

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

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни

“Дискретна математика”

Виконав:

студент групи КН-109

Коржов Володимир

Викладач:

Мельникова Н.І.

Львів — 2018

Основні операції над графами.

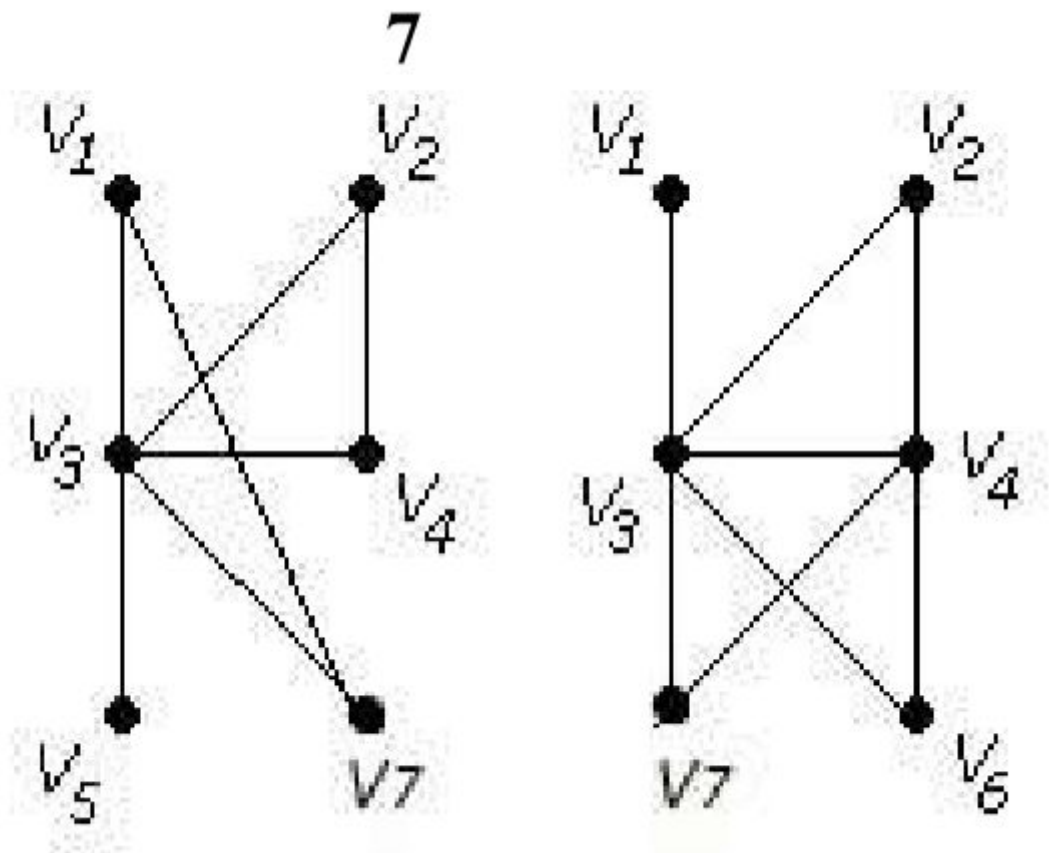
Знаходження остова мінімальної ваги за алгоритмом Прима-Краскала

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

Варіант №7

Завдання № 1

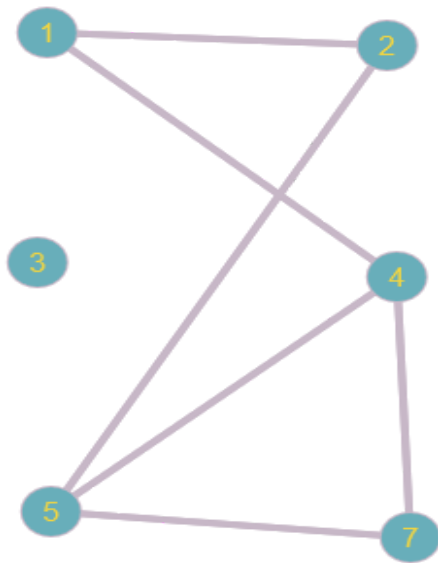
1. Виконати наступні операції над графами:



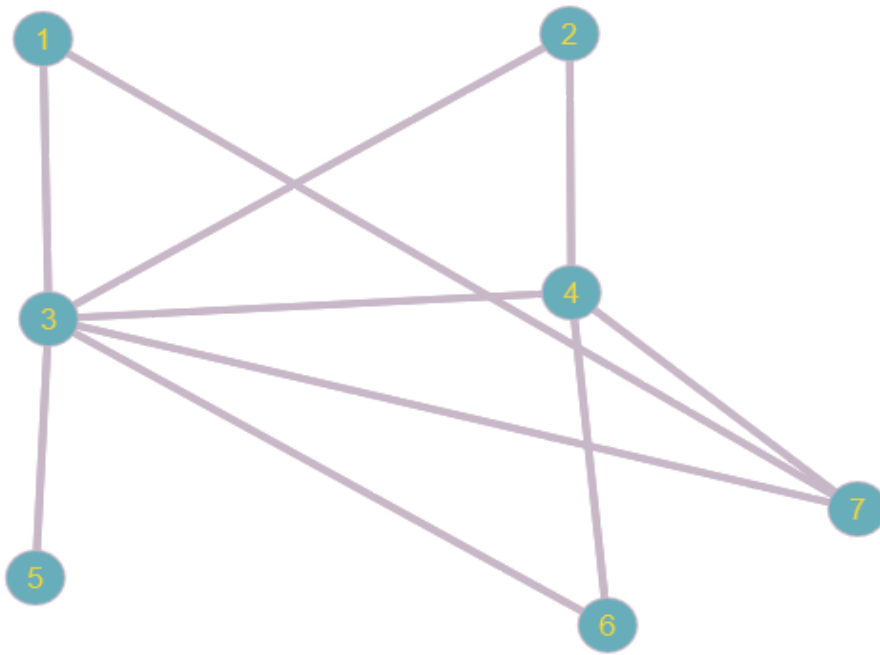
- 1) знайти доповнення до першого графу,
- 2) об'єднання графів,
- 3) кільцеву суму G_1 та G_2 ($G_1 + G_2$),

- 4) розщепити вершину у другому графі,
- 5) виділити підграф A , що складається з 3-х вершин в G_1 і знайти стягнення A в G_1 ($G_1 \setminus A$),
- 6) добуток графів.

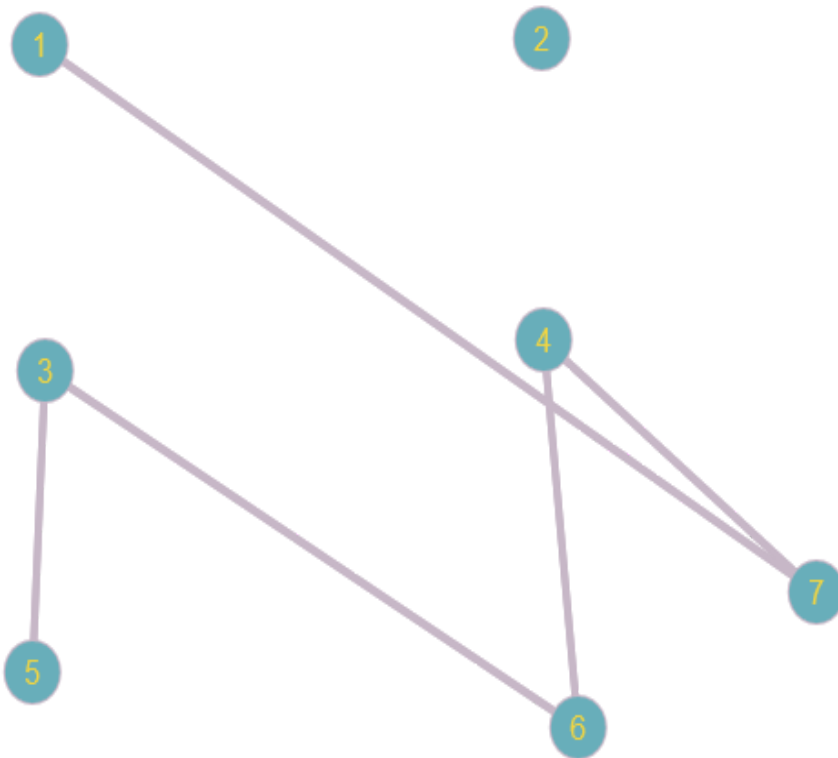
1)



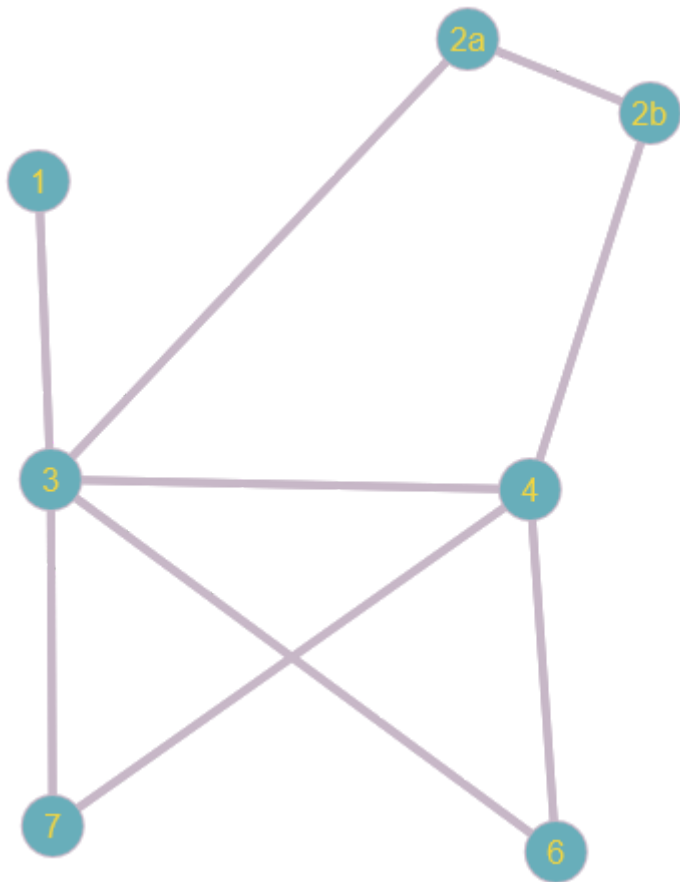
2)



3)

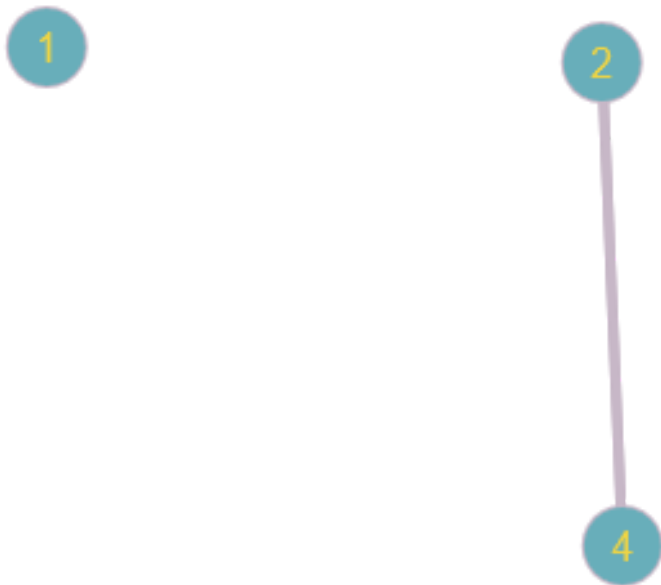


4)



5)

$A =$



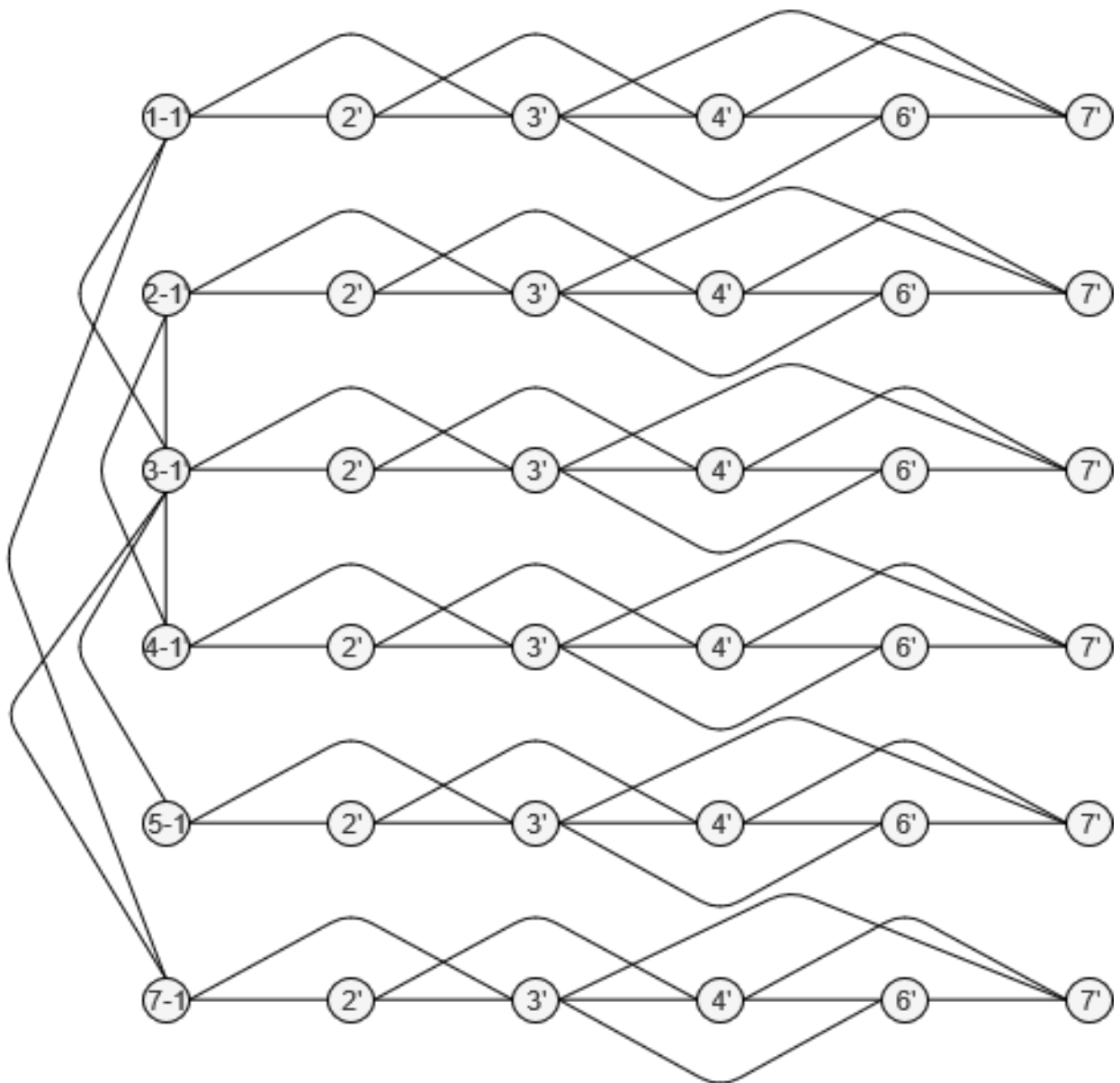
$G1A =$

1

2

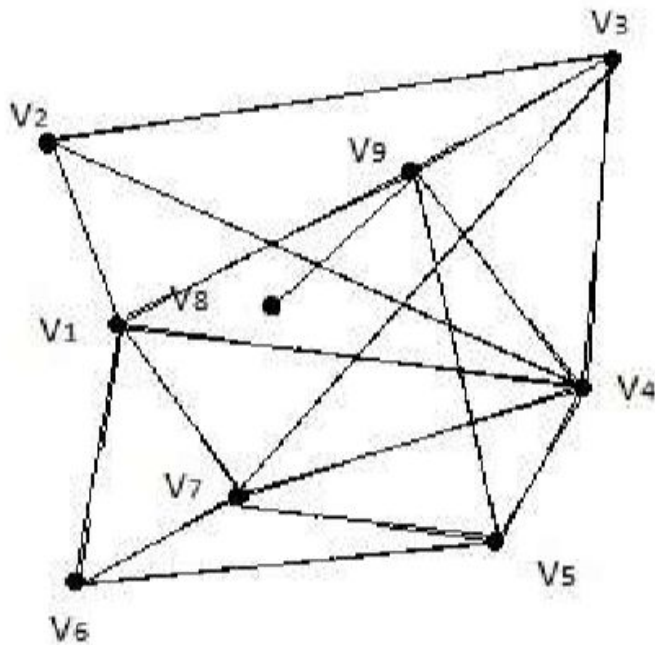
4

6)



2. Знайти таблицю суміжності та діаметр графа.

7

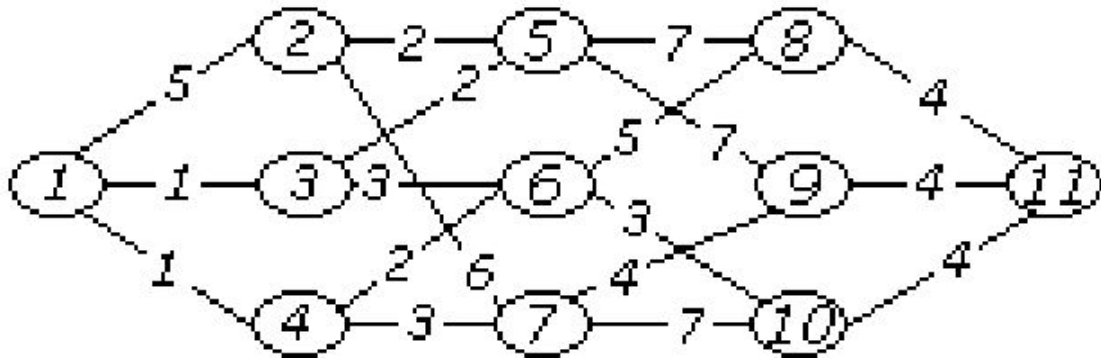


	v1	v2	v3	v4	v5	v6	v7	v8	v9
v1	0	1	0	1	0	1	1	0	1
v2	1	0	1	1	0	0	0	0	0
v3	0	1	0	1	0	0	1	0	1
v4	1	1	1	0	1	0	1	0	1
v5	0	0	0	1	0	1	1	0	1
v6	1	0	0	0	1	0	1	0	0
v7	1	0	1	1	1	1	0	0	0
v8	0	0	0	0	0	0	0	0	1
v9	1	0	1	1	1	0	0	1	0

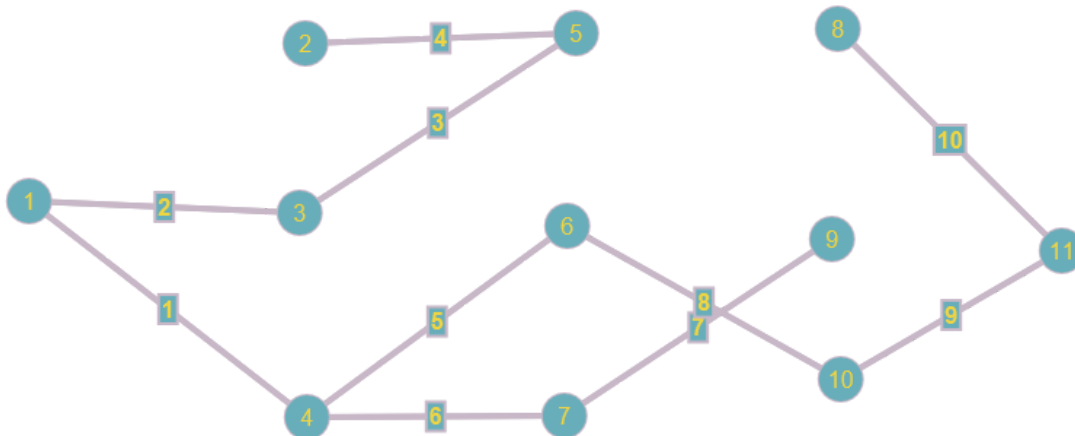
$$D = (v1 - v8) = (v6 - v8) = (v7 - v8) = 3;$$

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

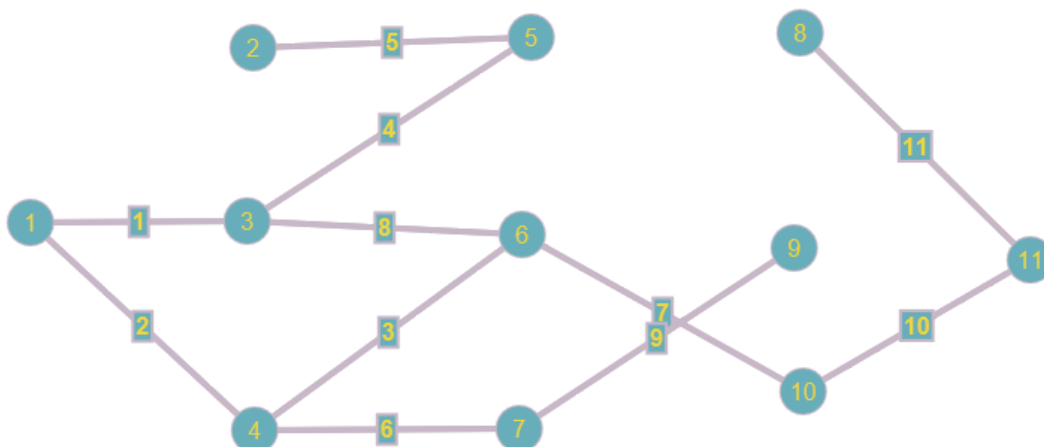
7



Метод Краскала:



Метод Прима:



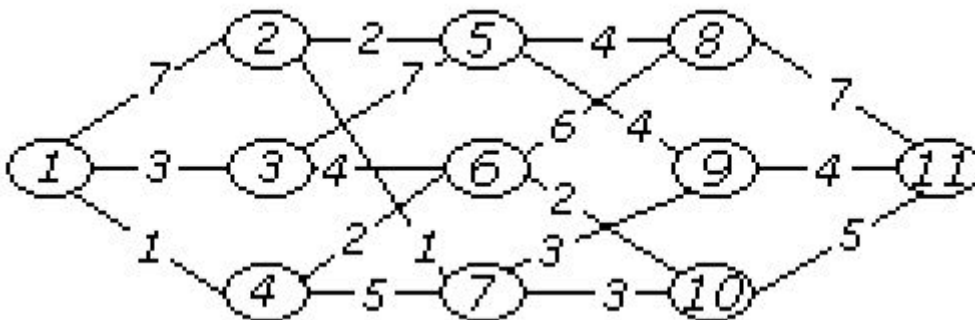
Примітка: на ребрах вказана не їхня вага, а послідовність побудови остового дерева.

Завдання №2.

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

Варіант №7

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



Код програми:

```
#include<stdio.h>
typedef struct
{
    int first_vertex;
    int second_vertex;
    int weight;
}rib;
int el_in_array(int arr[],int size,int element)
{
    for(int i = 0; i < size; i++)
    {
        if(arr[i] == element)
        {
            return 1;
        }
    }
}
```

```

        return 0;
    }
    rib min_weight(rib a[], int length)
    {
        rib min;
        min = a[0];
        for(int i = 0; i < length; i++)
        {
            if(a[i].weight < min.weight)
            {
                min = a[i];
            }
        }
        return min;
    }
    int ver_in_array(int arr[], int size, int vertex )
    {
        for(int i = 0; i < size; i++)
        {
            if(arr[i] == vertex)
            {
                return 1;
            }
        }
    }
    return 0;
}
int main()
{
    int i, num, counter = 0, ver_count = 1, a_rcount = 0;;
    printf("Enter the number of ribs for your tree: ");
    scanf("%d", &num);
    if(num > 20 || num < 0)
    {
        printf("Enter a number greater than 0 and less than 20");
        scanf("%d", &num);
    }
}

```

```

    }
    rib ribs[num];
    int vertexes[2*num], ost_vertexes[11], active_vertexes[11];
    for(i = 0; i < num; i++)
    {
        printf("Enter the first vertex for rib #%%d: \n", i);
        scanf("%d", &ribs[i].first_vertex);
        printf("Enter the second vertex for rib # %%d \n", i);
        scanf("%d", &ribs[i].second_vertex);
        printf("Enter the weight of rib #%%d:\n", i);
        scanf("%d", &ribs[i].weight);
    }
    printf("\n");
    for(i = 0; i < num; i++)
    {
        printf("Rib %%d-%%d has the weight of %%d\n", ribs[i].first_vertex,
ribs[i].second_vertex, ribs[i].weight);
    }
    printf("\n");
    for(i = 0; i < num; i++)
    {
        vertexes[i*2] = ribs[i].first_vertex;
        vertexes[i*2+1] = ribs[i].second_vertex;
    }
    printf("\n");
    for(i = 0; i < num*2; i++)
    {
        if(!el_in_array(ost_vertexes, counter, vertexes[i]))
        {
            ost_vertexes[counter] = vertexes[i];
            counter++;
        }
    }
    printf("\n");
    active_vertexes[0] = ost_vertexes[0];

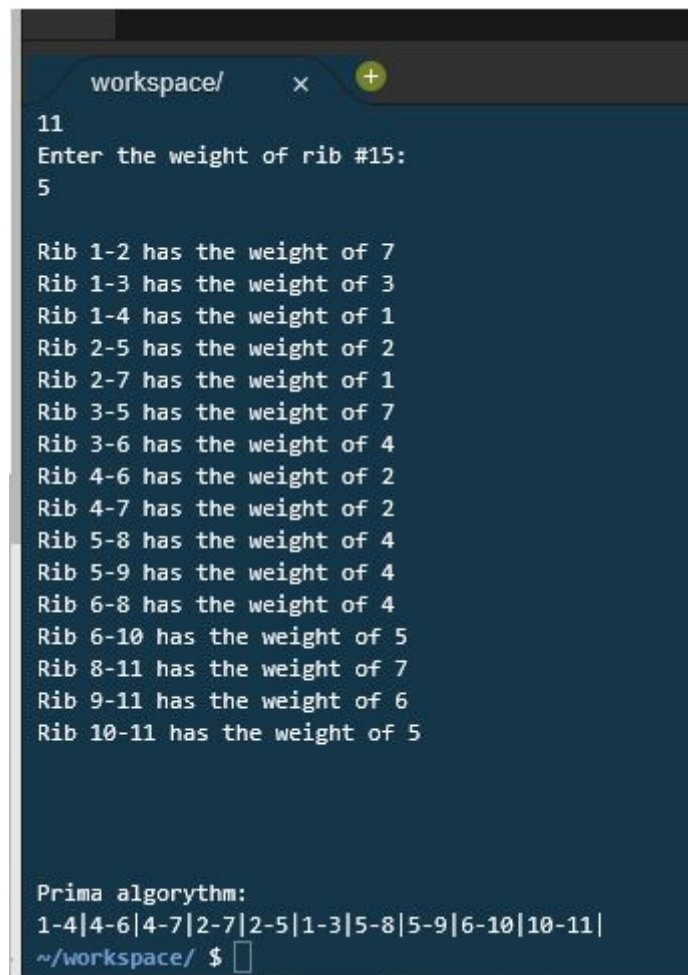
```

```

printf("Prima algorythm:\n");
do
{
rib ribs_selection[11];
int rcount = 0;
for(i = 0; i < num; i++)
{
if( (ver_in_array(active_vertexes, ver_count, ribs[i].first_vertex) +
ver_in_array(active_vertexes, ver_count, ribs[i].second_vertex)) % 2)
{
ribs_selection[rcount] = ribs[i];
rcount++;
}
}
rib min = min_weight(ribs_selection, rcount);
a_rcount++;
if(ver_in_array(active_vertexes, ver_count, min.first_vertex))
{
active_vertexes[ver_count] = min.second_vertex;
}
else
{
active_vertexes[ver_count] = min.first_vertex;
}
printf("%d-%d", min.first_vertex,min.second_vertex);
ver_count++;
}while(ver_count != 11);
printf("\n");
return 0;
}

```

Скріншот роботи програми:



```
workspace/ x +
11
Enter the weight of rib #15:
5

Rib 1-2 has the weight of 7
Rib 1-3 has the weight of 3
Rib 1-4 has the weight of 1
Rib 2-5 has the weight of 2
Rib 2-7 has the weight of 1
Rib 3-5 has the weight of 7
Rib 3-6 has the weight of 4
Rib 4-6 has the weight of 2
Rib 4-7 has the weight of 2
Rib 5-8 has the weight of 4
Rib 5-9 has the weight of 4
Rib 6-8 has the weight of 4
Rib 6-10 has the weight of 5
Rib 8-11 has the weight of 7
Rib 9-11 has the weight of 6
Rib 10-11 has the weight of 5

Prima algorythm:
1-4|4-6|4-7|2-7|2-5|1-3|5-8|5-9|6-10|10-11|
~/workspace/ $
```

Висновок: я набув практичних вмінь та навички з використання алгоритмів Прима і Краскала.