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

Лабораторна робота №4

З курсу “Дискретна математика ”

Виконав:
ст.гр. КН-110
Бохонко Андрій
Викладач:
Мельникова Н.І.

Львів – 2018

Лабораторна робота № 4.

В.3

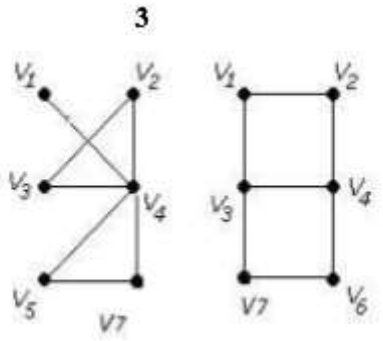
Тема: Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Пріма-Краскала

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

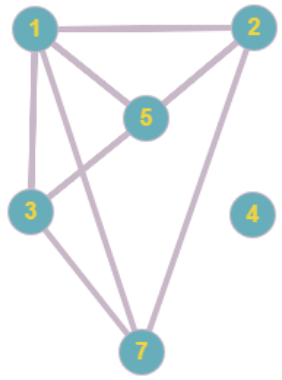
Розв'язки:

1.

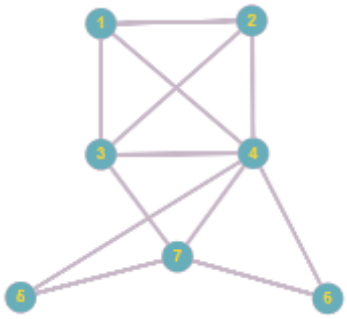
Виконати наступні операції над графами:



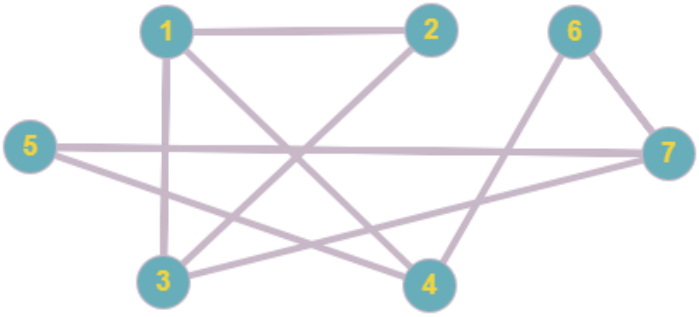
1) знайти доповнення до першого графу,



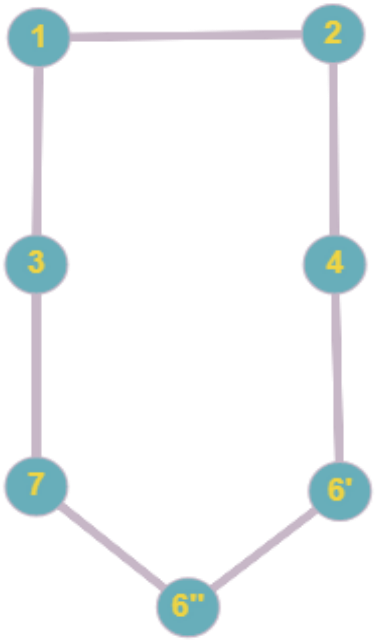
2) об'єднання графів,



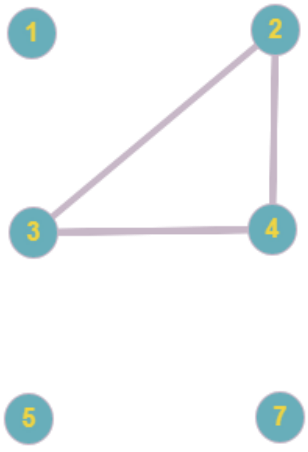
3) кільцеву суму *G1* та *G2* (*G1+G2*),

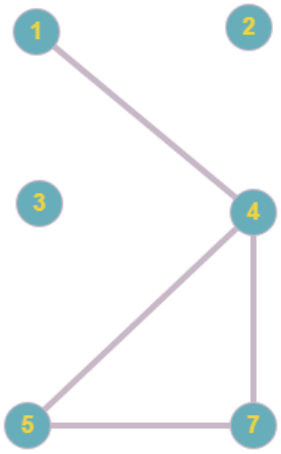


4) розщепити вершину у другому графі (розчепив вершину 6),

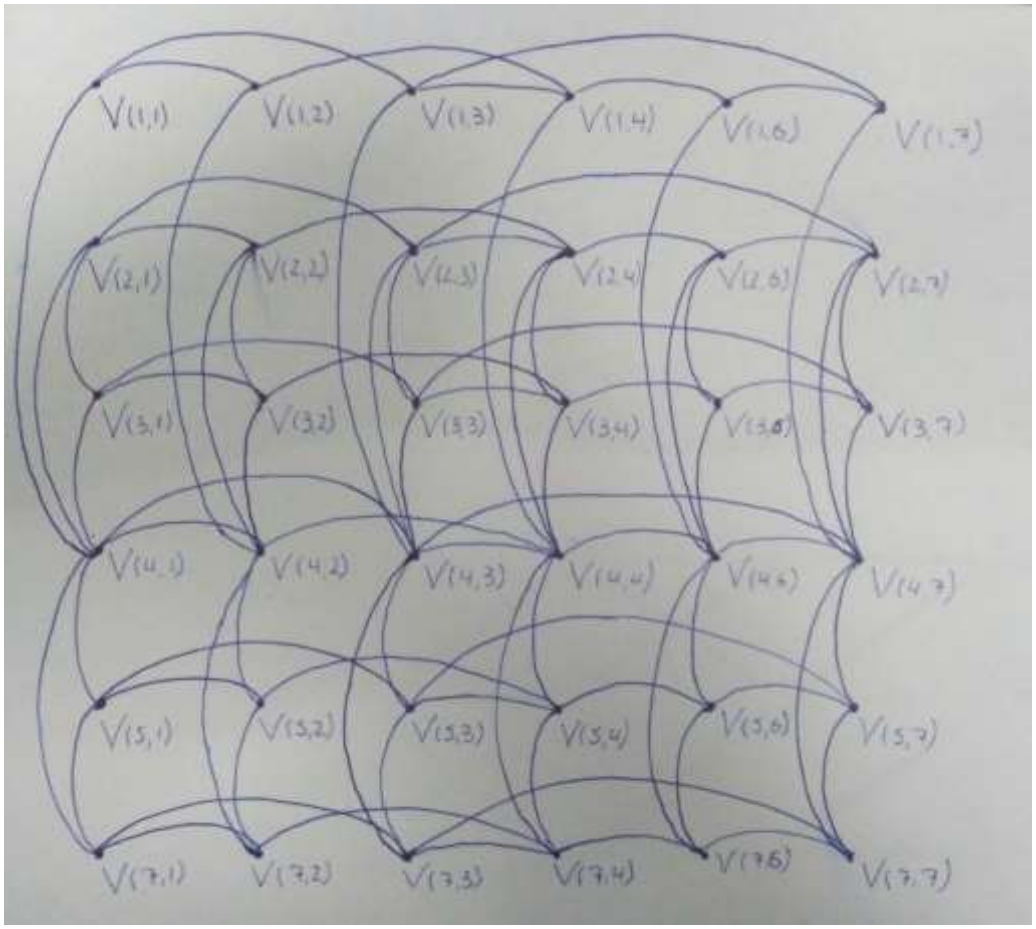


5) виділити підграф *A*, що складається з 3-х вершин в *G1* і знайти стягнення *A* в *G1* (*G1\ A*),

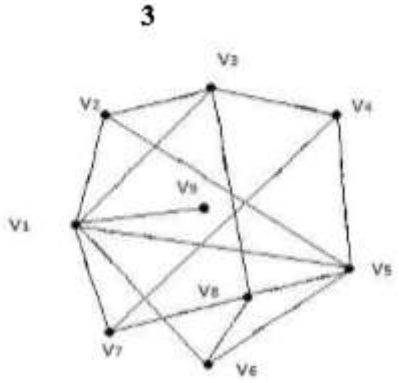




6) добуток графів.



2. Знайти таблицю суміжності та діаметр графа.

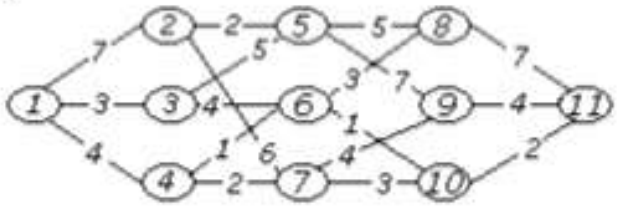


	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	1	0	1	1	1	0	1
V2	1	0	1	0	1	0	0	0	0
V3	1	1	0	1	0	0	0	1	0
V4	0	0	1	0	1	0	1	0	0
V5	1	1	0	1	0	1	1	0	0
V6	1	0	0	0	1	0	0	1	0
V7	1	0	0	1	0	0	0	1	0
V8	0	0	1	0	1	1	1	0	0
V9	1	0	0	0	0	0	0	0	0

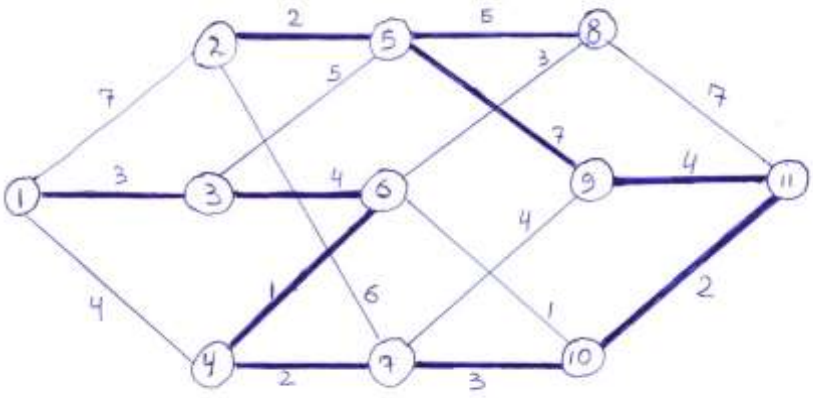
max {1,2,3} = 3 – діаметр графа

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

3

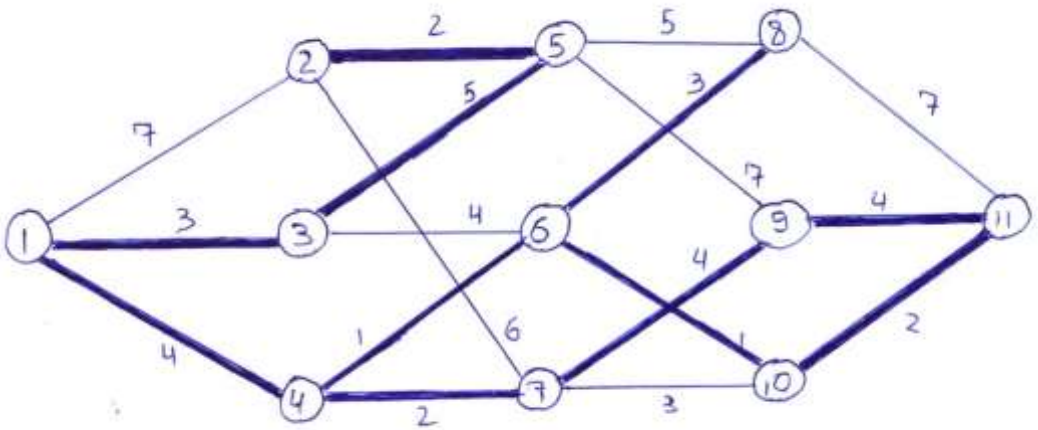


Алгоритм Прима.



Починаємо з вершини 1 і шукаємо ребро з найменшою вагою. Найменша вага (3) до вершини 3. З вершини 3 ребро з найменшою вагою (4) – ребро до вершини 6. З вершини 6 ребро з найменшою вагою (1) – ребро до вершини 4. З вершини 4 ребро з найменшою вагою (2) – ребро до вершини 7. З вершини 7 ребро з найменшою вагою (3) – ребро до вершини 10. З вершини 10 ребро з найменшою вагою (1) веде до вершини 6, але ця вершина вже задіяна тому рухатись до вершини 6 ми не можемо. Вибираємо наступне ребро з найменшою вагою(2) – ребро до вершини 11. З вершини 11 ребро з найменшою вагою (4) – ребро до вершини 9. З вершини 9 ребро з найменшою вагою (4) – ребро, яке веде до вершини 7. Ця вершина вже задіяна, тому ми вибираємо ребро з вагою (7) до вершини 5. З вершини 5 ребро з найменшою вагою (2) – ребро до вершини 2. Задіяні всі вершини, крім 8. Отже шукаємо ребро з найменшою вагою до вершини 8. Це ребро з вагою (5) від вершини 5 до вершини 8. Всі вершини задіяні. Отже мінімальне остове дерево за алгоритмом Прима знайдено.

Алгоритм Краскала.



Шукаємо ребра з найменшою вагою. Це ребро 4-6 і 6-10 з вагою (1). Виділяємо їх. Далі ребра з вагою (2) – 2-5, 4-7, 10-11. Виділяємо їх. Далі ребра з вагою (3) – 1-3, 6-8. Виділяємо їх. Також вагу (3) має ребро 7-10, але до мінімального остового дерева ми його не включаємо, бо тоді утвориться цикл 6-4-10. Далі ребра з вагою (4) – 9-11, 1-4, 7-9. Виділяємо їх. Ребро 3-6 не включаємо, бо тоді утвориться цикл 1-3-6-4. Далі ребра з вагою (5) – 3-5. Виділяємо його. Всі вершини задіяні. Отже за алгоритмом Краскала побудовано мінімальне остове дерево.

Код програми

```
 #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 i,j,total_cost;
   printf("Enter no. of vertices:");
   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]);
   for (i=0 ; i<n; i++){
      for (j=0; j<n; j++)
         {
         printf ("%d ",G[i][j]);
         }
      printf ("\n");
      }
   total_cost=prims();
   printf("\nspanning tree matrix:\n");

   for(i=0;i<n;i++)
   {
      printf("\n");
```

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

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

Результат програми

```
Your matrix:
0 7 5 2 0 0 0 0 0 0 0
7 0 0 0 7 0 5 0 0 0 0
5 0 0 0 7 4 0 0 0 0 0
2 0 0 0 0 3 1 0 0 0 0
0 7 7 0 0 0 0 4 4 0 0
0 0 4 3 0 0 0 4 0 2 0
0 5 0 1 0 0 0 0 3 2 0
0 0 0 0 4 4 0 0 0 0 3
0 0 0 0 4 0 3 0 0 0 6
0 0 0 0 0 2 2 0 0 0 1
0 0 0 0 0 0 0 3 6 1 0

spanning tree matrix:

0 0 0 2 0 0 0 0 0 0 0
0 0 0 0 0 0 5 0 0 0 0
0 0 0 0 0 4 0 0 0 0 0
2 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 4 0 0 0
0 0 4 0 0 0 0 0 0 2 0
0 5 0 1 0 0 0 0 3 2 0
0 0 0 0 4 0 0 0 0 0 3
0 0 0 0 0 0 3 0 0 0 0
0 0 0 0 0 2 2 0 0 0 1
0 0 0 0 0 0 0 3 0 1 0

Total cost of spanning tree=27
```

Висновок

В результаті цієї лабораторної роботи я освоїв операції над графами (об'єднання, доповнення, кільцева сума). Навчився шукати мінімальне остове дерево за допомогою алгоритмів Прима і Краскала та реалізовувати ці алгоритми програмно.