МІНІСТЕРСТВО ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСЕТЕТ
«ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра штучного інтелекту

Лабораторна робота №4

З дисципліни

«Дискретна математика»

Виконав:

Студент групи КН-115

Курило Валентин

Викладач:

Мельникова Н.І.

Львів-2019

Тема: Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Пріма-Краскалаю

Мета роботи: набуття практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.

Варіант – 14.

Завдання № 1.

Розв'язати на графах наступні задачі:

1. Виконати наступні операції над графами:

1) знайти доповнення до першого графу.

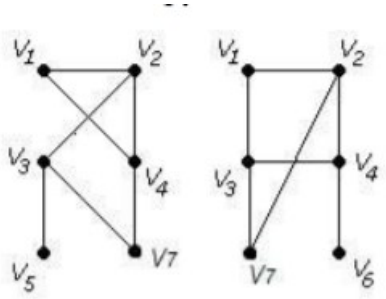
2) об’єднання графів.

3) кільцеву суму G1 та G2 (G1+G2).

4) розщепити вершину у другому графі.

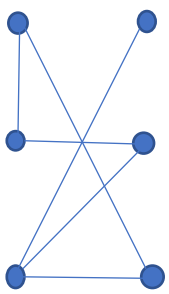
5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A).

6) добуток графів.

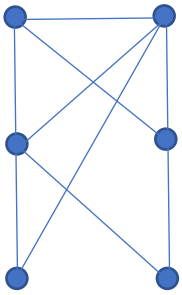


Розв’язання:

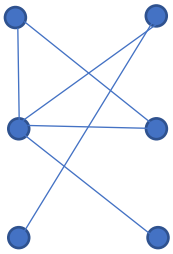
1) знайти доповнення до першого графу:



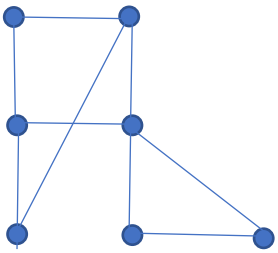
2) об’єднання графів:



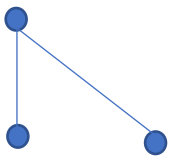
3) кільцеву суму G1 та G2 (G1+G2):



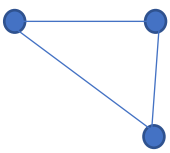
4) розщепити вершину у другому графі:



5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A):

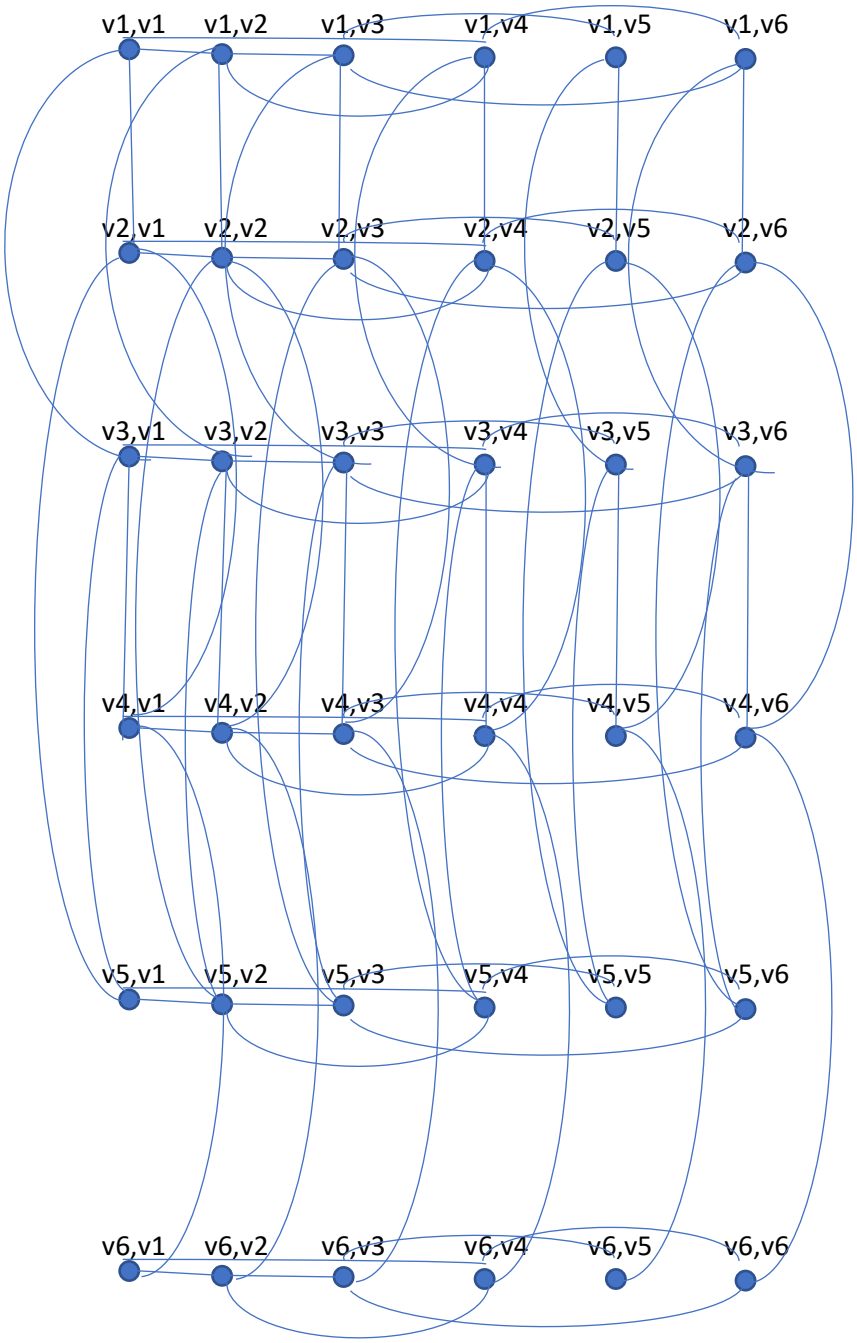


- Виділенний підгпаф з вершин v3,v5,v6.

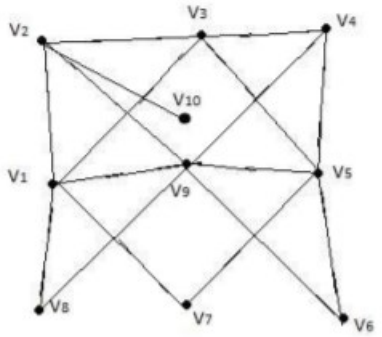


- Стянутий граф з вершинами v1,v2,v4.

6) добуток графів:



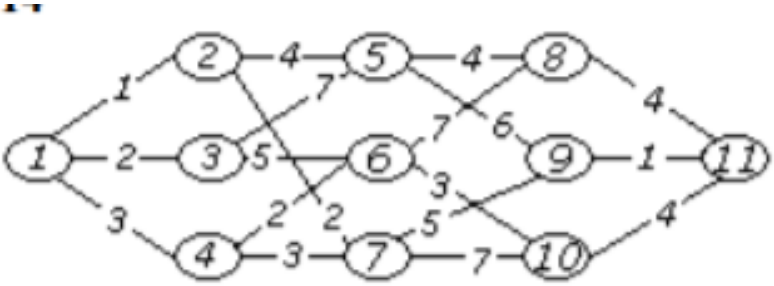
2. Знайти таблицю суміжності та діаметр графа.



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	1	1	0	0	0	1	1	1	0
V2	1	0	1	0	0	0	0	0	1	1
V3	1	1	0	1	1	0	0	0	0	0
V4	0	0	1	0	1	0	0	0	1	0
V5	0	0	1	1	0	1	1	0	1	0
V6	0	0	0	0	1	0	0	0	1	0
V7	1	0	0	0	1	0	0	0	0	0
V8	1	0	0	0	0	0	0	0	1	0
V9	1	1	0	1	1	1	0	1	0	0
V10	0	1	0	0	0	0	0	0	0	0

Діаметр: 3;

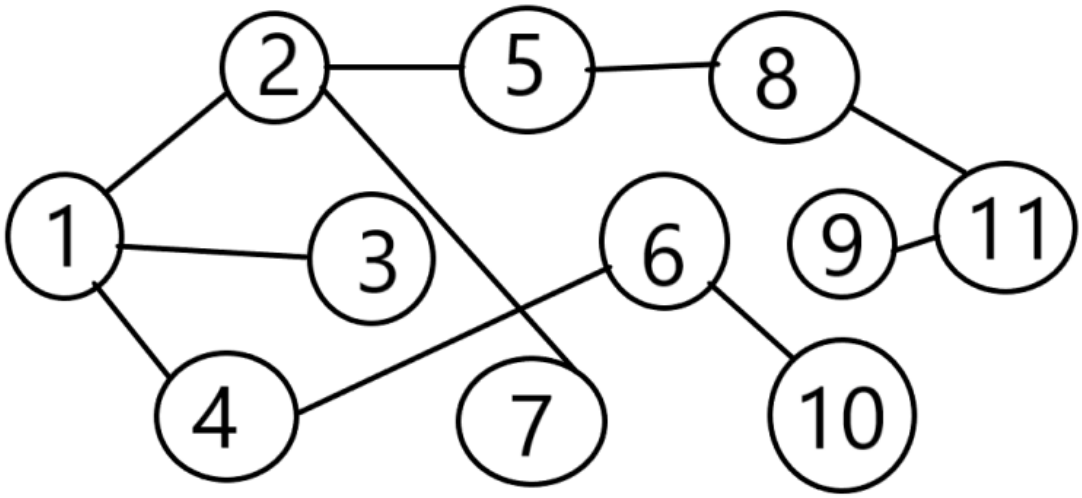
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



Метод Краскала:

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

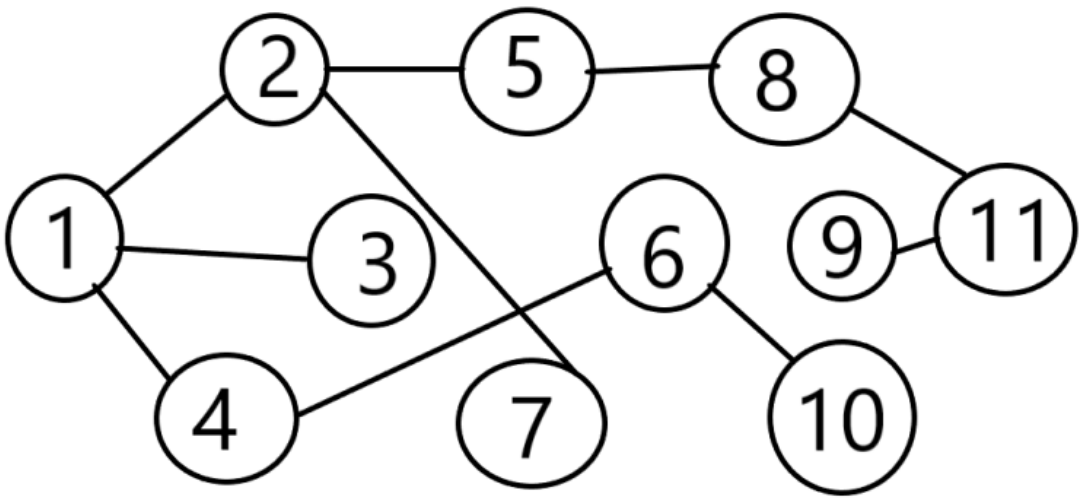
R = ({9,11},{1,2},{1,3},{4,6},{2,7},{1,4},{6,10},{2,5},{5,8},{8,11})



Метод Прима:

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

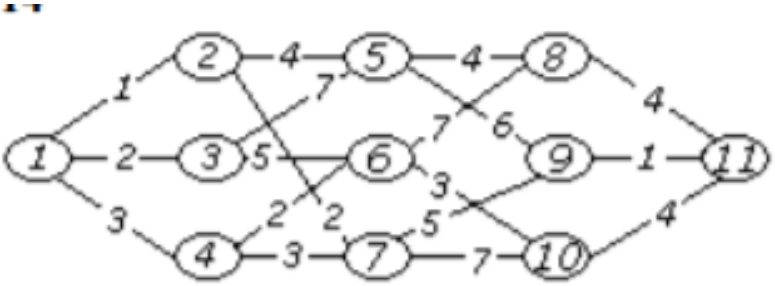
R = ({1,2},{1,3},{2,7},{1,4},{4,6},{6,10},{2,5},{5,8},{8,11},{11,9})



Завдання №2.

Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги згідно свого варіанту.

За алгоритмом Краскала знайти мінімальне остове дерево графа. Етапи розв'язання задачі виводити на екран. Протестувати розроблену програму на наступному графі:



Програма:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
using namespace std;
struct Edge
{
    int src, dest, weight;
};
struct Graph
{
        int V, E;
        struct Edge* edge;
};
struct Graph* createGraph(int V, int E)
{
    struct Graph* graph = (struct Graph*) malloc(sizeof(struct Graph));
    graph->V = V;
    graph->E = E;
    graph->edge = (struct Edge*) malloc(graph->E * sizeof(struct Edge));
    return graph;
}
struct subset
{
        int parent;
        int rank;
};
int find(struct subset subsets[], int i)
{
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);

    return subsets[i].parent;
}
```

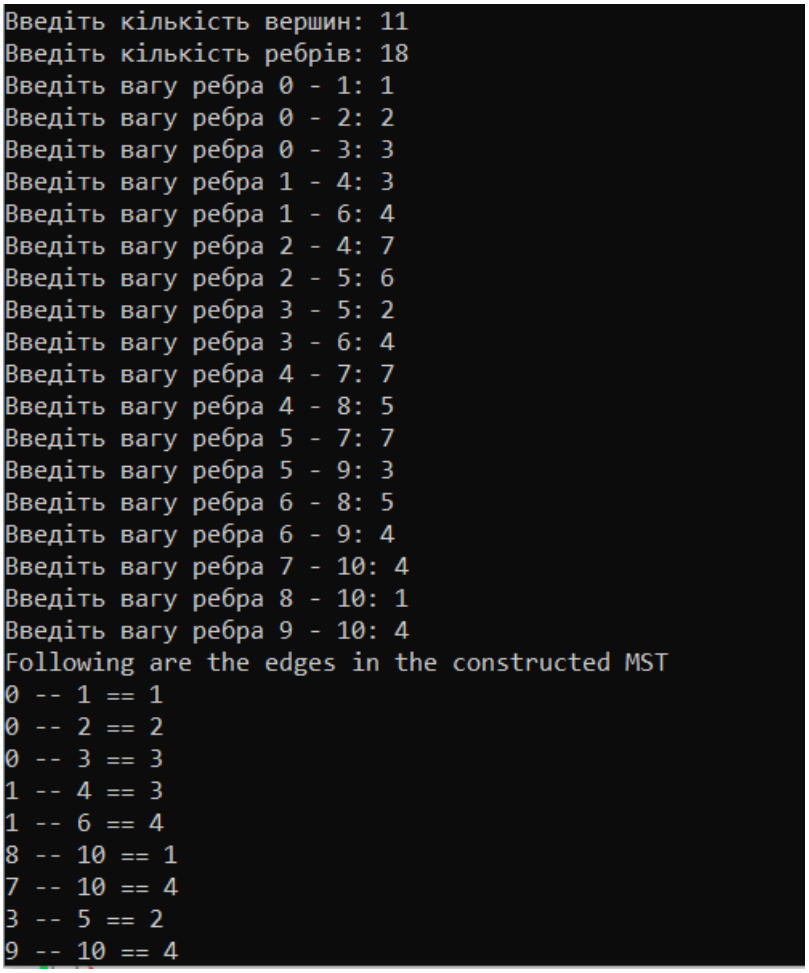
```
void Union(struct subset subsets[], int x, int y)
{
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);
    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else
    {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}
int myComp(const void* a, const void* b)
{
    struct Edge* a1 = (struct Edge*) a;
    struct Edge* b1 = (struct Edge*) b;
    return a1->weight > b1->weight;
}
void KruskalMST(struct Graph* graph)
{
    int V = graph->V;
    struct Edge result[V];
    int e = 0;
    int i = 0;
    qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);
    struct subset *subsets = (struct subset*) malloc(V * sizeof(struct subset));
    for (int v = 0; v < V; ++v)
    {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }
```

```
    while (e < V - 1)
    {
        struct Edge next_edge = graph->edge[i++];
        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);
        if (x != y)
        {
            result[e++] = next_edge;
            Union(subsets, x, y);
        }
    }
    cout<<"Following are the edges in the constructed MST\n";
    for (i = 0; i < e; ++i)
        printf("%d -- %d == %d\n", result[i].src, result[i].dest,
                result[i].weight);
    return;
}
int main()
{
    setlocale(LC_CTYPE, "ukr");
    int x1, x2;
    cout << "Введіть кількість вершин: ";
    cin >> x1;
    cout << "Введіть кількість ребрів: ";
    cin >> x2;
    int V = x1;
    int E = x2;
    int a1, a2,a3, b1,b2,c1,c2, q1,q2, e1,e2, w1, w2, s1,s2,d1,t1,y1;
    struct Graph* graph = createGraph(V, E);
    graph->edge[0].src = 0;
    graph->edge[0].dest = 1;
    cout << "Введіть вагу ребра 0 - 1: ";
    cin >> a1;
    graph->edge[0].weight = a1;

    graph->edge[1].src = 0;
    graph->edge[1].dest = 2;
    cout << "Введіть вагу ребра 0 - 2: ";
    cin >> a2;
    graph->edge[1].weight = a2;
```

```
graph->edge[0].weight = a1;

graph->edge[1].src = 0;
graph->edge[1].dest = 2;
cout << "Введіть вагу ребра 0 - 2: ";
cin >> a2;
graph->edge[1].weight = a2;

graph->edge[2].src = 0;
graph->edge[2].dest = 3;
cout << "Введіть вагу ребра 0 - 3: ";
cin >> a3;
graph->edge[2].weight = a3;
////////////////////////////////
graph->edge[3].src = 1;
graph->edge[3].dest = 4;
cout << "Введіть вагу ребра 1 - 4: ";
cin >> b1;
graph->edge[3].weight = b1;

graph->edge[4].src = 1;
graph->edge[4].dest = 6;
cout << "Введіть вагу ребра 1 - 6: ";
cin >> b2;
graph->edge[4].weight = b2;
////////////////////////////////
graph->edge[5].src = 2;
graph->edge[5].dest = 4;
cout << "Введіть вагу ребра 2 - 4: ";
cin >> c1;
graph->edge[5].weight = c1;

graph->edge[6].src = 2;
graph->edge[6].dest =5;
cout << "Введіть вагу ребра 2 - 5: ";
cin >> c2;
graph->edge[6].weight = c2;
////////////////////////////////////
graph->edge[7].src = 3;
graph->edge[7].dest = 5;
cout << "Введіть вагу ребра 3 - 5: ";
cin >> g1;
```

```
graph->edge[11].src = 5;
graph->edge[11].dest = 7;
cout << "Введіть вагу ребра 5 - 7: ";
cin >> w1;
graph->edge[11].weight = w1;

graph->edge[12].src = 5;
graph->edge[12].dest = 9;
cout << "Введіть вагу ребра 5 - 9: ";
cin >> w2;
graph->edge[12].weight = w2;
////////////////////////////////////////
graph->edge[13].src = 6;
graph->edge[13].dest = 8;
cout << "Введіть вагу ребра 6 - 8: ";
cin >> s1;
graph->edge[13].weight = s1;

graph->edge[14].src = 6;
graph->edge[14].dest = 9;
cout << "Введіть вагу ребра 6 - 9: ";
cin >> s1;
graph->edge[14].weight = s2;
////////////////////////////////////
graph->edge[15].src = 7;
graph->edge[15].dest = 10;
cout << "Введіть вагу ребра 7 - 10: ";
cin >> d1;
graph->edge[15].weight = d1;
////////////////////////////////////
graph->edge[16].src = 8;
graph->edge[16].dest = 10;
cout << "Введіть вагу ребра 8 - 10: ";
cin >> t1;
graph->edge[16].weight = t1;
////////////////////////////////////
graph->edge[17].src = 9;
graph->edge[17].dest = 10;
cout << "Введіть вагу ребра 9 - 10: ";
cin >> y1;
graph->edge[17].weight = y1;
KruskalMST(graph);
return 0;
```

Результат:


Введіть кількість вершин: 11
Введіть кількість ребрів: 18
Введіть вагу ребра 0 - 1: 1
Введіть вагу ребра 0 - 2: 2
Введіть вагу ребра 0 - 3: 3
Введіть вагу ребра 1 - 4: 3
Введіть вагу ребра 1 - 6: 4
Введіть вагу ребра 2 - 4: 7
Введіть вагу ребра 2 - 5: 6
Введіть вагу ребра 3 - 5: 2
Введіть вагу ребра 3 - 6: 4
Введіть вагу ребра 4 - 7: 7
Введіть вагу ребра 4 - 8: 5
Введіть вагу ребра 5 - 7: 7
Введіть вагу ребра 5 - 9: 3
Введіть вагу ребра 6 - 8: 5
Введіть вагу ребра 6 - 9: 4
Введіть вагу ребра 7 - 10: 4
Введіть вагу ребра 8 - 10: 1
Введіть вагу ребра 9 - 10: 4
Following are the edges in the constructed MST
0 -- 1 == 1
0 -- 2 == 2
0 -- 3 == 3
1 -- 4 == 3
1 -- 6 == 4
8 -- 10 == 1
7 -- 10 == 4
3 -- 5 == 2
9 -- 10 == 4