МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ

“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 4

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-112

Стаськів Максим

Викладач:

Мельникова Н.І.

Львів – 2019р.

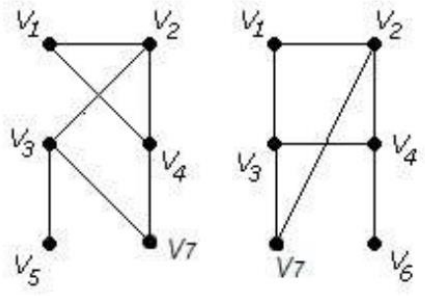
Варіант 14

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

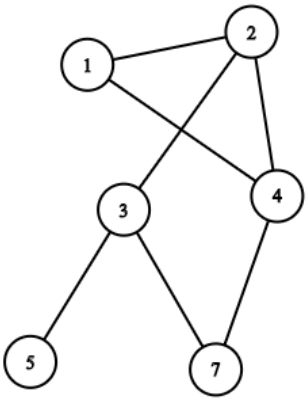
Мета роботи: набуття практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.

Завдання № 1. Розв'язати на графах наступні задачі:

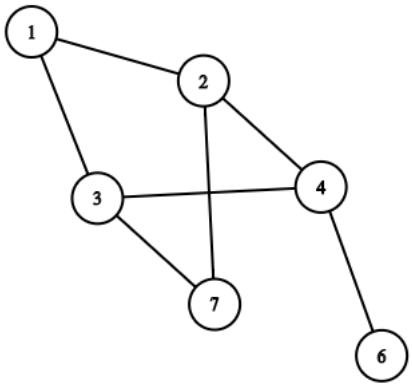
1. Виконати наступні операції над графами:
1) знайти доповнення до першого графу,
2) об’єднання графів,
3) кільцеву суму G1 та G2 (G1+G2),
4) розщепити вершину у другому графі,
5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A),
6) добуток графів.



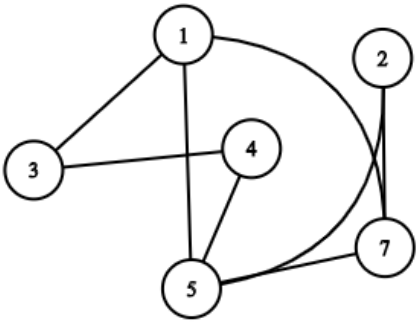
G1:



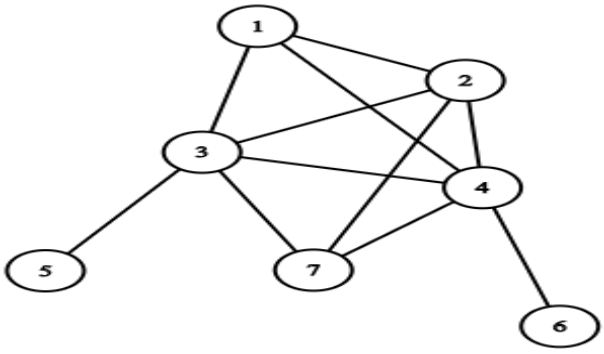
G2:



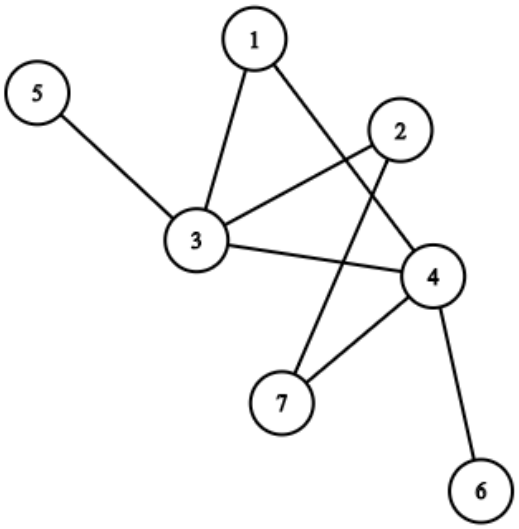
1)



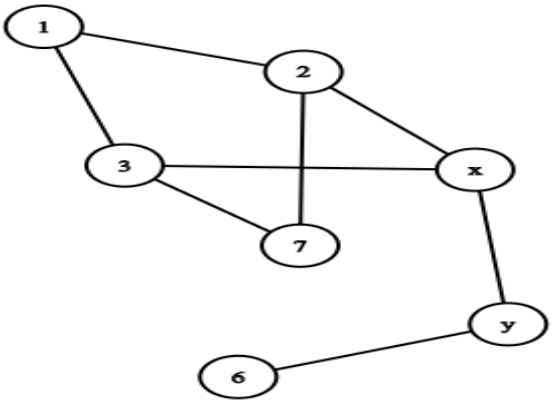
2)



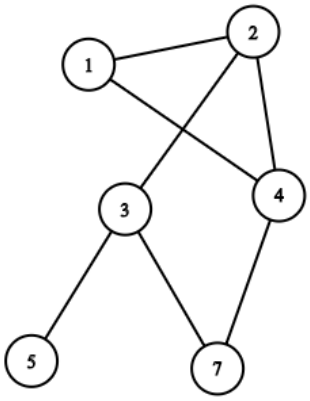
3)

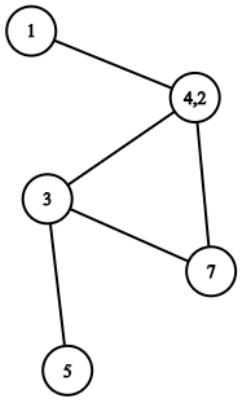


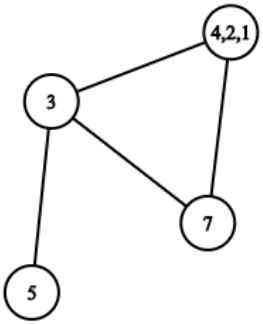
4)

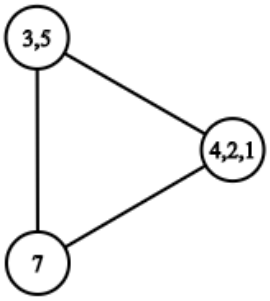


5) G`=(V,E); V(G`)={1,2,4}; V(G1)={1,2,3,4,5,7}

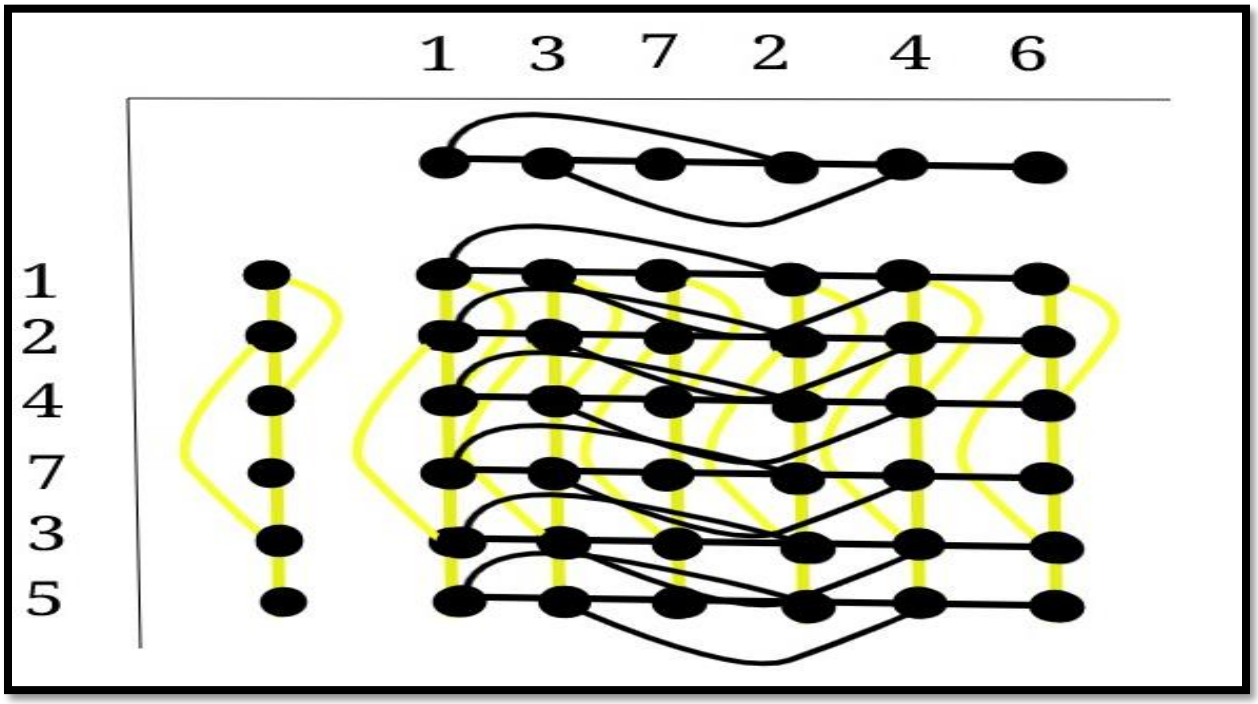




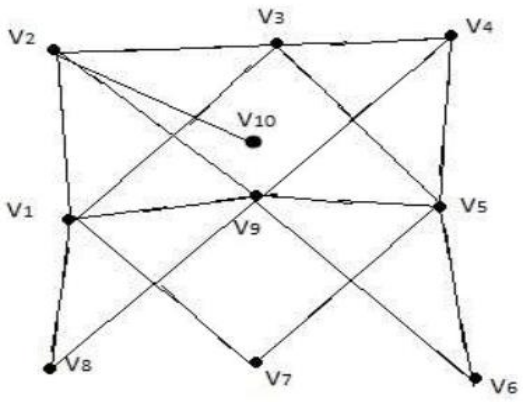




6)



2. Знайти таблицю суміжності та діаметр графа.



Таблиця Суміжності

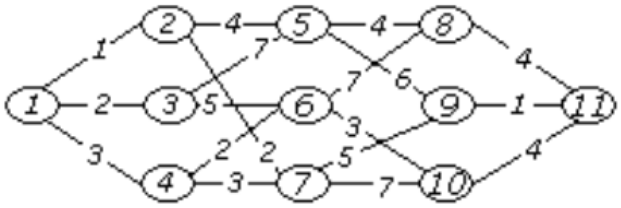
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	1	0	0	0	1	1	1	0
2	1	0	1	0	0	0	0	0	1	1
3	1	1	0	1	1	0	0	0	0	0
4	0	0	1	0	1	0	0	0	1	0
5	0	0	1	1	0	1	1	0	1	0
6	0	0	0	0	1	0	0	0	1	0
7	1	0	0	0	1	0	0	0	0	0
8	1	0	0	0	0	0	0	0	1	0
9	1	1	0	1	1	1	0	1	0	0
10	0	1	0	0	0	0	0	0	0	0

Таблиця Ваг

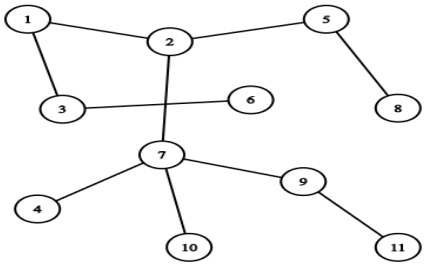
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	1	2	2	2	1	1	1	2
2	1	0	1	2	2	2	2	2	1	1
3	1	1	0	1	1	2	2	2	2	2
4	2	2	1	0	1	2	2	2	1	3
5	2	2	1	1	0	1	1	2	1	3
6	2	2	2	2	1	0	2	2	1	3
7	1	2	2	2	1	2	0	2	2	3
8	1	2	2	2	2	2	2	0	1	3
9	1	1	2	1	1	1	2	1	0	2
10	2	1	2	3	3	3	3	3	2	0

Діаметр = 13.

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



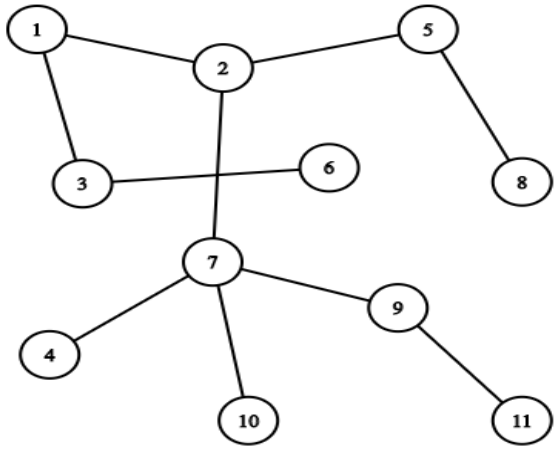
Краскал:



V={1,2,3,7,4,5,8,6,9,11,10}

E={(1,2),(1,3),(2,7),(2,5),(7,4),(3.6),(5,8),(7,9),(9,11),(7, 10).

Прима:

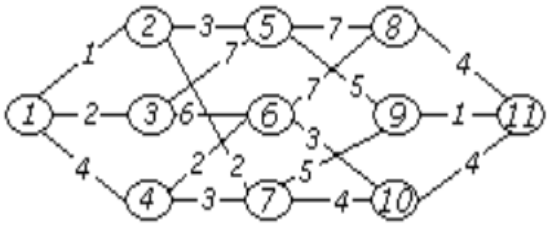


V={1,2,9,11,3,7,4,5,8,6,10}

E={(1,2), (9,11),(1,3),(2,7),(2,5),(7,4),(3.6),(5,8),(7,9),(7, 10).

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:



*main.cpp

```cpp
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

class Edge
{
    public:
    int src, dest, weight;
};

class Graph
{
    public:
    int V, E;
    Edge* edge;
};

Graph* createGraph(int V, int E)
{
    Graph* graph = new Graph;
    graph->V = V;
    graph->E = E;
    graph->edge = new Edge[E];
    return graph;
}


class subset
{   public:
    int parent;
    int rank;
};


int find(subset subsets[], int i)
{
    if (subsets[i].parent != i)
    subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}

void Union(subset subsets[], int x, int y)
```

*main.cpp

```
42  void Union(subset subsets[], int x, int y)
43  {
44      int xroot = find(subsets, x);
45      int yroot = find(subsets, y);
46
47
48      if (subsets[xroot].rank < subsets[yroot].rank)
49          subsets[xroot].parent = yroot;
50      else if (subsets[xroot].rank > subsets[yroot].rank)
51          subsets[yroot].parent = xroot;
52      else
53      {
54          subsets[yroot].parent = xroot;
55          subsets[xroot].rank++;}
56  }
57
58  int myComp(const void* a, const void* b)
59  {
60      Edge* a1 = (Edge*)a;
61      Edge* b1 = (Edge*)b;
62      return a1->weight > b1->weight;
63  }
64
65  void Kruskal(Graph* graph)
66  {
67      int V = graph->V;
68      Edge result[V];
69      int e = 0;
70      int i = 0;
71
72      qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);
73      subset *subsets = new subset[( V * sizeof(subset) )];
74
75      for (int v = 0; v < V; ++v)
76      {   subsets[v].parent = v;
77          subsets[v].rank = 0;}
78
79      while (e < V - 1 && i < graph->E)
80      {
81          Edge next_edge = graph->edge[i++];
82
83          int x = find(subsets, next_edge.src);
```

*main.cpp

```
74
75      for (int v = 0; v < V; ++v)
76      {   subsets[v].parent = v;
77          subsets[v].rank = 0;}
78
79      while (e < V - 1 && i < graph->E)
80      {
81          Edge next_edge = graph->edge[i++];
82
83          int x = find(subsets, next_edge.src);
84          int y = find(subsets, next_edge.dest);
85
86          if (x != y)
87          {   result[e++] = next_edge;
88              Union(subsets, x, y);}
89      }
90      cout<<"Following are the edges in the constructed Kruskal graph: \n";
91      for (i = 0; i < e; ++i)
92          cout<<result[i].src<<" -- "<<result[i].dest<<" == "<<result[i].weight<<endl;
93      return;
94  }
95
96  int main()
97  {
98      int V;
99      int E;
100     cout<<"Enter number of Vershun:";
101     cin>>V;
102     cout<<"Enter number of Rebes";
103     cin>>E;
104     Graph* graph = createGraph(V, E);
105
106     for(int k=0; k<18; k ++){
107     cout<<"Enter two Vershuna:";
108     cin>>graph->edge[k].src ;
109     cin>>graph->edge[k].dest ;
110     cout<<"Enter weight of edge:";
111     cin>>graph->edge[k].weight;}
112
113     Kruskal(graph);
114     return 0;
115 }
```

Results:

```
Select "D:\labys\dyskretka\Laba2\Laba 4\bin\D...
Enter weight of edge:4
Enter two Vershuna:10 7
Enter weight of edge:4
Enter two Vershuna:7 4
Enter weight of edge:3
Enter two Vershuna:3 1
Enter weight of edge:4
Enter two Vershuna:1 3
Enter weight of edge:2
Enter two Vershuna:3 6
Enter weight of edge:6
Enter two Vershuna:11 9
Enter weight of edge:1
Enter two Vershuna:2 7
Enter weight of edge:2
Enter two Vershuna:3 5
Enter weight of edge:7
Enter two Vershuna:6 4
Enter weight of edge:2
Enter two Vershuna:7 9
Enter weight of edge:5
Enter two Vershuna:6 10
Enter weight of edge:3
Enter two Vershuna:6 8
Enter weight of edge:7
Enter two Vershuna:9 5
Enter weight of edge:5
Following are the edges in the constructed MST
1 -- 2 == 1
2 -- 5 == 3
6 -- 10 == 3
8 -- 11 == 4
11 -- 10 == 4
10 -- 7 == 4
7 -- 4 == 3
3 -- 1 == 4
9 -- 5 == 5
11 -- 9 == 1

Process returned 0 (0x0)   execution time : 301.666
 s
Press any key to continue.
```

Висновок:

Я набув практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.