

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА
ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту



Лабораторна робота №4
З курсу “Дискретна математика”

Виконала:
ст.гр. КН-110
Ямнюк Аліна
Викладач:
Мельникова Н.І.

Львів – 2018

Лабораторна робота № 4.

Тема: Основні операції над графами. Знаходження остовамінімальної ваги за алгоритмом Прима-Краскала.

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

ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

Графом G називається пара множин (V, E) , де V – множина вершин v , а E – множина упорядкованих або неупорядкованих пар $e = \{v', v''\}$, що $v' \in V$ & $v'' \in V$. Неорієнтований та орієнтований граф (орграф) відрізняються упорядкованістю пар $e \in E$.

Кратними ребрами називають ребра, що зв'язують одні і ті ж множини. Ребро, що входить у ту ж вершину, звідки й виходить, називається петлею

Мультиграф – граф, що має кратні ребра. Псевдографи мають петлі, а прості графи не мають ні кратних ребер, ні петель.

Будь-яке ребро є інцидентним вершинам (v', v'') , які воно з'єднує. Дві вершини називають суміжними, якщо вони належать до спільного ребра й несуміжними у протилежному випадку. Степенем вершини v називається кількість інцидентних їй ребер.

Граф, що не має ребер, називається пустим. Граф, що не має вершин – нульграфом. Вершина графа, що не інцидентна до жодного з ребер є ізольованою. Вершина із одиничним степенем є листком.

Граф $G' = (V', E')$ є підграфом графа $G = (V, E)$, якщо $V' \subseteq V$ і $E' = \{(v', v'') \mid v' \in V' \text{ & } v'' \in V' \text{ & } (v', v'') \in E\}$. G' також називається кістяковим підграфом, якщо виконано умову $V' = V$ & $E' \subseteq E$.

Над графами можна виконувати деякі операції.

1. Вилученням ребра $e \in E$ можна отримати новий граф $G' = (V, E \setminus \{e\})$
2. Доповненням графа $G = (V, E)$, можна отримати новий граф $G' = (V', E')$, де $E' = \{(v1, v2) \mid (v1, v2) \notin E\}$;
3. Об'єднанням графів $G1 = (V1, E1)$ та $G2 = (V2, E2)$ є новий граф $G = (V, E)$, в якому $V = V1 \cup V2$, $E = E1 \cup E2$;
4. Кільцевою сумою графів $G1 = (V1, E1)$ та $G2 = (V2, E2)$ є граф $G = (V, E)$, в якому $V = V1 \cup V2$, а $E = E1 \Delta E2$;
5. Розщепленням вершин є операція, за якої на місці однієї вершини з'являється дві, інцидентні одна до одної. Нові вершини довільно успадковують інцидентність зі старими вершинами.
6. Стягненням ребра (a, b) є операція, при якій це ребро зникає, і замість точок a та b виникає c , що успадковує їхню інцидентність.
7. Добутком графів $G1 = (V1, E1)$ та $G2 = (V2, E2)$ є граф $G = G1 \times G2 = (V, E)$, в якого $V = V1 \times V2$, а вершини $(v1', v2')$ та $(v1'', v2'')$ суміжні тільки якщо $(v1' = v1'') \text{ & } (v2', v2'') \in E2$ або $(v2' = v2'') \text{ & } (v1', v1'') \in E1$.

Матрицею суміжності $R = [rij]$ графа $G = (V, E)$ є квадратна матриця порядку $|V|$, елементи rij якої визначаються за формулою $rij = \begin{cases} 1, & (vi, vj) \in E \\ 0, & (vi, vj) \notin E \end{cases}$.

Діаметром зв'язного графа є максимально можлива довжина між двома його вершинами.

В неорієнтованому графі $G = (V, E)$ маршрутом довжини j є послідовність $M = \{(v_1, v_2), (v_2, v_3), \dots, (v_{j-2}, v_{j-1}), (v_{j-1}, v_j)\}$, що складається з ребер $e \in E$. При цьому кожен два сусідні ребра мають спільну кінцеву вершину. Маршрут називається ланцюгом, якщо всі його ребра різні. Відкритий ланцюг називається шляхом, якщо всі його вершини різні. Замкнений ланцюг називається циклом, якщо всі його вершини, за винятком кінцевих, є різними. Шлях і цикл називаються гамільтоновими, якщо вони проходять через усі вершини графа.

Для пошуку кістякового дерева мінімальної ваги у зв'язному графі можна використовувати алгоритми Прима та Краскала.

1.1 Алгоритм Прима

Нехай будемо дерево $G' = (V', E')$, де $V' = E' = \emptyset$. Над графом $G = (V, E)$ алгоритм Прима буде мати такий порядок дій:

1. Вибрати довільну точку $v \in V$, і додати її у V' .
2. Вибрати перше найлегше ребро $e = (a, b) \in E$, де $a \in V'$, $b \in V$, що не утворює циклів.
 - а. Якщо такого ребра не знайдено – завершити роботу, повернувши дерево (V', E') .
3. Додати точку b та ребро e у G' .
4. Повернутись до кроку 2.

Якщо граф заданий матрицею суміжності, то ціну цього алгоритму можна оцінити як $O(|V|^2)$. Якщо задати граф списком суміжності, ціною алгоритму буде $O(|E| \log |V|)$.

1.2 Алгоритм Краскала

Нехай будемо дерево $G' = (V', E')$, де $V' = E' = \emptyset$. Над графом $G = (V, E)$ алгоритм Краскала буде мати такий порядок дій:

1. Вибрати найлегше ребро $e = (u, v) \in E$ таке, щоби воно не утворювало циклів.
 - а. Якщо таких ребер немає – завершити роботу, повернувши дерево (V', E') .
2. Додати точки u і v та ребро (u, v) у G' .
3. Повернутись до кроку 1.

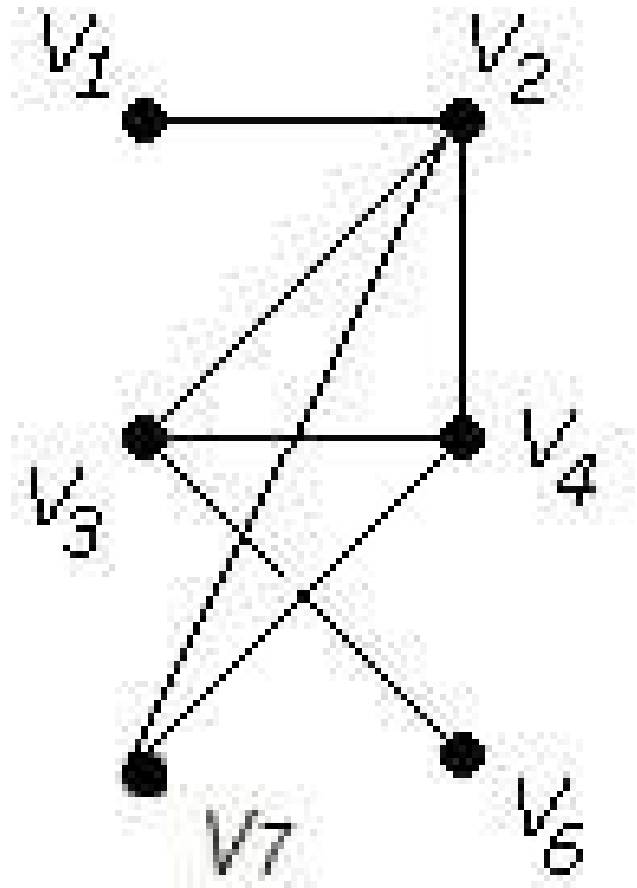
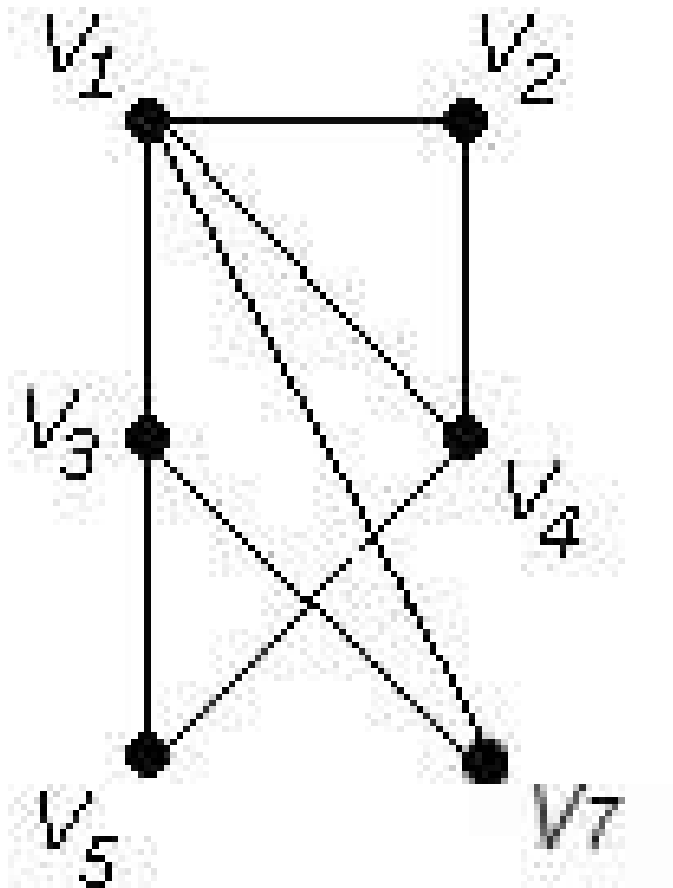
Ціна цього алгоритму - $O(|E| \log |E|)$

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

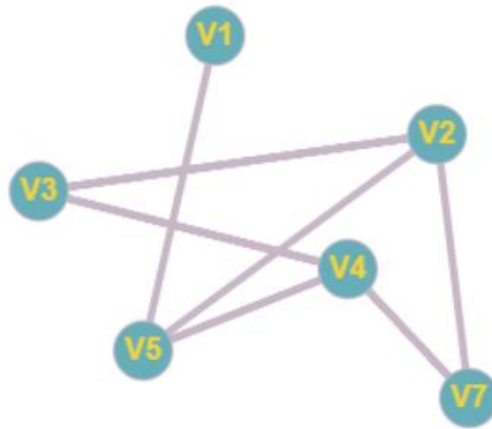
Завдання № 1. Розв'язати на графах наступні задачі:

1. Виконати наступні операції над графами:
 - 1) знайти доповнення до першого графу,
 - 2) об'єднання графів,

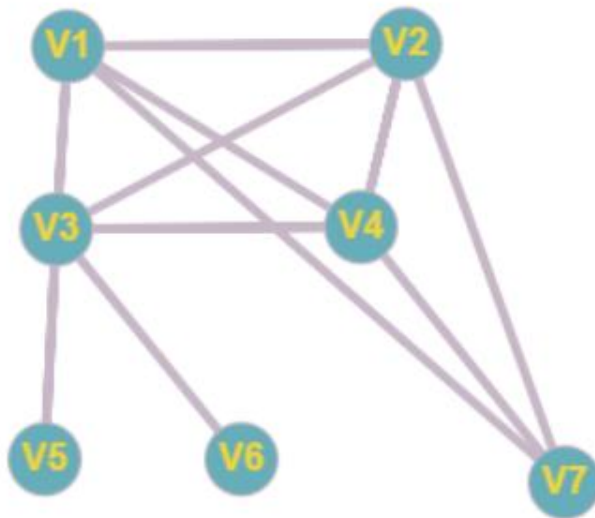
- 3) кільцеву суму $G1$ та $G2$ ($G1+G2$),
- 4) розщепити вершину у другому графі,
- 5) виділити підграф A , що складається з 3-х вершин в $G1$ і знайти стягнення A в $G1$ ($G1 \setminus A$),
- 6) добуток графів.



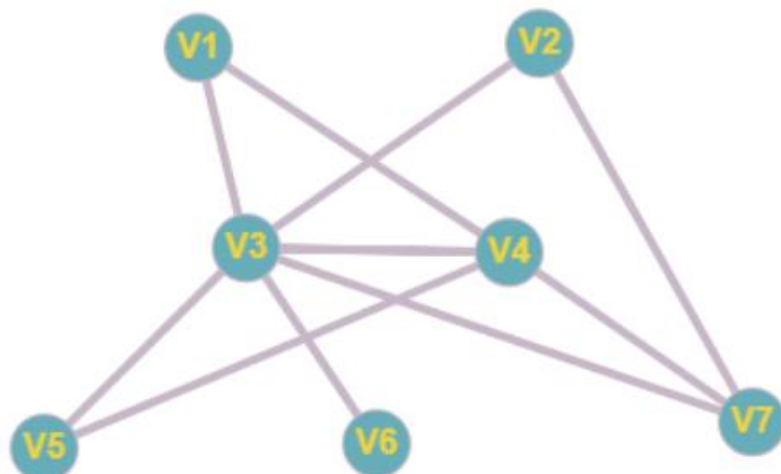
- 1) знайти доповнення до першого графа



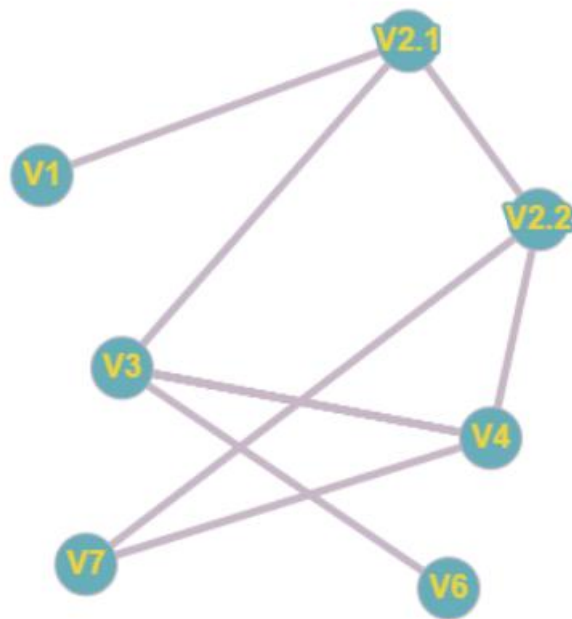
2) об'єднання графів



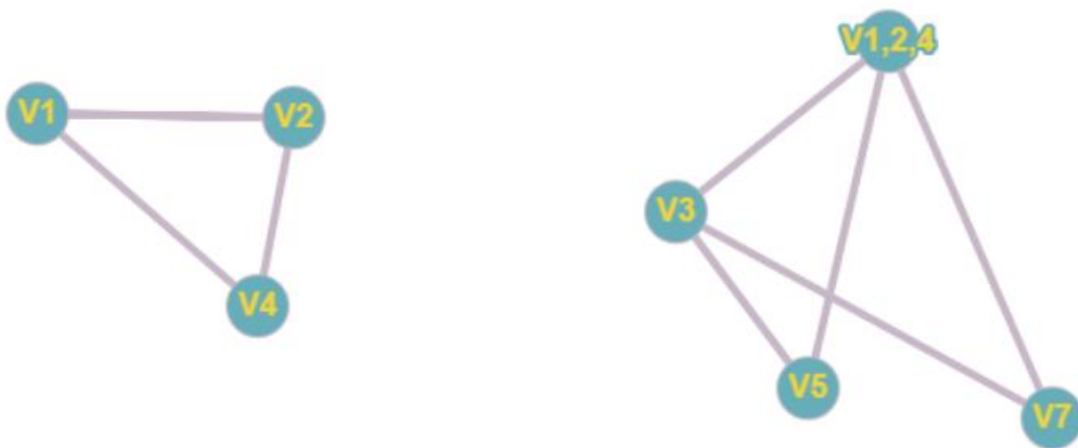
3) кільцеву суму $G1$ та $G2$ ($G1+G2$)



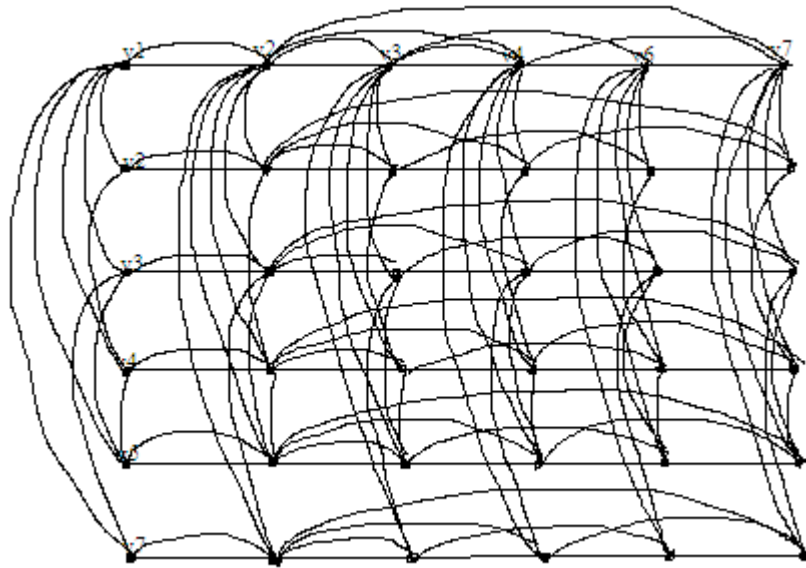
4) розщепити вершину у другому графі



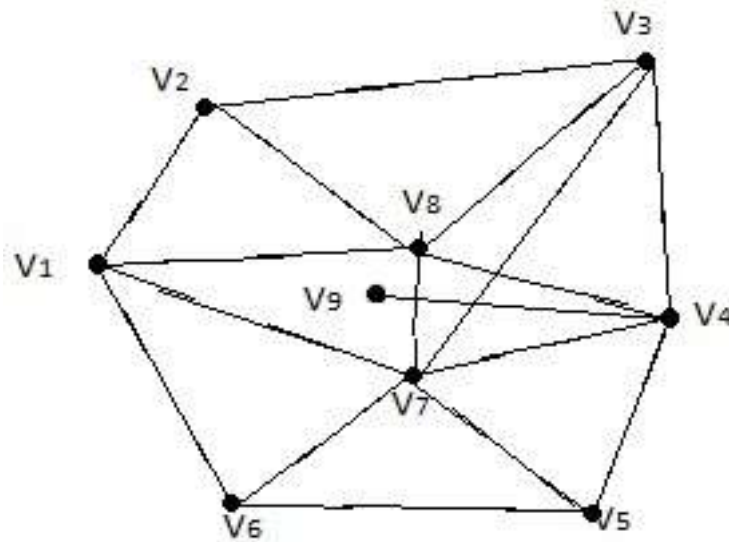
5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 ($G1 \setminus A$)



6) добуток графів



2. Знайти таблицю суміжності та діаметр графа.



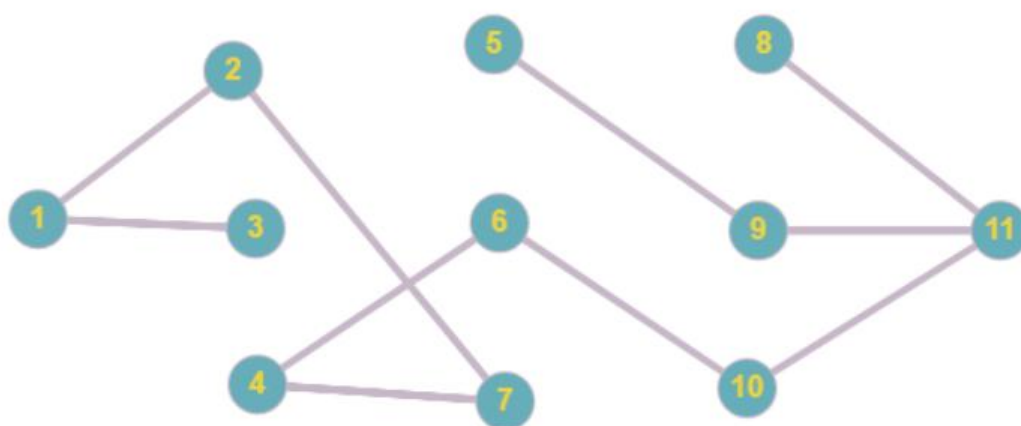
Діаметр = 3.

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

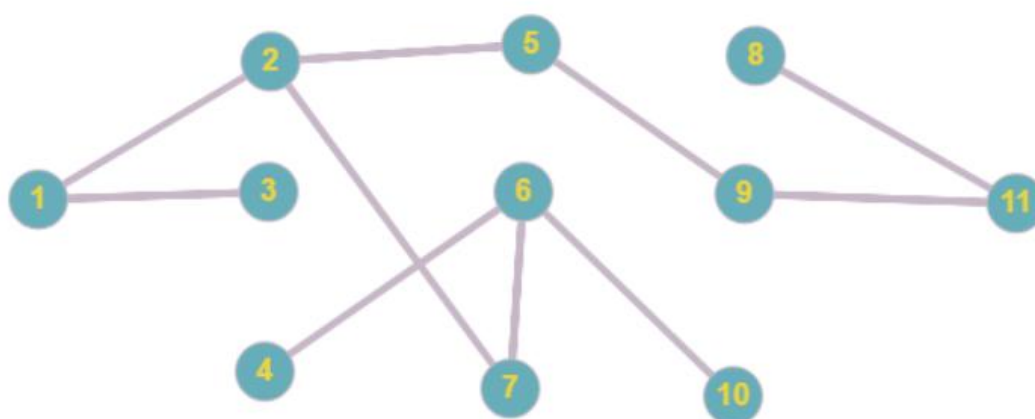
3. Знайти двома методами (Краскала і Прима) мінімальне остоведерево графа.



Метод Прима:

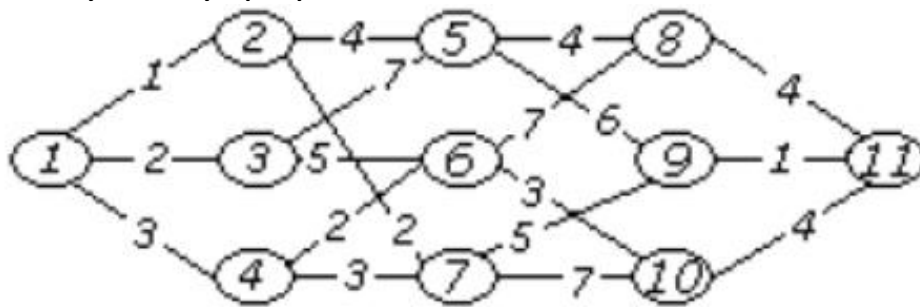


Метод Краскала:



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:



Код:

```
1 #include<stdio.h>
2 #define MAX 30
3
4 typedef struct edge
5 {
6     int u,v,w;
7 }edge;
8
9 typedef struct edgelist
10 {
11     edge data[MAX];
12     int n;
13 }edgelist;
14
15 edgelist elist;
16
17 int G[MAX][MAX],n;
18 edgelist spanlist;
19
20 void kruskal();
21 int find(int belongs[],int vertexno);
22 void union1(int belongs[],int c1,int c2);
23 void sort();
24 void print();
25
26 int main()
27 {
28     int i,j;
29
30     printf("\nEnter number of vertices:");
31
32     scanf("%d",&n);
33
34     printf("\nEnter the adjacency matrix:\n");
35
36     for(i=0;i<n;i++)
37         for(j=0;j<n;j++)
38             scanf("%d",&G[i][j]);
39
40     for(i=0;i<n;i++){
```

```

41         for(j=0;j<n;j++)
42         {
43             printf("%d ",G[i][j]);
44         }
45         printf ("\n");
46     }
47     printf("\n");
48     kruskal();
49     print();
50 }
51
52 void kruskal()
53 {
54     int belongs[MAX],i,j,cno1,cno2;
55     elist.n=0;
56
57     for(i=1;i<n;i++)
58         for(j=0;j<i;j++)
59             {
60                 if(G[i][j]!=0)
61                 {
62                     elist.data[elist.n].u=i;
63                     elist.data[elist.n].v=j;
64                     elist.data[elist.n].w=G[i][j];
65                     elist.n++;
66                 }
67             }
68
69     sort();
70
71     for(i=0;i<n;i++)
72         belongs[i]=i;
73
74     spanlist.n=0;
75
76     for(i=0;i<elist.n;i++)
77     {
78         cno1=find(belongs,elist.data[i].u);
79         cno2=find(belongs,elist.data[i].v);
80
81         if(cno1!=cno2)
82         {
83             spanlist.data[spanlist.n]=elist.data[i];
84             spanlist.n=spanlist.n+1;
85             union1(belongs,cno1,cno2);
86         }
87     }
88 }
89
90 int find(int belongs[],int vertexno)
91 {
92     return(belongs[vertexno]);
93 }
94
95 void union1(int belongs[],int c1,int c2)
96 {
97     int i;
98
99     for(i=0;i<n;i++)
100         if(belongs[i]==c2)
101             belongs[i]=c1;
102 }
103
104 void sort()
105 {
106     int i,j;
107     edge temp;
108
109     for(i=1;i<elist.n;i++)
110         for(j=0;j<elist.n-1;j++)
111             if(elist.data[j].w>elist.data[j+1].w)
112             {
113                 temp=elist.data[j];
114                 elist.data[j]=elist.data[j+1];
115                 elist.data[j+1]=temp;
116             }
117 }
118
119 void print()
120 {

```

```

121     int i, cost=0;
122
123     for(i=0; i<spanlist.n; i++)
124     {
125         printf("\n%d\t%d\t%d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);
126         cost=cost+spanlist.data[i].w;
127     }
128
129     printf("\n\nCost of the spanning tree=%d", cost);
130     printf("\n");
131 }

```

Результат:

Enter number of vertices:11

Enter the adjacency matrix:

```

0 1 2 3 0 0 0 0 0 0 0
1 0 0 4 0 2 0 0 0 0 0
2 0 0 0 7 5 0 0 0 0 0
3 0 0 0 0 2 3 0 0 0 0
0 4 7 0 0 0 0 4 6 0 0
0 0 5 2 0 0 0 7 0 3 0
0 2 0 3 0 0 0 0 5 7 0
0 0 0 0 4 7 0 0 0 0 4
0 0 0 0 6 0 5 0 0 0 1
0 0 0 0 0 3 7 0 0 0 4
0 0 0 0 0 0 0 4 1 4 0

```

```

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

```

Cost of the spanning tree=26

Висновок: набула практичних навичок з використанням алгоритмів Прима і Краскала.