МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра систем штучного інтелекту

Лабораторна робота №

З курсу “Дискретна математика”

Виконав:
ст.гр. КН-110
Андрусяк Нестор

Львів – 2018

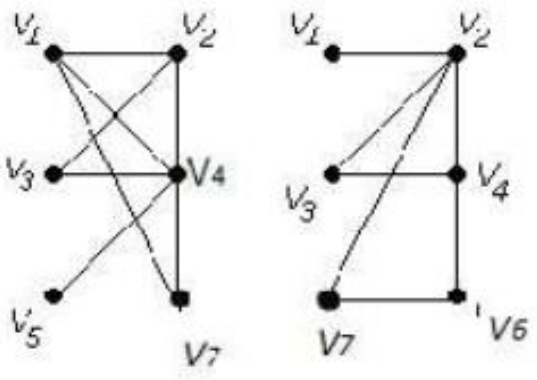
Лабораторна робота № 4.

Тема: Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Пріма-Краскала
Мета роботи: набуття практичних вмінь та навичок з використання алгоритмів Пріма і Краскала

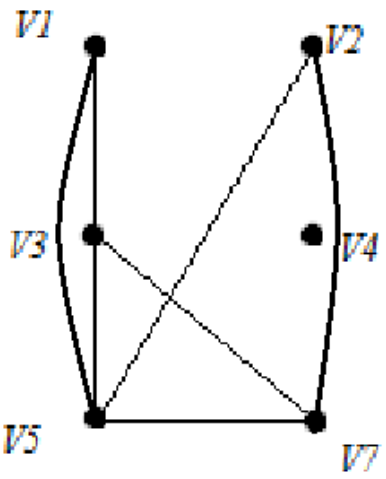
Завдання № 1. Розв'язати на графах наступні задачі:
1. Виконати наступні операції над графами:
1) знайти доповнення до першого графу,
2) об’єднання графів,
3) кільцеву суму G1 та G2 (G1+G2),
4) розщепити вершину у другому графі,
5) виділити підграф А, що складається з 3-х вершин в G1 і знайти стягнення А в G1 (G1\ A)

6) добуток графів.

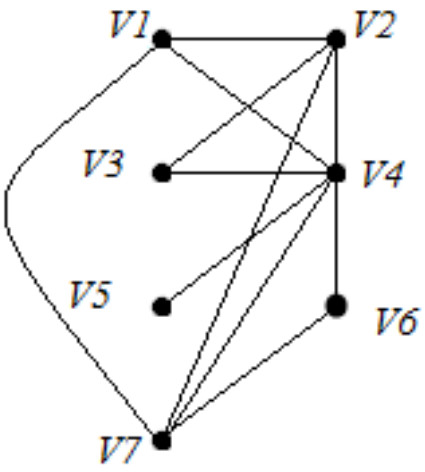
1



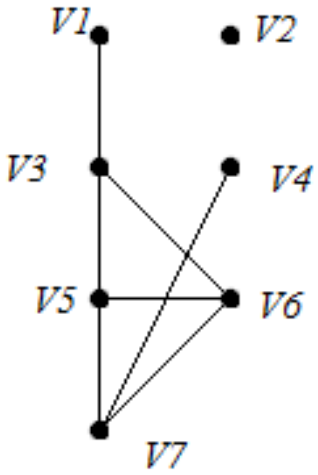
1. Доповнення до першого графу:



2. Об’єднання графів:

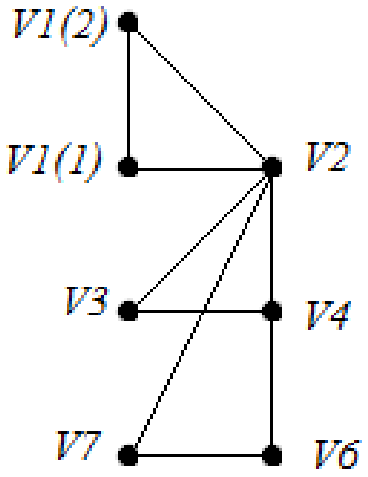


3. Кільцева сума G1 та G2:

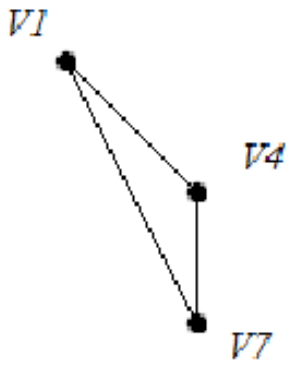


4. Розщепити вершину у другому графі:

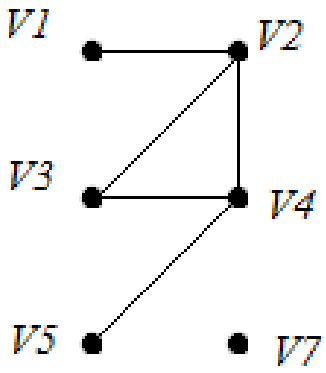
Розщепимо V1:



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

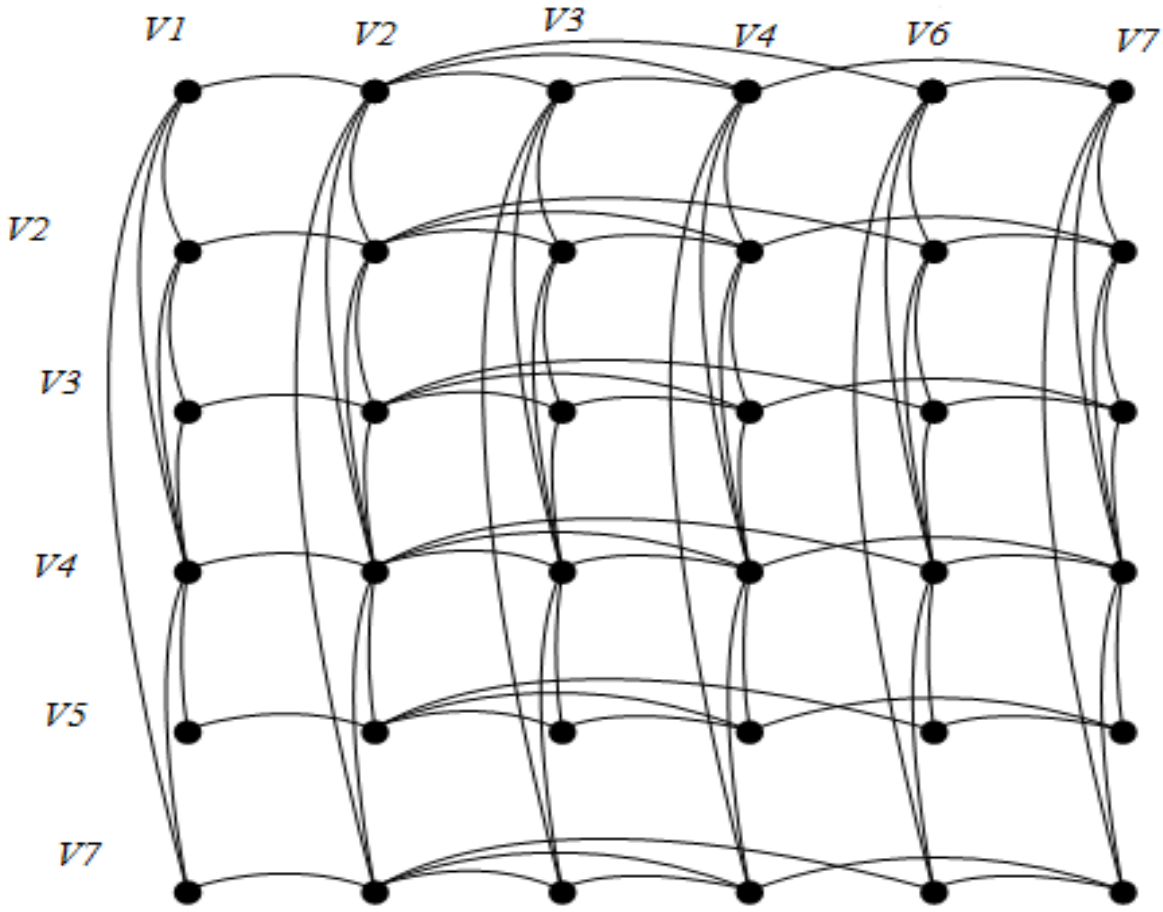


Підграф А

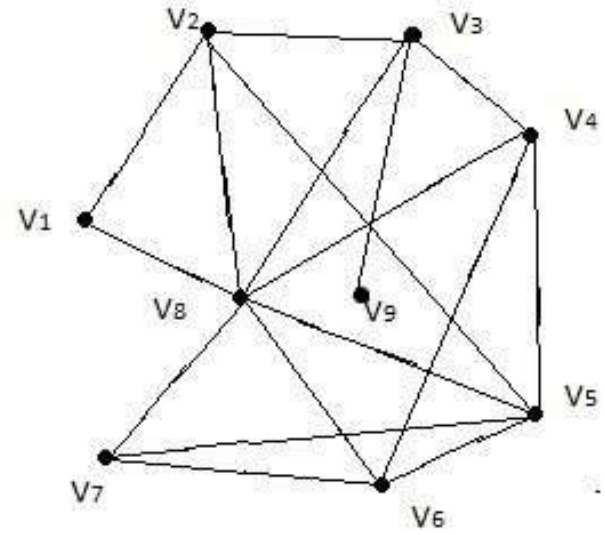


G1\A

6. Добуток



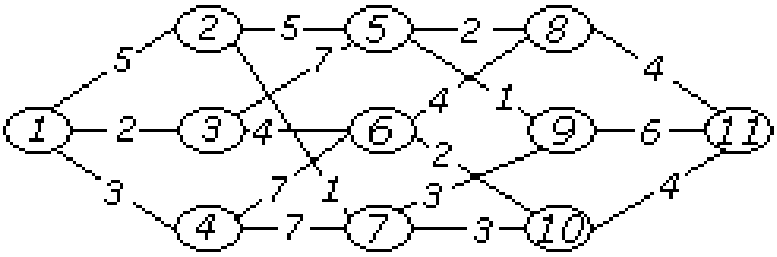
2. Знайти таблицю суміжності та діаметр графа.



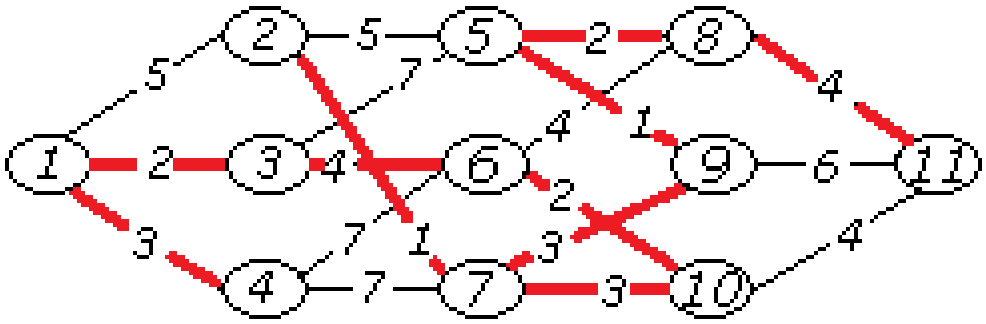
	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	0	0	0	0	0	1	0
V2	1	0	1	0	0	0	0	1	0
V3	0	1	0	1	0	0	0	0	1
V4	0	0	0	0	1	0	0	1	0
V5	0	1	0	1	0	1	1	1	0
V6	0	0	0	1	1	0	1	1	0
V7	0	0	0	0	1	1	0	1	0
V8	1	1	1	1	1	1	1	0	0
V9	0	0	1	0	0	0	0	0	0

Діаметр графа 3.

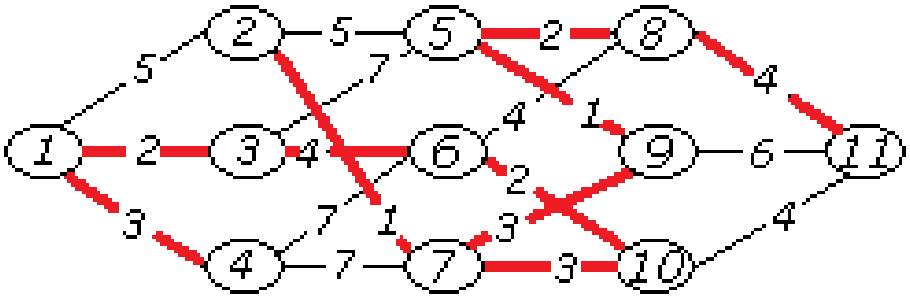
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



1) Прима:



2) Краскала:



Кроки:

1. 2-7, 5-9

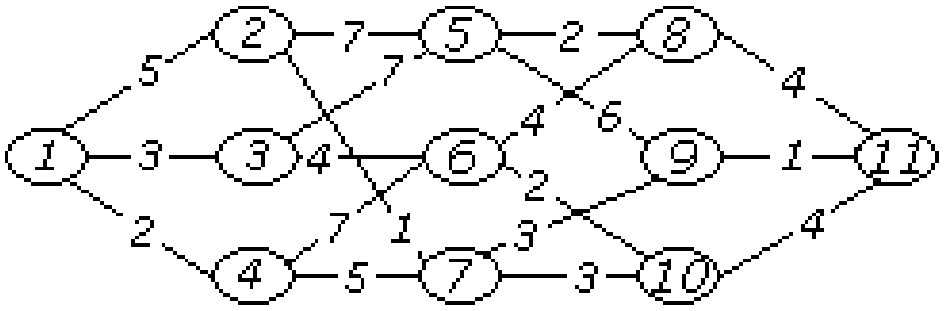
2. 1-3, 5-8

3. 1-4, 7-10

4. 3-6, 6-8, 8-10

Завдання №2. Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги згідно свого варіанту.

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



Код:

```
#include<stdio.h>
#include<stdlib.h>

#define infinity 9999
#define MAX 20

int G[MAX][MAX],spanning[MAX][MAX],n;

int prims();

int main()
{
   int cost;
   printf("Enter the ammount of nodes:");
   scanf("%d", &n);
```

```
    printf("\nEnter the adjacency matrix:\n");

    for(i = 0; i < n; i++)
        for(j=0;j<n;j++)
            scanf("%d",&G[i][j]);

    cost=prims();
    printf("\nSpanning tree matrix:\n");

    for(i=0;i<n;i++)
    {
        printf("\n");
        for(j=0;j<n;j++)
            printf("%d\t",spanning[i][j]);
    }

    printf("\n\nTotal cost of spanning tree=%d",cost);
    return 0;
}

int prims()
{
    int cost[MAX][MAX];
    int u,v,min_distance,distance[MAX],from[MAX];
    int visited[MAX],no_of_edges,i,min_cost,j;

    //create cost[][] matrix,spanning[][]
```

```
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
   {
      if(G[i][j]==0)
         cost[i][j]=infinity;
      else
         cost[i][j]=G[i][j];
         spanning[i][j]=0;
   }

//initialise visited[],distance[] and from[]
distance[0]=0;
visited[0]=1;

for(i=1;i<n;i++)
{
   distance[i]=cost[0][i];
   from[i]=0;
   visited[i]=0;
}

min_cost=0;       //cost of spanning tree
no_of_edges=n-1;       //no. of edges to be added

while(no_of_edges>0)
{
   //find the vertex at minimum distance from the tree
   min_distance=infinity;
```

```
    for(i=1;i<n;i++)
        if(visited[i]==0&&distance[i]<min_distance)
        {
            v=i;
            min_distance=distance[i];
        }

    u=from[v];

    //insert the edge in spanning tree
    spanning[u][v]=distance[v];
    spanning[v][u]=distance[v];
    no_of_edges--;
    visited[v]=1;

    //updated the distance[] array
    for(i=1;i<n;i++)
        if(visited[i]==0&&cost[i][v]<distance[i])
        {
            distance[i]=cost[i][v];
            from[i]=v;
        }

    min_cost=min_cost+cost[u][v];
  }

  return(min_cost);
}
```