МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

З дисципліни

«Дискретна математика»

Виконав:

студент групи КН-112

Матвіїв Остап-Василь

Викладач:

Мельникова Н.І.

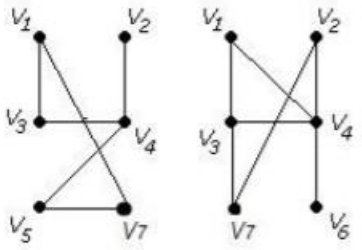
Львів – 2019 р.

Варіант-17

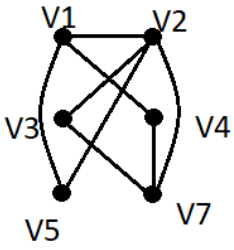
Завдання № 1

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.

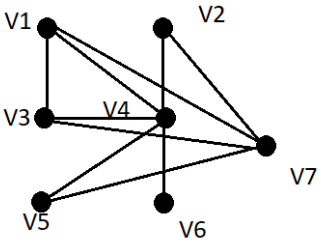
17)



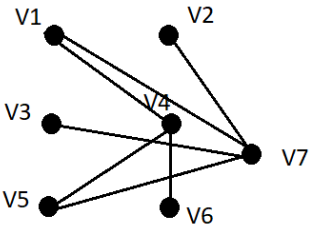
1)



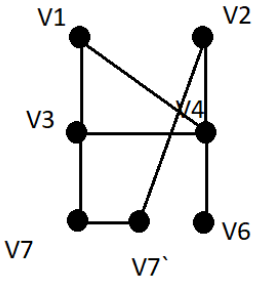
2)



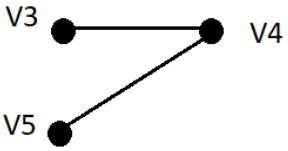
3)

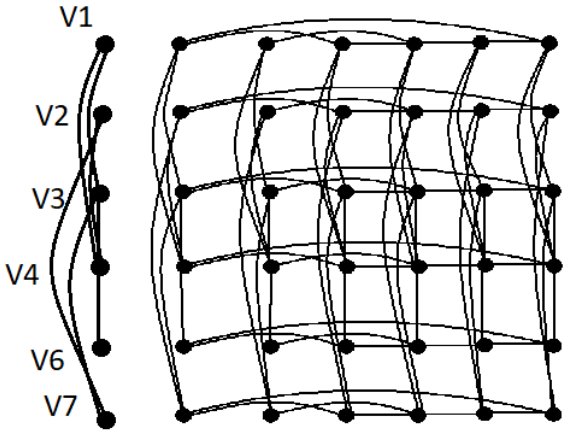


4)



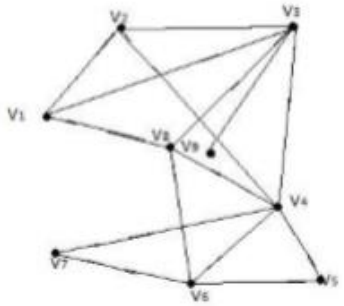
5)





6)

17)



Завдання № 2

Скласти таблицю суміжності для орграфа.

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

Завдання № 3

Для графа з другого завдання знайти діаметр.

Найдовша відстань між будь якими двома вершинами = 3

Діаметр = 3

Завдання № 4

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

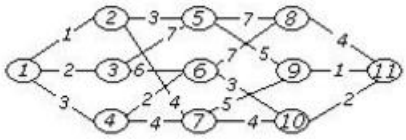
```
#include <iostream>
using namespace std;
const int n = 10;
int i, j;
bool* visited = new bool[n];
int graph[n][n] =
{
{0,1,1,0,0,0,0,1,0},
{1,0,1,1,0,0,0,0,0},
{1,1,0,1,0,0,0,0,0},
{0,1,1,0,1,1,1,1,0},
{0,0,0,1,0,1,0,0,0},
{0,0,0,1,1,0,1,1,0},
{0,0,0,1,0,1,0,0,0},
{1,0,1,1,0,1,0,0,0},
{0,0,1,0,0,0,0,0,0}
};
void DFS(int st)
{
    int r;
    cout << st + 1 << " ";
    visited[st] = true;
    for (r = 0; r <= n; r++)
        if ((graph[st][r] != 0) && (!visited[r]))
            DFS(r);
}
void main()
{
    setlocale(LC_ALL, "Ukr");
    int start;
    cout << "Матриця суміжності: " << endl;
    for (i = 0; i < n; i++)
    {
        visited[i] = false;
        for (j = 0; j < n; j++)
            cout << " " << graph[i][j];
        cout << endl;
    }
    cout << "Стартова вершина >> "; cin >> start;
    bool* vis = new bool[n];
    cout << "Порядок обходу: ";
    DFS(start - 1);
    delete[]visited;
}
```

```
Матриця сум?жност?:
 0 1 1 0 0 0 0 1 0 0
 1 0 1 1 0 0 0 0 0 0
 1 1 0 1 0 0 0 0 0 0
 0 1 1 0 1 1 1 1 0 0
 0 0 0 1 0 1 0 0 0 0
 0 0 0 1 1 0 1 1 0 0
 0 0 0 1 0 1 0 0 0 0
 1 0 1 1 0 1 0 0 0 0
 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
Стартова вершина >> 5
Порядок обходу: 5 4 2 1 3 8 6 7 9
```

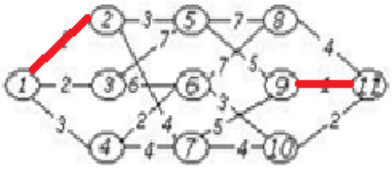
Завдання № 5

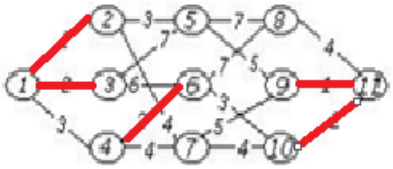
Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

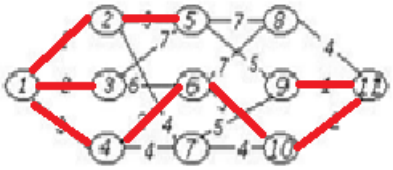
17)

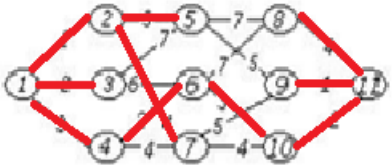


1) Метод Краскала:





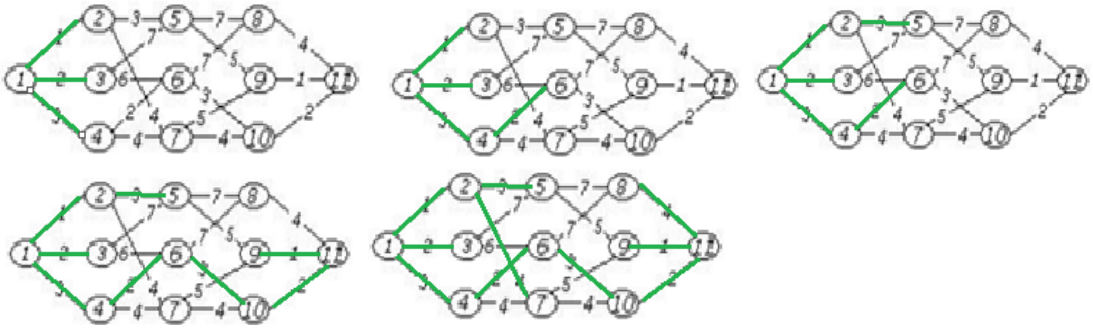




```cpp
#include <iostream>
#define INT_MAX 2147483647

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-%d) \t cost:%d \n", edge_count++ + 1, a + 1, b + 1, min);
        min_cost += min;
    }
    printf("\nTotal weight= %d \n", min_cost);
}
int main()
{
    int cost[][v] = {
        {INT_MAX,1,2,3,INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX},
        {1,INT_MAX,INT_MAX,INT_MAX,3,INT_MAX,4,INT_MAX,INT_MAX,INT_MAX,INT_MAX},
        {2,INT_MAX,INT_MAX,INT_MAX,7,6,INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX},
        {3,INT_MAX,INT_MAX,INT_MAX,INT_MAX,2,4,INT_MAX,INT_MAX,INT_MAX,INT_MAX},
        {INT_MAX,3,7,INT_MAX,INT_MAX,INT_MAX,INT_MAX,7,5,INT_MAX,INT_MAX},
        {INT_MAX,INT_MAX,6,2,INT_MAX,INT_MAX,INT_MAX,7,INT_MAX,3,INT_MAX},
        {INT_MAX,4,INT_MAX,4,INT_MAX,INT_MAX,INT_MAX,INT_MAX,5,4,INT_MAX},
        {INT_MAX,INT_MAX,INT_MAX,INT_MAX,7,7,INT_MAX,INT_MAX,INT_MAX,INT_MAX,4},
        {INT_MAX,INT_MAX,INT_MAX,INT_MAX,5,INT_MAX,5,INT_MAX,INT_MAX,INT_MAX,1},
        {INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX,3,4,INT_MAX,INT_MAX,INT_MAX,2},
        {INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX,4,1,2,INT_MAX}

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

```
Edge 1:(1-2)     cost:1
Edge 2:(9-11)    cost:1
Edge 3:(1-3)     cost:2
Edge 4:(4-6)     cost:2
Edge 5:(10-11)   cost:2
Edge 6:(1-4)     cost:3
Edge 7:(2-5)     cost:3
Edge 8:(6-10)    cost:3
Edge 9:(2-7)     cost:4
Edge 10:(8-11)   cost:4

Total weight= 25
```

2)Метод Прима:



```cpp
#include <iostream>
using namespace std;
int main()
{
    int graph[11][11] = {   {0,1,2,3,0,0,0,0,0,0,0},
                            {1,0,0,0,3,0,4,0,0,0,0},
                            {3,0,0,0,7,6,0,0,0,0,0},
                            {3,0,0,0,0,2,4,0,0,0,0},
                            {0,3,7,0,0,0,0,7,5,0,0},
                            {0,0,6,2,0,0,0,7,0,3,0},
                            {0,4,0,4,0,0,0,0,5,4,0},
                            {0,0,0,0,7,7,0,0,0,0,4},
                            {0,0,0,0,5,0,5,0,0,0,1},
                            {0,0,0,0,0,3,4,0,0,0,2},
                            {0,0,0,0,0,0,0,4,1,2,0}};
    int n = 11, sumweight = 0;
    bool* visited = new bool[n];
    memset(visited, false, sizeof(bool) * n);
    visited[0] = true;
    for (int l = 0; l < n - 1; l++) {
        int minx = -1, miny = -1;
        for (int i = 0; i < n; i++) {
            if (visited[i]) {
                for (int j = 0; j < n; j++) {
                    if (!visited[j] && graph[i][j] > 0 && (minx == -1 || graph[i][j] < graph[minx][miny]))
                        static_cast<void>(miny = i), minx = j;
                    }
                }
            }
        }
        visited[minx] = true;
        sumweight += graph[miny][minx];
        cout << miny + 1 << "-" << minx + 1 << " weight: " << graph[miny][minx] << endl;
    }
    cout << "Total weight: " << sumweight << endl;
    return 0;
}
```

```
1-2 weight: 1
1-3 weight: 2
1-4 weight: 3
4-6 weight: 2
2-5 weight: 3
6-10 weight: 3
10-11 weight: 2
11-9 weight: 1
2-7 weight: 4
11-8 weight: 4
Total weight: 25
```

17)

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

Завдання № 6

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

```
Вершина 1:
1->5->2->3->8->4->7->6->1
Вага: 19
```

1->6->7->4->8->3->2->5->1
Вага: 19

Вершина 2:
2->3->8->4->7->6->1->5->2
Вага: 19
2->5->1->6->7->4->8->3->2
Вага: 19

Вершина 3:
3->2->5->1->6->7->4->8->3
Вага: 19
3->8->4->7->6->1->5->2->3
Вага: 19

Вершина 4:
4->7->6->1->5->2->3->8->4
Вага: 19
4->8->3->2->5->1->6->7->4
Вага: 19

Вершина 5:
5->1->6->7->4->8->3->2->5
Вага: 19
5->2->3->8->4->7->6->1->5
Вага: 19

Вершина 6:
6->1->5->2->3->8->4->7->6
Вага: 19
6->7->4->8->3->2->5->1->6
Вага: 19

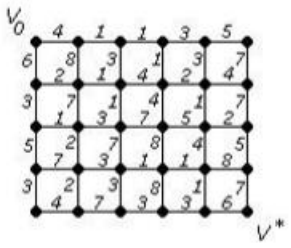
Вершина 7:
7->4->8->3->2->5->1->6->7
Вага: 19
7->6->1->5->2->3->8->4->7
Вага: 19

Вершина 8:
8->3->2->5->1->6->7->4->8
Вага: 19
8->4->7->6->1->5->2->3->8
Вага: 19

```cpp
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <string>
#include <algorithm>
 using namespace std;
class Komivoiser {
public:
    string name;
    int number;
    Komivoiser() {}
};


int main()
{
    setlocale(LC_ALL, "Ukrainian");
    int v = 0;
    cout << "Кількість вершин: ";
    cin >> v;
    int** graph = new int* [v];
    for (int j = 0; j < v; j++) {
        graph[j] = new int[v];
    }
    cout << "Вага ребер: " << endl;
    for (int a = 0; a < v; a++) {
        for (int j = 0; j < v; j++) {
            cin >> graph[a][j];
        }
    }
    int* a = new int[v];
    for (int i = 0; i < v; i++)
        a[i] = i + 1;
    int n = sizeof(a) / sizeof(a[0]);
    vector<Komivoiser> Path;
    int min_path = 0;
    sort(a, a + v);
    for (int i = 1; i < v; i++) {
        min_path += graph[a[i - 1] - 1][a[i] - 1];
    }
    min_path += graph[a[v - 1] - 1][a[0] - 1];
    do {
        Komivoiser t;
        t.name = to_string(a[0]); t.number = 0;
        for (int i = 1; i < v; i++) {
            t.name += "->" + to_string(a[i]);
            t.number += graph[a[i - 1] - 1][a[i] - 1];
        }
        t.name += "->" + to_string(a[0]);
      t.number += graph[a[v - 1] - 1][a[0] - 1];
      Path.push_back(t);
      if (min_path > t.number) min_path = t.number;
    } while (next_permutation(a, a + v));
    cout << "Оптимальні шляхи: " << endl;
    for (int i = 0; i < Path.size(); i++) {
        if (Path[i].number == min_path) {
            cout << "Path: " << Path[i].name << " weight: " << Path[i].number << endl;
        }
    }
    return 0;
}
```

```
К?льк?сть вершин: 8
Вага ребер:
0 6 6 6 1 3 1 3
6 0 5 5 1 6 1 5
6 5 0 7 7 7 7 5
6 5 7 0 6 5 1 2
1 1 7 6 0 6 6 6
3 6 7 5 6 0 1 2
1 1 7 1 6 1 0 2
3 5 5 2 6 2 2 0
Оптимальн? шляхи:
Path: 1->5->2->3->8->4->7->6->1 weight: 19
Path: 1->6->7->4->8->3->2->5->1 weight: 19
Path: 2->3->8->4->7->6->1->5->2 weight: 19
Path: 2->5->1->6->7->4->8->3->2 weight: 19
Path: 3->2->5->1->6->7->4->8->3 weight: 19
Path: 3->8->4->7->6->1->5->2->3 weight: 19
Path: 4->7->6->1->5->2->3->8->4 weight: 19
Path: 4->8->3->2->5->1->6->7->4 weight: 19
Path: 5->1->6->7->4->8->3->2->5 weight: 19
Path: 5->2->3->8->4->7->6->1->5 weight: 19
Path: 6->1->5->2->3->8->4->7->6 weight: 19
Path: 6->7->4->8->3->2->5->1->6 weight: 19
Path: 7->4->8->3->2->5->1->6->7 weight: 19
Path: 7->6->1->5->2->3->8->4->7 weight: 19
Path: 8->3->2->5->1->6->7->4->8 weight: 19
Path: 8->4->7->6->1->5->2->3->8 weight: 19
```

17)



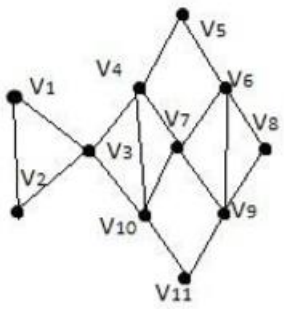
Завдання № 7

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

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#define SIZE 30
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("Введіть матрицю суміжності:\n");
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++) {
            scanf("%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("\nНайкоротші відстані до вершин: \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("\nВивід найкоротшого шляху:");
    for (int i = k - 1; i >= 0; i--)
        printf("%3d ", ver[i]);
    return 0;
}
```

```
Введіть матрицю суміжності:
0 4 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 0 1 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 3 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 3 0 5 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 5 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 2 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 8 0 0 0 0 2 0 1 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 4 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 5 0 0 0 0 4 0 2 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 3 0 0 0 0 2 0 4 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 7 0 0 0 0 4 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 3 0 0 0 0 0 0 1 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 7 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 0 1 0 0 0 0 3 0 7 0 0 0 0 7 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 4 0 0 0 0 7 0 5 0 0 0 0 8 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 5 0 2 0 0 0 0 4 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 2 0 0 0 0 0 0 5 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 7 0 0 0 0 3 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 7 0 3 0 0 0 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 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 0 8 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 0 4 0 0 0 0 1 0 8 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 8 0 0 0 0 0 7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 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 2 0 0 0 0 4 0 7 0 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 7 0 3 0
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 3 0 3
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 6
0 0 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 6 0

Найкоротші відстані до вершин:
  0  4  5  6  9 14  1  3  4  8 10 14  4  5  5 12 11 13  9  7 10 11 12 18 12  9 13 16 13 19
Вивід найкоротшого шляху:  1   7  13  14  20  21  22  23  29  30
```

17)

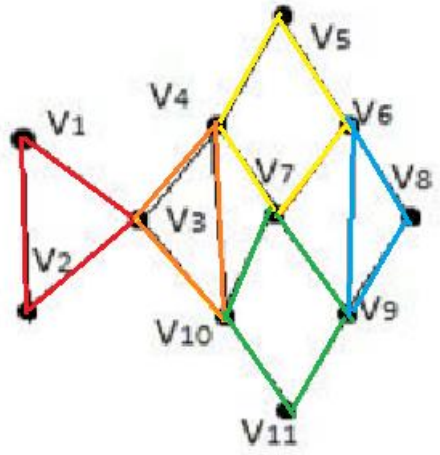


Завдання № 8

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

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

б)1=>2=>3=>4=>5=>6=>7=>4=>10=>7=>9=>8=>6=>9=>11=>10=>3=>1




```cpp
#include <iostream>
#include <vector>
#define NODE 132
using namespace std;
int graph[NODE][NODE]{
    {0,1,1,0,0,0,0,0,0,0,0},
    {1,0,1,0,0,0,0,0,0,0,0},
    {1,1,0,1,0,0,0,0,0,1,0},
    {0,0,1,0,1,0,1,0,0,1,0},
    {0,0,0,1,0,1,0,0,0,0,0},
    {0,0,0,0,1,0,1,1,1,0,0},
    {0,0,0,1,0,1,0,0,1,1,0},
    {0,0,0,0,0,1,0,0,1,0,0},
    {0,0,0,0,0,1,1,1,0,0,1},
    {0,0,1,1,0,0,1,0,0,0,1},
    {0,0,0,0,0,0,0,0,1,1,0}
};
int tempGraph[NODE][NODE];
int findStartVert() {
    for (int i = 1; i < NODE; i++) {
        int deg = 0;
        for (int j = 0; j < NODE; j++) {
            if (tempGraph[i][j])
                deg++;
        }
        if (deg % 2 != 0)
            return i;
    }
    return 0;
}
bool isBridge(int u, int v) {
    int deg = 0;
    for (int i = 0; i < NODE; i++)
        if (tempGraph[v][i])
            deg++;
        if (deg > 1) {
            return false;
        }
        return true;
}
int edgeCount() {
    int count = 0;
    for (int i = 0; i < NODE; i++)
        for (int j = i; j < NODE; j++)
            if (tempGraph[i][j])
                count++;
    return count;
}
void fleuryAlgorithm(int start) {
    static int edge = edgeCount();
    for (int v = 0; v < NODE; v++) {
        if (tempGraph[start][v]) {
            if (edge <= 1 || !isBridge(start, v)) {
                cout << start + 1 << '-' << v + 1 << endl;
                tempGraph[start][v] = tempGraph[v][start] = 0;
                edge--;
                fleuryAlgorithm(v);
            }
        }
    }
}

int main()
{
    for (int i = 0; i < NODE; i++)
        for (int j = 0; j < NODE; j++)
            tempGraph[i][j] = graph[i][j];
    cout << "Euler Path Or Circuit: "<<endl;
    fleuryAlgorithm(findStartVert());
}
```

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

Завдання №9

Спростити формули (привести їх до скороченої ДНФ). 17. $x\bar{y} \vee \bar{x}\bar{z} \vee yz$

X	Y	Z	$\overline{Y}$	$X\overline{Y}$	$\overline{X}$	$\overline{Z}$	$\overline{X}\overline{Z}$	$X\overline{Y} \vee \overline{X}\overline{Z}$	YZ	$X\overline{Y} \vee \overline{X}\overline{Z} \vee YZ$
0	0	0	1	0	1	1	1	1	0	1
0	0	1	1	0	1	0	0	0	0	0
0	1	0	0	0	1	1	1	1	0	1
0	1	1	0	0	1	0	0	0	1	1
1	0	0	1	1	0	1	0	1	0	1
1	0	1	1	1	0	0	0	1	0	1
1	1	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1

Карта Карно:

X\YZ	00	01	11	10
0	1	0	1	1
1	1	1	1	0

СДНФ:

$XZ \vee \overline{X}Y \vee \overline{Y}\overline{Z}$