МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ
“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

з дисципліни

«Дискретна математика»

Виконав:

Студент групи КН-113

Добосевич Данило

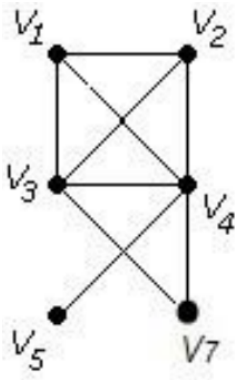
Перевірила:

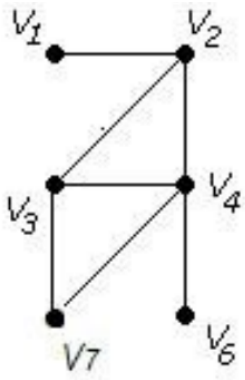
Мельникова Н.І.

Львів - 2019

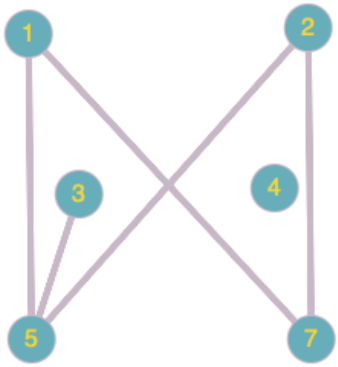
Варіант №15

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.

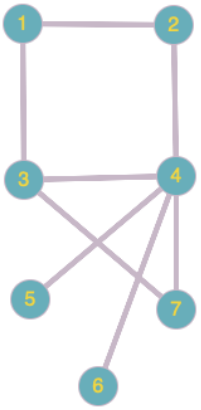




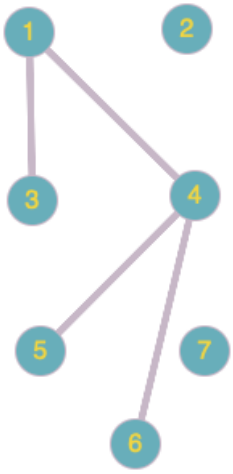
1) Доповнення до 1 графу



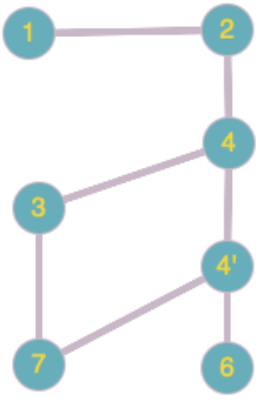
2) Об’єднання графів



3) Кільцева сума G1 та G2 (G1+G2)

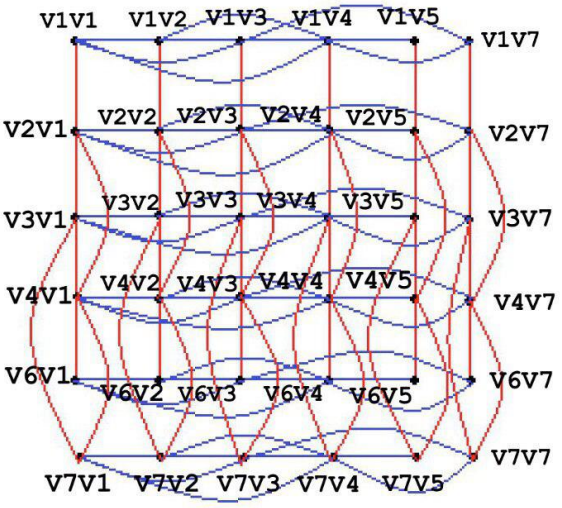


4) розмножити вершину другому графі

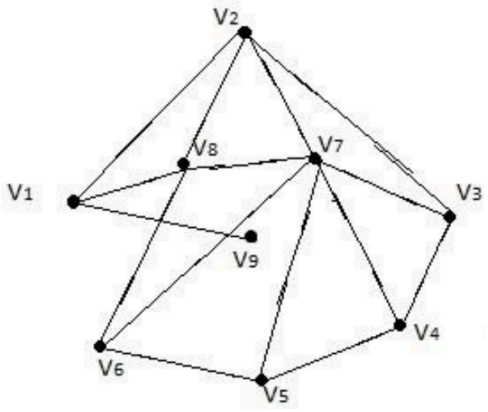


5) виділити підграф A - що скадається з 3-х вершин в G1

6) добуток графів



Завдання №2 Скласти таблицю суміжності для орграфа.



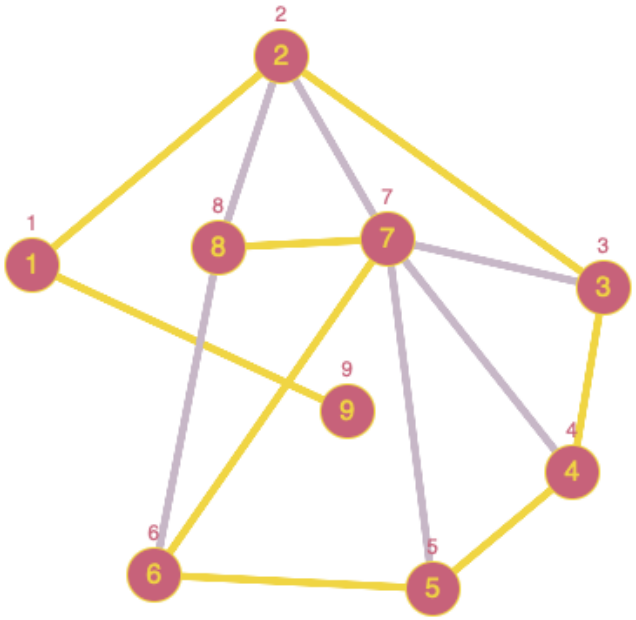
	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	0	0	0	0	0	1	1
V2	1	0	1	0	0	0	1	1	0
V3	0	1	0	1	0	0	1	0	0
V4	0	0	1	0	1	0	1	0	0
V5	0	0	0	1	0	1	1	0	0
V6	0	0	0	0	1	0	1	1	0
V7	0	1	1	1	1	1	0	1	0
V8	1	1	0	0	0	1	1	0	0
V9	1	0	0	0	0	0	0	0	0

Завдання №3 Для графа з другого завдання знайти діаметр.

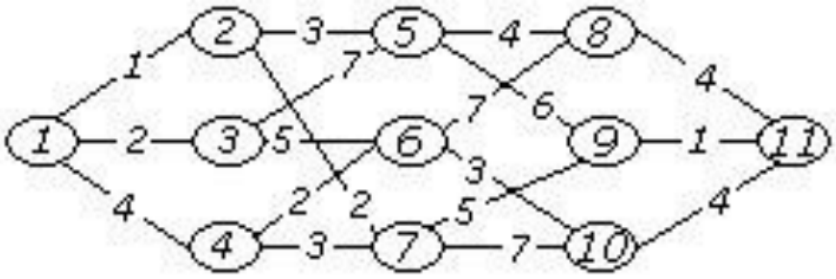
Діаметр графа: 4

V4 -> V7 -> V8 -> V1 -> V9

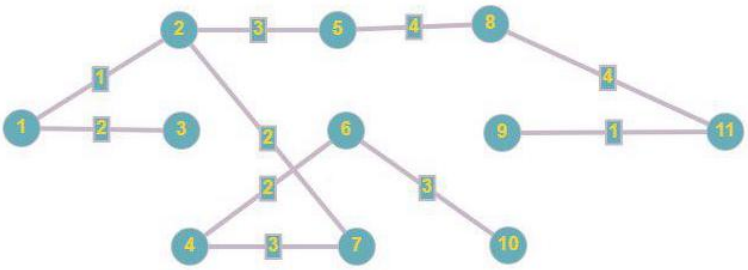
Завдання №4 Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).



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



Алгоритм Краскала:



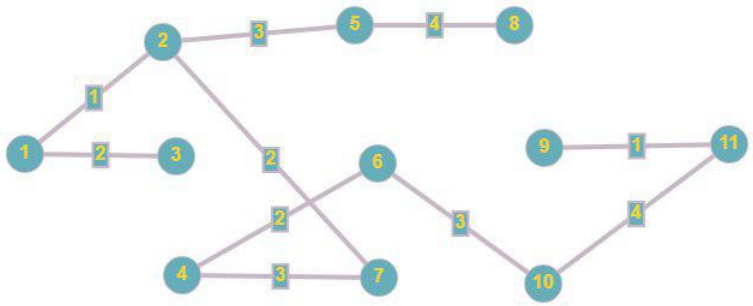

```c
#include <stdio.h>
#define a_rebr 18
#define a_top 11
#define c_1 4
#define c_2 2

int main (void)
{
    int i,j,k,a;
    int rebro [a_rebr][c_1] =
            {
                    {1, 2, 1, 0}, //point from, point to, weight, +/- in tree
                    {1, 3, 2, 0},
                    {1, 4, 4, 0},
                    {2, 5, 3, 0},
                    {2, 7, 2, 0},
                    {3, 5, 7, 0},
                    {3, 6, 5, 0},
                    {4, 6, 2, 0},
                    {4, 7, 3, 0},
                    {5, 8, 4, 0},
                    {5, 9, 6, 0},
                    {6, 8, 7, 0},
                    {6, 10, 3, 0},
                    {7, 9, 5, 0},
                    {7, 10, 7, 0},
                    {8, 11, 4, 0},
                    {9, 11, 1, 0},
                    {10, 11, 4, 0}
            };

    int top [a_top][c_2] =
            {
                    {1, 0},
                    {2, 0},
                    {3, 0},
                    {4, 0},
                    {5, 0},
                    {6, 0},
                    {7, 0},
                    {8, 0},
                    {9, 0},
                    {10, 0},
                    {11, 0}
            };

    //sort
    for (i = 0; i < a_rebr; i++)
    {
        for (j = 0; j < a_rebr - i - 1; j++)
        {
            if (rebro [j][2] > rebro [j + 1][2])
            {
                for (k = 0, a = 0; k < c_1; k++)
                {
                    a = rebro [j + 1][k];
                    rebro [j + 1][k] = rebro [j][k];
                    rebro [j][k] = a;
                }
            }
        }
    }

    int count = 1;
    for (i = 0; i < a_rebr; i++) {
        if (top [rebro[i][0] - 1][1] != top [rebro[i][1] - 1][1]) {
            rebro [i][3] = 1;
            if (top [rebro[i][0] - 1][1] == 0 && top [rebro[i][1] - 1][1] != 0) {
                top [rebro[i][0] - 1][1] = top [rebro[i][1] - 1][1];
            }
            if (top [rebro[i][0] - 1][1] != 0 && top [rebro[i][1] - 1][1] == 0) {
                top [rebro[i][1] - 1][1] = 1;
            }
            if (top [rebro[i][0] - 1][1] != 0 && top [rebro[i][1] - 1][1] != 0) {
                for (j = 0; j < a_top; j++)
                {
                    if (top [j][1] == top [rebro[i][1] - 1][1])
                    {
                        top [j][1] = top [rebro[i][0] - 1][1];
                    }
                }
```

```
            }
        }
        if (top [rebro[i][0] - 1][1] == 0 && top [rebro[i][1] - 1][1] == 0)
        {
            rebro [i][3] = 1;
            top [rebro[i][0] - 1][1] = count;
            top [rebro[i][1] - 1][1] = count;
            count++;
        }
    }

    for (i = 0; i < a_rebr; i++)
    {
        if (rebro[i][3] == 1)
        {
            printf ("%i - %i (%i)\n", rebro[i][0], rebro[i][1], rebro [i][2]);
        }
    }

    return 0;
}
```

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

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

Алгоритм Прима



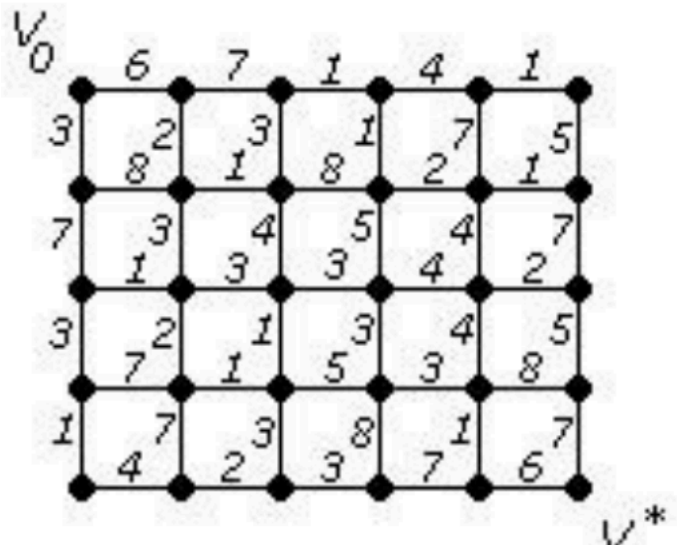
1-2 -> 1-3 -> 2-7 -> 2-5 -> 7-4 -> 4-6 -> 6-10 -> 5-8 -> 10-11 -> 11-9

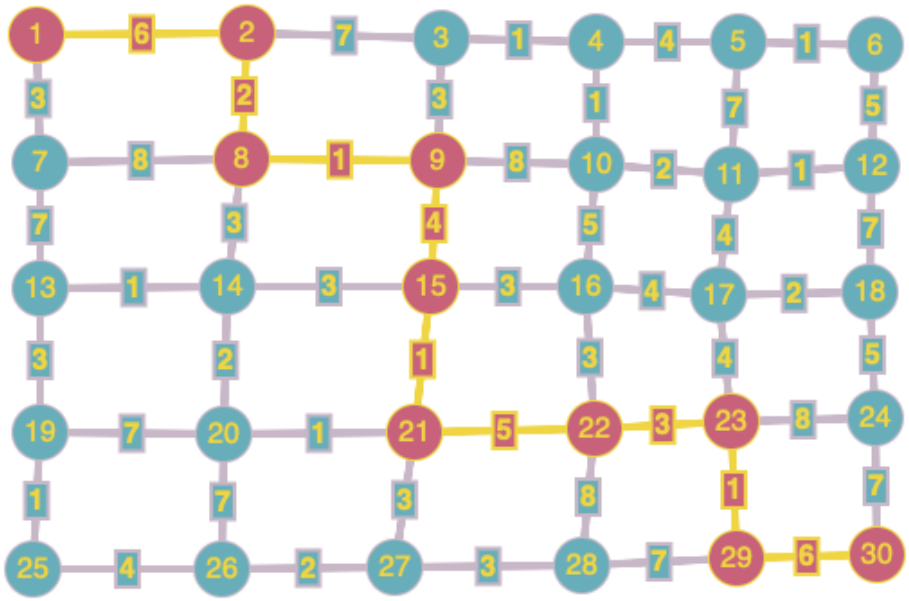
```cpp
#include <iostream>

using namespace std;

int main(void)
{
    int versh, cnt = 0, min_ = 0, n, m;
    bool c = false;

    cout << "The quantity of tops: "; cin >> versh; cout << "\nMatrix: \n";

    int **graph = new int*[versh];
    for(int i = 0; i < versh; ++i)
        graph[i] = new int[versh];

    int **rebr = new int*[versh - 1];
    for(int i = 0; i < versh - 1; ++i)
        rebr[i] = new int[2];

    for(int i = 0; i < versh; ++i)
        for (int j = 0; j < versh; ++j)
            cin >> graph[i][j];

    int *tops = new int[versh];
    tops[cnt] = 1;
    ++cnt;

    for(int i = 0; cnt < versh; ++i){
        for(int j = 0; j < cnt; ++j){
            for(int x = 0; x < versh; ++x){
                for(int y = 0; y < cnt; ++y)
                    if(tops[y] == x + 1)
                        c = true;
                if(c == true)
                {
                    c = false;
                    continue;
                }

                if(min_ == 0 && graph[tops[j] - 1][x] > 0)
                {
                    min_ = graph[tops[j] - 1][x];
                    n = rebr[cnt - 1][0] = tops[j];
                    m = rebr[cnt - 1][1] = x + 1;
                    continue;
                }

                if(graph[tops[j] - 1][x] > 0 && graph[tops[j] - 1][x] < min_)
                {
                    min_ = graph[tops[j] - 1][x];
                    n = rebr[cnt - 1][0] = tops[j];
                    m = rebr[cnt - 1][1] = x + 1;
                }
            }
        }

        graph[n - 1][m - 1] = 0;
        graph[m - 1][n - 1] = 0;
        tops[cnt] = m;
        ++cnt;
        min_ = 0;
    }

    cout << endl << "Rebra: ";
    for (int i = 0; i < versh - 1; ++i)
        cout << "(" << rebr[i][0] << ", " << rebr[i][1] << ") ";

}
```

```
The quantity of tops: 11

Matrix:
0 1 2 4 0 0 0 0 0 0 0
1 0 0 0 3 0 2 0 0 0 0
2 0 0 0 7 5 0 0 0 0 0
4 0 0 0 0 2 3 0 0 0 0
0 3 7 0 0 0 0 4 6 0 0
0 0 5 2 0 0 0 7 0 3 0
0 2 0 3 0 0 0 0 5 7 0
0 0 0 0 4 7 0 0 0 0 4
0 0 0 0 6 0 5 0 0 0 1
0 0 0 0 0 6 7 0 0 0 4
0 0 0 0 0 0 0 4 1 4 0

Rebra: (1, 2) (1, 3) (2, 7) (2, 5) (7, 4) (4, 6) (6, 10) (5, 8) (10, 11) (11, 9)
```

Завдання №7 За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V*.

1 -> 2 -> 8 ->9 -> 15 -> 21 -> 22 -> 23 -> 29 -> 30 =30





```c
#include <stdio.h>

void print_the_shortest_way(int array [30][4],int node) {
    if(array[node][3]==-1) {
        printf("V%d=%d ",array[node][0],array[node][2]);
    }
    else {
        print_the_shortest_way(array, node: array[node][3]);
        printf("V%d=%d ",array[node][0],array[node][2]);
    }
}

int main(void) {
    int i,k,lenth=0;
    int result [30] [2];
    int nodes [30] [4] = { //{number,times_used,path_value,last_node}
        {1,1,0,-1},{2,0,0,0},{3,0,0,0},{4,0,0,0},{5,0,0,0},{6,0,0,0},
        {7,0,0,0},{8,0,0,0},{9,0,0,0},{10,0,0,0},{11,0,0,0},{12,0,0,0},
        {13,0,0,0},{14,0,0,0},{15,0,0,0},{16,0,0,0},{17,0,0,0},{18,0,0,0},
        {19,0,0,0},{20,0,0,0},{21,0,0,0},{22,0,0,0},{23,0,0,0},{24,0,0,0},
        {25,0,0,0},{26,0,0,0},{27,0,0,0},{28,0,0,0},{29,0,0,0},{30,0,0,0}
    };
    int edges [49] [3] = { //{from,to,weight}
        {1,2,6},{2,3,7},{3,4,1},{4,5,1},{5,6,1},
        {1,7,3},{2,8,2},{3,9,3},{4,10,1},{5,11,7},{6,12,5},
        {7,8,8},{8,9,1},{9,10,8},{10,11,2},{11,12,1},
        {7,13,7},{8,14,3},{9,15,4},{10,16,5},{11,17,4},{12,18,7},
        {13,14,1},{14,15,3},{15,16,3},{16,17,4},{17,18,2},
        {13,19,3},{14,20,2},{15,21,1},{16,22,3},{17,23,4},{18,24,5},
        {19,20,7},{20,21,1},{21,22,5},{22,23,3},{23,24,8},
        {19,25,1},{20,26,7},{21,27,3},{22,28,8},{23,29,1},{24,30,7},
        {25,26,4},{26,27,2},{27,28,3},{28,29,7},{29,30,6}
    };
    //sorting edges
    int temp,pointer;
    do{
        pointer=0;
        for (i = 0; i < 48; i++){
            if (edges[i][2] > edges[i + 1][2]) {
                for (k = 0, temp = 0; k < 3; k++) {
                    temp = edges [i + 1][k];
                    temp = edges [i + 1][k];
                    edges [i + 1][k] = edges [i][k];
                    edges [i][k] = temp;
                }
                pointer++;
            }
        }
    }while(pointer!=0);

    while(nodes[29][1]==0) {
        for(k=1;k<9;k++) {
            for (i = 0; i < 49; i++) {
                if (nodes [edges[i][0] - 1][1] != nodes [edges[i][1] - 1][1] && edges[i][2]==k) {
                    if (nodes [edges[i][0] - 1][1] == 0 && nodes [edges[i][1] - 1][1] != 0) {
                        nodes [edges[i][0] - 1][1] = nodes [edges[i][1] - 1][1];
                        nodes [edges[i][1] - 1][1] = nodes [edges[i][1] - 1][1] + 1;
                        nodes [edges[i][0] - 1][2] = nodes [edges[i][1] - 1][2] + edges[i][2];
                        result[lenth][0] = nodes[edges[i][0] - 1][0] - 1;
                        result[lenth][1] = nodes[edges[i][0] - 1][2];
                        nodes [edges[i][0] - 1][3] = nodes [edges[i][1] - 1][0]-1;
                        lenth++;
                    }
                    else if (nodes [edges[i][0] - 1][1] != 0 && nodes [edges[i][1] - 1][1] == 0) {
                        nodes [edges[i][1] - 1][1] = nodes [edges[i][0] - 1][1];
                        nodes [edges[i][0] - 1][1] = nodes [edges[i][0] - 1][1] + 1;
                        nodes [edges[i][1] - 1][2] = nodes [edges[i][0] - 1][2] + edges[i][2];
                        result[lenth][0] = nodes[edges[i][1] - 1][0] - 1;
                        result[lenth][1] = nodes[edges[i][1] - 1][2];
                        nodes [edges[i][1] - 1][3] = nodes [edges[i][0] - 1][0]-1;
                        lenth++;
                    }
                }
            }
        }
    }
    //sorting valuable nodes
    do{
        pointer=0;
        for (i = 0; i < lenth; i++){
            if (result[i][1] > result[i + 1][1]) {
```

```
            if (result[i][1] > result[i + 1][1]) {
                for (k = 0, temp = 0; k < 2; k++) {
                    temp = result [i + 1][k];
                    result[i + 1][k] = result[i][k];
                    result[i][k] = temp;
                }
                pointer++;
            }
        }
    }while(pointer!=0);
    printf("Used edges: \n");
    for(i=1;i<=lenth;i++) {
        printf("V%d = %d\n",result[i][0],result[i][1]);
    }
    printf("the shortest way is: \n");
    print_the_shortest_way(nodes, node: 29);

    return 0;
}
```

```
Used edges:
V6 = 3
V1 = 6
V12 = 10
V7 = 11
V13 = 11
V8 = 12
V2 = 13
V19 = 13
V18 = 13
V3 = 14
V14 = 14
V24 = 14
V4 = 15
V9 = 15
V5 = 16
V10 = 17
V15 = 17
V21 = 20
V25 = 20
V16 = 21
V11 = 21
V26 = 22
V22 = 23
V28 = 24
V20 = 25
V17 = 28
V27 = 28
V29 = 30
V23 = 31
the shortest way is:
V1=0 V7=3 V13=10 V14=11 V15=14 V16=17 V22=20 V23=23 V29=24 V30=30
```

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

$$x\bar{z} \vee xy \vee yz = \text{yz}\wedge\text{x!z}$$