

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Лабораторна робота №4

З дисципліни

«Дискретна математика»

Виконав :

Студент КН-113

Сайкевич В.А.

Викладач:

Мельникова Н.І.

Тема : «Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Прима-Краскала»

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

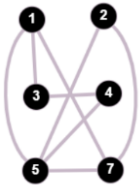
Варіант №14

Завдання 1

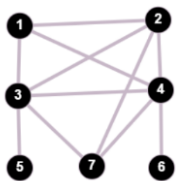
1. Виконати наступні операції над графами:



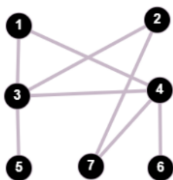
1) знайти доповнення до першого графу



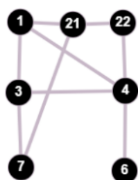
2) об'єднання графів



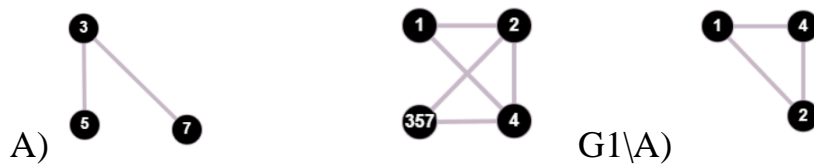
3) кільцеву суму $G1$ та $G2$ ($G1+G2$)



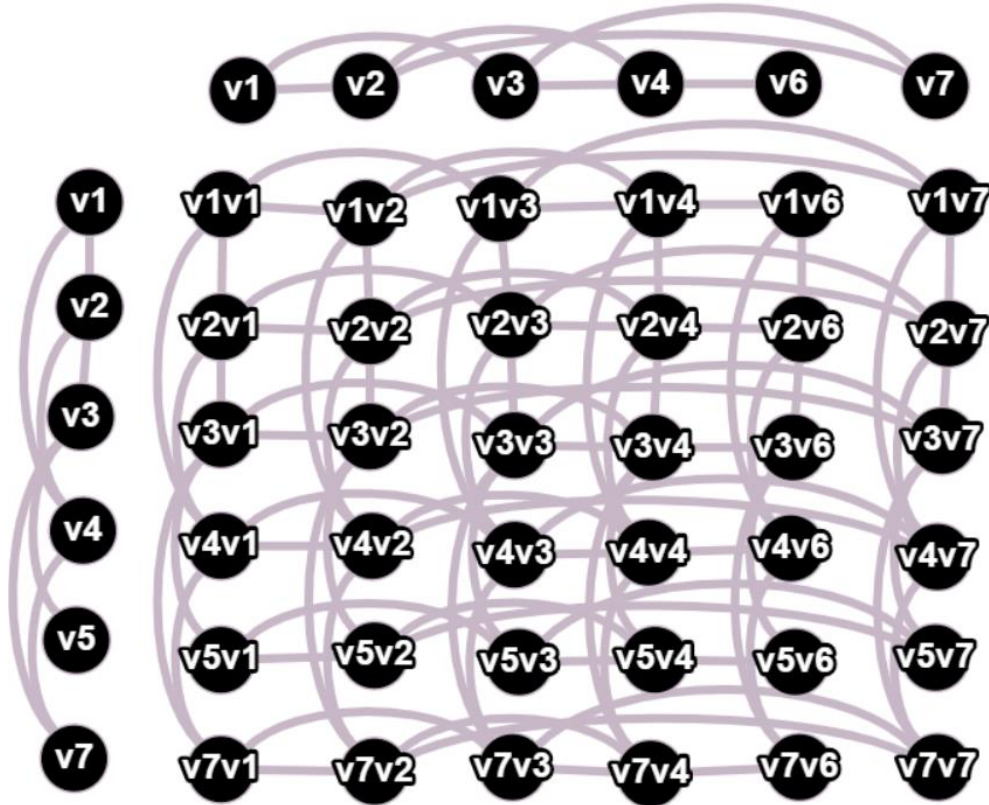
4) розщепити вершину у другому графі



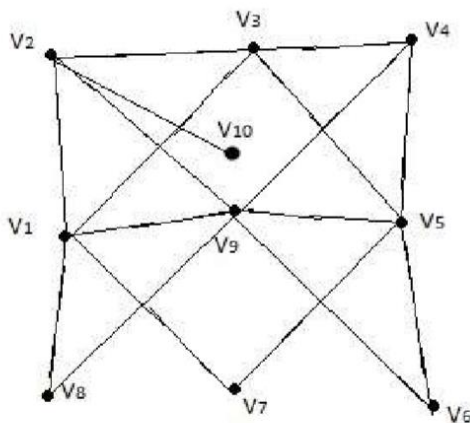
5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 ($G1 \setminus A$)



б) добуток графів



2. Знайти таблицю суміжності та діаметр графа



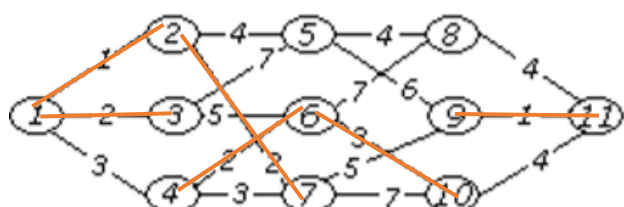
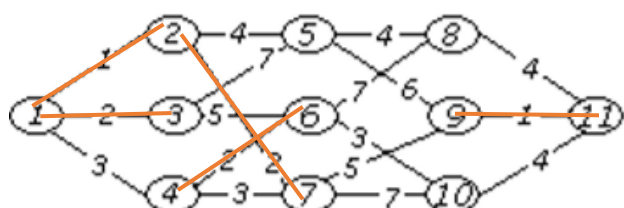
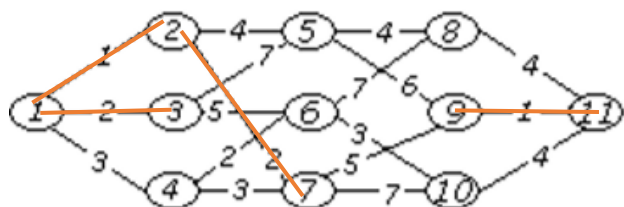
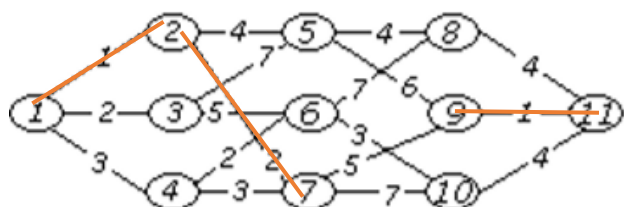
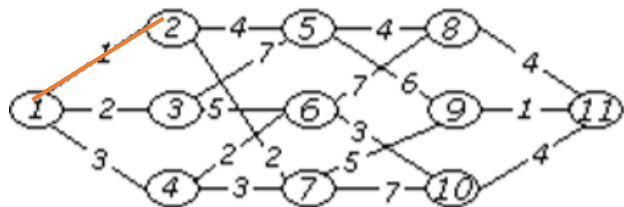
	1	2	3	4	5	6	7	8	9	10
1	—	1	1	0	0	0	1	1	1	0
2	1	—	1	0	0	0	0	0	1	1
3	1	1	—	1	1	0	0	0	0	0
4	0	0	1	—	1	0	0	0	1	0
5	0	0	1	1	—	1	1	0	1	0
6	0	0	0	0	1	—	0	0	1	0
7	1	0	0	0	1	0	—	0	0	0
8	1	0	0	0	0	0	0	—	1	0
9	1	1	0	1	1	1	0	1	—	0
10	0	1	0	0	0	0	0	0	0	—

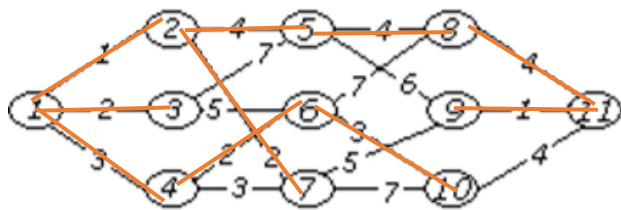
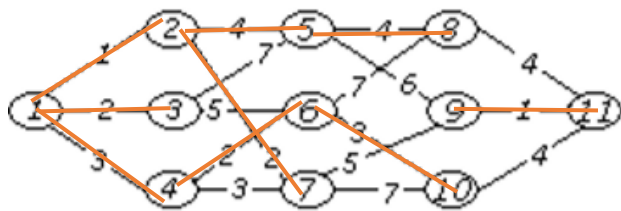
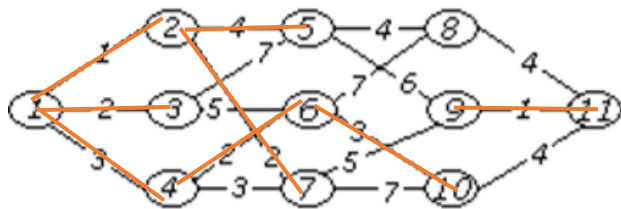
Діаметр графа=3

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

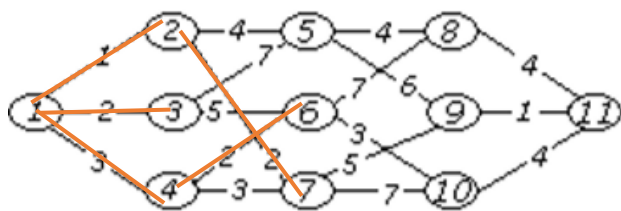
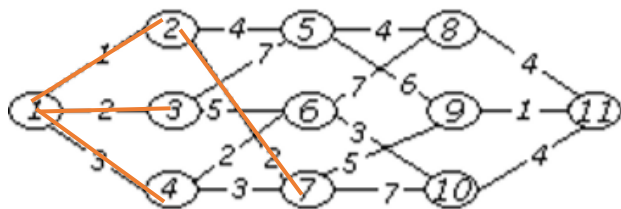
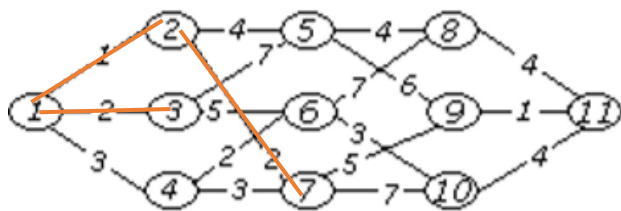
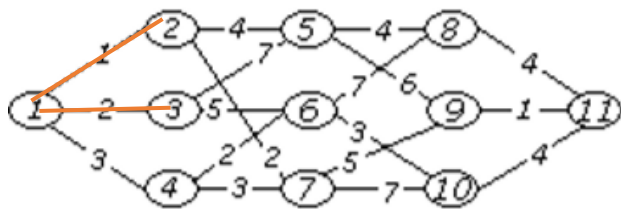
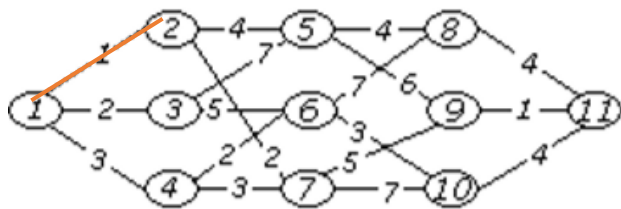


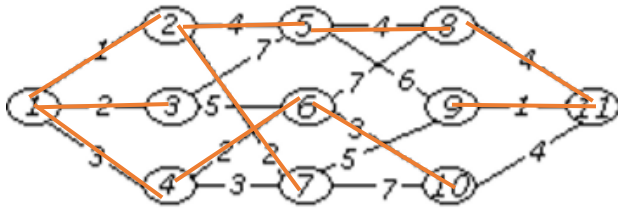
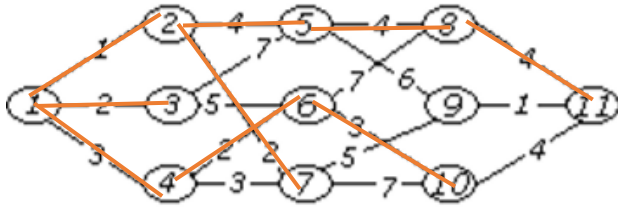
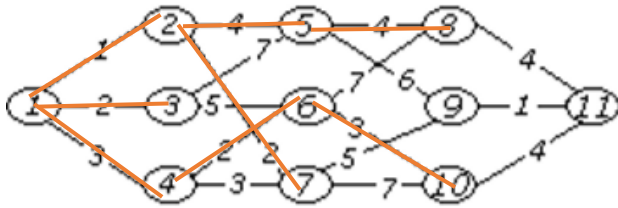
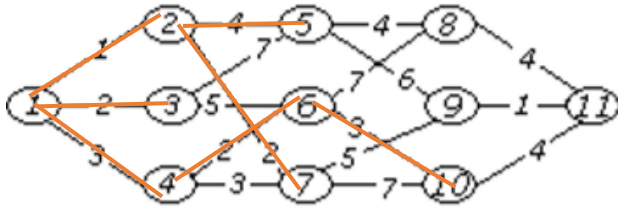
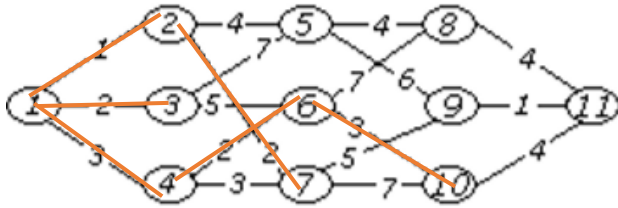
Краскала:





Прима:





Завдання 2

Завдання : Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги за алгоритмом Краскала. Етапи розв'язання задачі виводити на екран. Протестувати розроблену програму на наступному графі:



Код

1) Прима

```
1 #include <iostream>
2 #include <math.h>
3 #include <conio.h>
4
5 using namespace std;
6
7 void output(int size, int** adjrcencyarr) {
8     cout << " ";
9     for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; }
10    for (int i = 0; i < size; i++) {
11        cout << endl << setw(3) << i + 1;
12        for (int j = 0; j < i; j++) {
13            cout << setw(3) << adjrcencyarr[j][i];
14        }
15        cout << " -";
16        for (int j = i + 1; j < size; j++) {
17            cout << setw(3) << adjrcencyarr[i][j];
18        }
19    }
20 }
21
22 void main()
23 {
24     int size;
25     cout << "Enter amount of vertices ";
26     cin >> size;
27     int** adjrcencyarr = new int* [size];
28     bool* vertex = new bool[size];
29     cout << "Enter adjacency matrix \n";
30     cout << " ";
31
32     for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; adjrcencyarr[i] = new int[size]; vertex[i] = 0; }
33     cout << endl;
34     for (int i = 0; i < size; i++) {
35         cout << setw(3) << i + 1;
36         for (int j = 0; j < i; j++) { cout << setw(3) << adjrcencyarr[j][i]; }
37         cout << " -";
38         for (int j = i + 1; j < size; j++) { cin >> adjrcencyarr[i][j]; }
39     }
40     cout << endl;
41     vertex[0] = 1;
42     int min, minI, minJ;
43     for (int n = 0; n < size - 1; n++) {
44         min = 100, minI = 0, minJ = 0;
45         for (int i = 0; i < size; i++) {
46             for (int j = i + 1; j < size; j++) {
47                 if (adjrcencyarr[i][j] < min && adjrcencyarr[i][j] != 0 && (vertex[i] && !vertex[j])) {
48                     min = adjrcencyarr[i][j];
49                     minI = i, minJ = j;
50                 }
51             }
52         }
53         vertex[minI] = 1, vertex[minJ] = 1;
54         cout << n + 1 << "connected " << minI + 1 << " - " << minJ + 1 << endl;
55     }
```

Результати :

```
Enter amount of vertices 11
Enter adjacency matrix
 1  2  3  4  5  6  7  8  9 10 11
1 -  1  2  4  0  0  0  0  0  0  0
2 1 -  0  0  3  0  2  0  0  0  0
3 2 0 -  0  7  6  0  0  0  0  0
4 4 0 0 -  0  2  3  0  0  0  0
5 0 3 7 0 -  0  0  7  5  0  0
6 0 0 6 2 0 -  0  7  0  3  0
7 0 2 0 3 0 0 -  0  5  4  0
8 0 0 0 0 7 7 0 -  0  0  4
9 0 0 0 0 5 0 5 0 -  0  1
10 0 0 0 0 0 0 3 4 0 0 -  4
11 0 0 0 0 0 0 0 4 1 4 -
1)connected 1-2
2)connected 1-3
3)connected 2-7
4)connected 2-5
5)connected 1-4
6)connected 4-6
7)connected 6-10
8)connected 10-11
9)connected 5-9
10)connected 5-8
```

2) Краскала

```

1  #include <iostream>
2  #include <iomanip>
3
4  using namespace std;
5
6  int find(int i,int* vertex)
7  {
8      while (vertex[i] != i)
9          i = vertex[i];
10     return i;
11 }
12
13 void output(int size, int** adjacencyarr) {
14     cout << " ";
15     for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; }
16     for (int i = 0; i < size; i++) {
17         cout << endl << setw(3) << i + 1;
18         for (int j = 0; j < i; j++) {
19             cout << setw(3) << adjacencyarr[j][i];
20         }
21         cout << " -";
22         for (int j = i + 1; j < size; j++) {
23             cout << setw(3) << adjacencyarr[i][j];
24         }
25     }
26 }
27
28 void main()
29 {
30     int size;
31
32     cout << "Enter amount of vertices ";
33     cin >> size;
34     int** adjacencyarr = new int* [size];
35     int* vertex = new int[size];
36     cout << "Enter adjacency matrix \n";
37     cout << " ";
38     for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; adjacencyarr[i] = new int[size]; vertex[i] = i; }
39     cout << endl;
40     for (int i = 0; i < size; i++) {
41         cout << setw(2) << i + 1;
42         for (int j = 0; j < i; j++) { cout << setw(3) << adjacencyarr[j][i]; }
43         cout << " -";
44         for (int j = i + 1; j < size; j++) { cin >> adjacencyarr[i][j]; }
45     }
46     cout << endl;
47     int min, minI, minJ;
48     for (int n = 0; n < size - 1; n++) {
49         min = 100, minI = 0, minJ = 0;
50         for (int i = 0; i < size; i++) {
51             for (int j = i + 1; j < size; j++) {
52                 if (adjacencyarr[i][j] < min && adjacencyarr[i][j] != 0 && (find(i, vertex) != find(j, vertex))) {
53                     min = adjacencyarr[i][j];
54                     minI = i, minJ = j;
55                 }
56             }
57         }
58         vertex[find(minI, vertex)] = find(minJ, vertex);
59         cout << n + 1 << " )connected " << minI + 1 << " - " << minJ + 1 << endl;
60     }
}

```

Результати :

```

Enter amount of vertices 11
Enter adjacency matrix
  1  2  3  4  5  6  7  8  9 10 11
1  -  1  2  4  0  0  0  0  0  0  0
2  1  -  0  0  3  0  2  0  0  0  0
3  2  0  -  0  7  6  0  0  0  0  0
4  4  0  0  -  0  2  3  0  0  0  0
5  0  3  7  0  -  0  0  7  5  0  0
6  0  0  6  2  0  -  0  7  0  3  0
7  0  2  0  3  0  0  -  0  5  4  0
8  0  0  0  0  7  7  0  -  0  0  4
9  0  0  0  0  5  0  5  0  -  0  1
10 0  0  0  0  0  3  4  0  0  -  4
11 0  0  0  0  0  0  0  4  1  4  -
1)connected 9-11
2)connected 1-2
3)connected 4-6
4)connected 2-7
5)connected 1-3
6)connected 6-10
7)connected 4-7
8)connected 2-5
9)connected 10-11
10)connected 8-11

```

Висновок:

Ми набули практичних вмінь та навичок з використання алгоритмів Прима і Краскала.