most(int u, int vr) {
    int stp = 0;
    for (int i = 0; i < v; i++)
        if (temp[vr][i])
```

```cpp
    for (int i = 0; i < v; i++)
        if (temp[vr][i])
            stp++;
    if (stp > 1) {
        return false;
    }
    return true;
}
int edgecount() {
    int counter = 0;
```

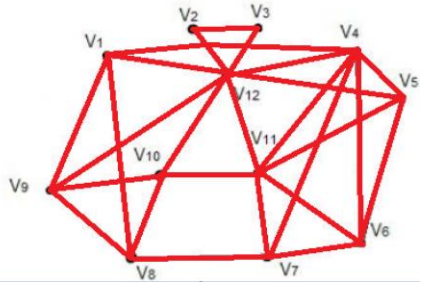
24



25

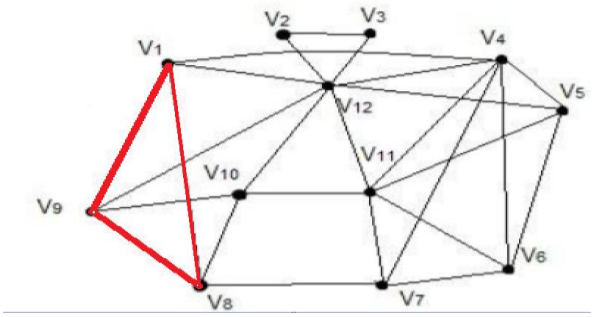


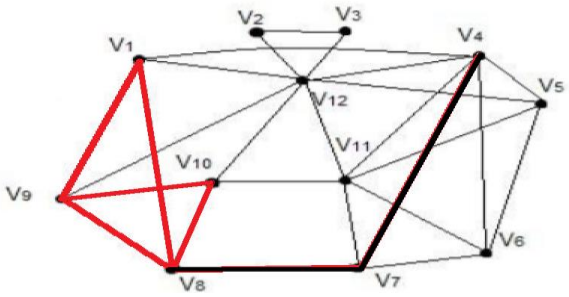
26

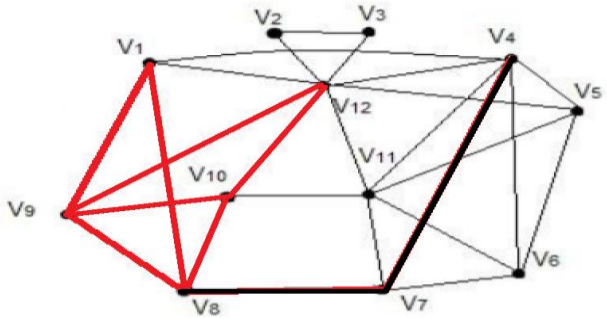


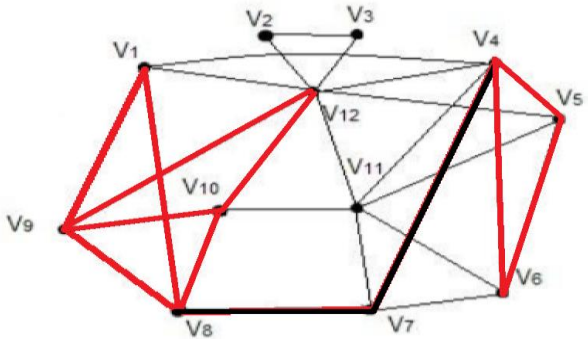

```
1->9->8->7->4->11->12->4->6->5->11->6->7->11->10->9->12->10->8->1->4->5->12->2->3->12->1
```

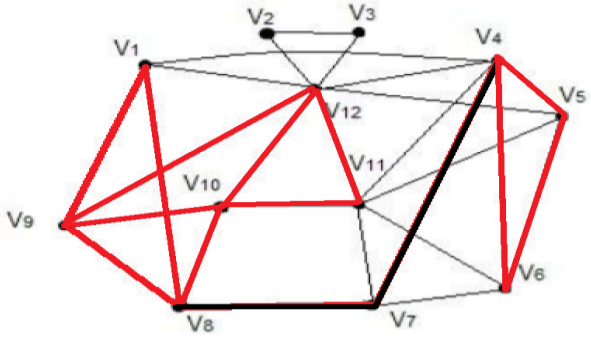
метод елементарних циклів:

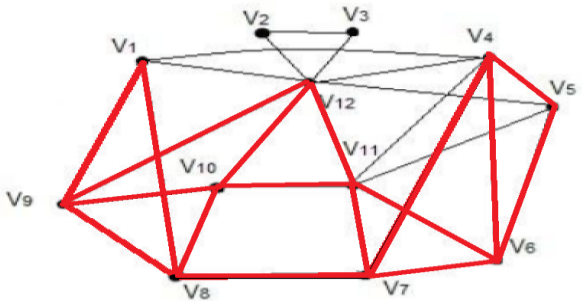


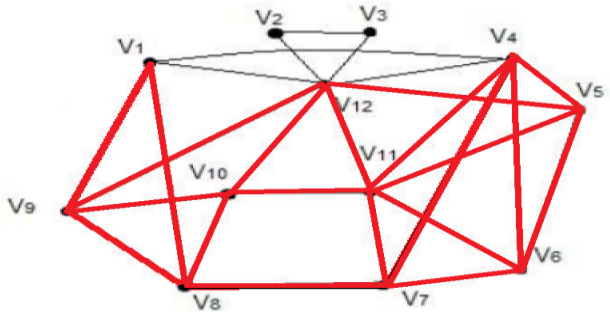

V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

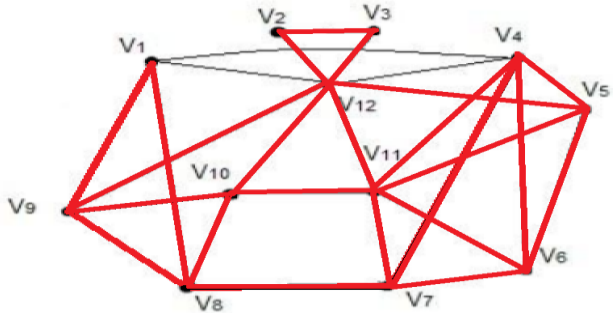

V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12

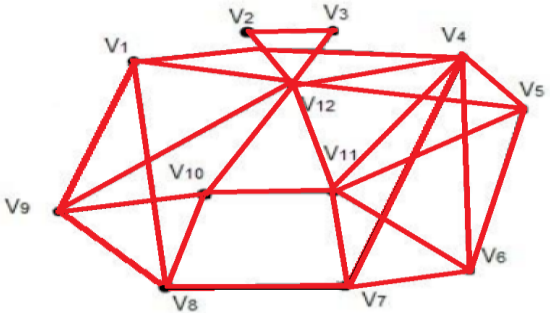

V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12







Знайдено всі елементарні цикли, ось результат їх поєднання:



```cpp
#include <iostream>
#include <vector>
using namespace std;
vector<int> Vcon;
int Inf = 999;
bool check(vector<int> V, int pork) {
    for (auto i = V.begin(); i != V.end(); i++) {
        if (*i == pork)return false;
    }
    return true;
}
void Find(vector<int>* V, int** arr, int n, int pos, int start_pork) {
    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_pork && (*V).size() > 2) {
                    if (Inf > V->size()) {
                        Vcon.clear();
                        Vcon.push_back(start_pork + 1);
                        for (auto it = (*V).begin(); it != (*V).end();
                            it++)
                            Vcon.push_back(*it + 1);
                        Vcon.push_back(start_pork + 1);
                        Inf = V->size();
                        break;
                    }
                }
                else {
                    (*V).push_back(j);
                    Find(V, arr, n, j, start_pork);
                }
            }
    }
    if (V->size() != 0)
        V->pop_back();
}
int main() {
    int n;
    cout << "Enter number of porks: ";
    cin >> n;
    int** arr = new int* [n];
```

```cpp
    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++) {
            Inf = 999;
            Find(&V, arr, n, j, j);
            for (int i = 1; i <= Vcon.size(); i++) {
                cout << Vcon[i - 1] << " ";
            }
            cout << endl;
            Vcon.clear();
        }
    }
    else
        cout << "tikai z sela\n";
    cout << endl;
    return 0;
}
```

```
Enter number of porks: 12
0 0 0 1 0 0 0 1 1 0 0 1
0 0 1 0 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 1 1 0 0 0 1 1
0 0 0 1 0 1 0 0 0 0 1 1
0 0 0 1 1 0 1 0 0 0 1 0
0 0 0 1 0 1 0 1 0 0 1 0
1 0 0 0 0 0 1 0 1 1 0 0
1 0 0 0 0 0 0 1 0 1 0 1
0 0 0 0 0 0 0 1 1 0 1 1
0 0 0 1 1 1 1 0 0 1 0 1
1 1 1 1 1 0 0 0 1 1 1 0


1 4 5 12 1


4 1 8 7 4
5 4 1 12 5
6 4 5 11 6
7 4 1 8 7
8 1 4 7 8
9 1 4 12 9
10 8 1 9 10
11 4 1 12 11
12 1 4 5 12
```

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

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

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 ∨ x ∨ y=x ∨ y;
  Відповідь : x ∨ y;