МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни
«Дискретна математика»

Виконала:
студентка КН-113
Пеленська Софія
Викладач:
Мельникова Н.І.

Львів – 2019 р.

Тема:

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

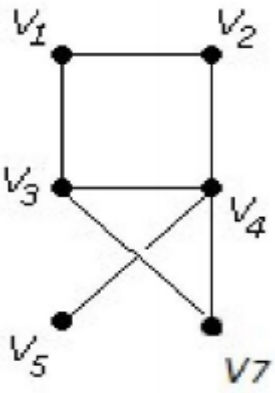
Мета:

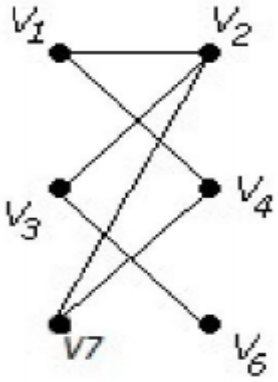
набуття практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.

Варіант № 8

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

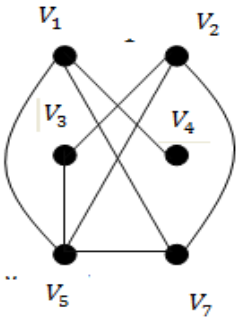
1. G_1 G_2



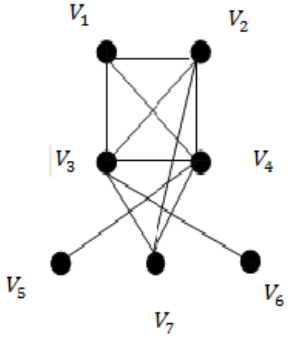


- Знайти дповнення до графу G_1 :

G_1^1



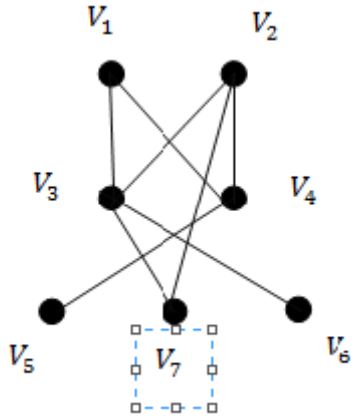
- Знайти обєднання графів G_1 та G_2:



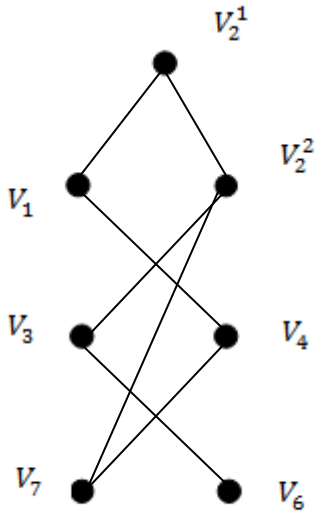
- Знайти кільцеву суму графів G_1 та G_2:

Всі вершини об’єднюються, а ребра симетрична різниця

$$E_1 \Delta E_2 = (E_1 \cup E_2)/(E_1 \cap E_2)$$

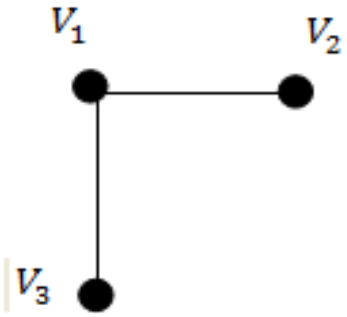


- Розщипити вершину в G_2:
  За умовою не сказану яку потрібно розщипити вершину. Нехай візьмемо вершину V_2.

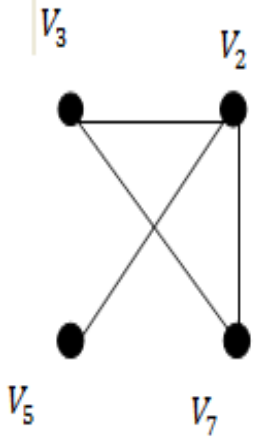


- Виділити підграф А першого графа G_1, який складається з 3 вершин(нехай це V_1, V_2, V_3) і знайти стягнення А в G_1 (G_1/А):ї

А

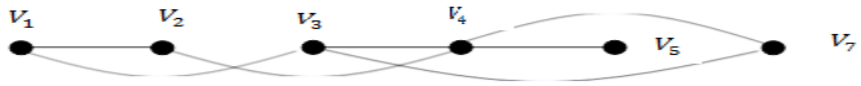


Отже треба стягнути ребра (V_1, V_2) та(V_1, V_3):

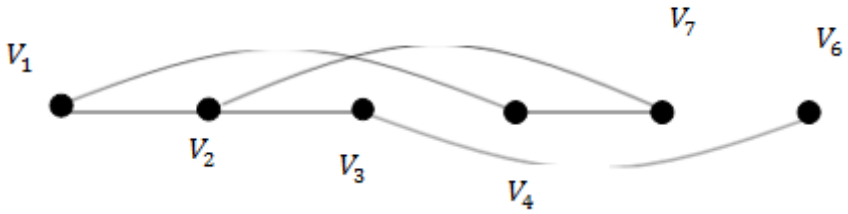


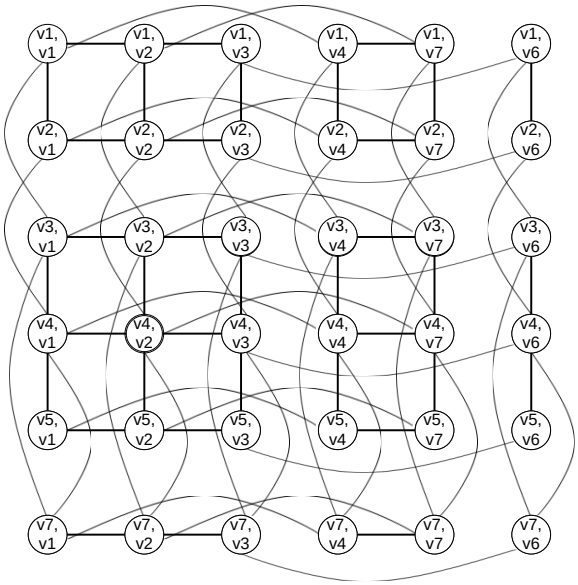
- Знайти добуток графів G_1 та G_2:

Кожен граф має 6 вершин. Тому добуток матиме 36 вершин:
G_1 :

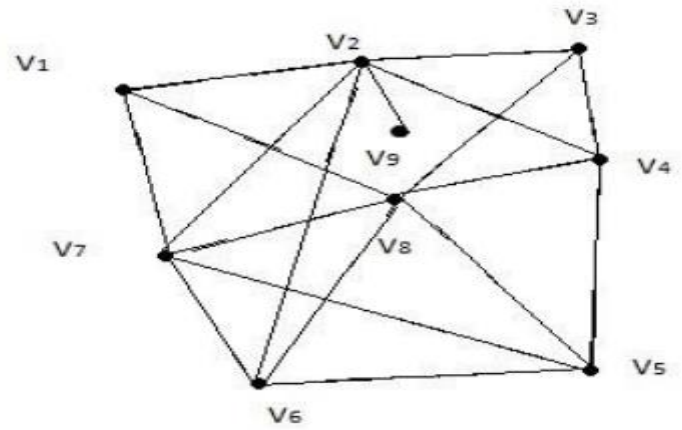


G_2





2. Знайти таблицю суміжності та діаметр графа



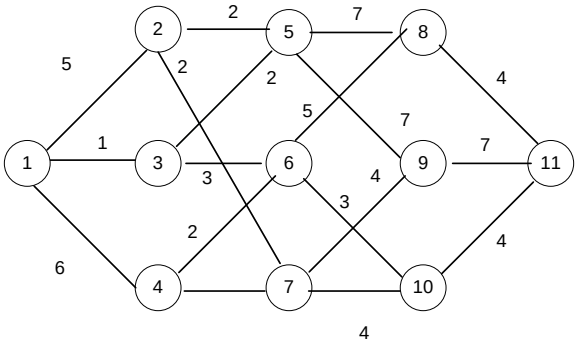
.

Матриця суміжності:

$$\begin{matrix} V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 & V_9 \end{matrix}$$
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

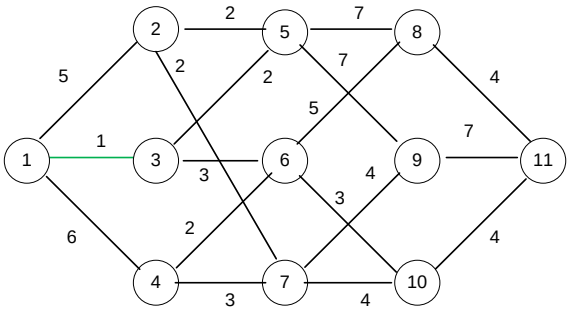
Діаметр графа буде дорівнювати 3. Тому що максимальна довжина найкоротшого шляху між вершинами V_5 та V_9.

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

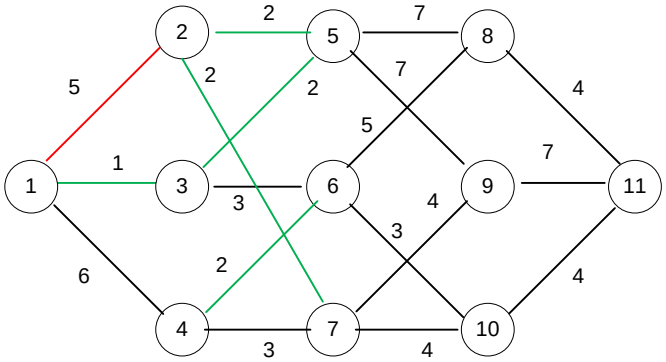


<u>Метод Краскала:</u>

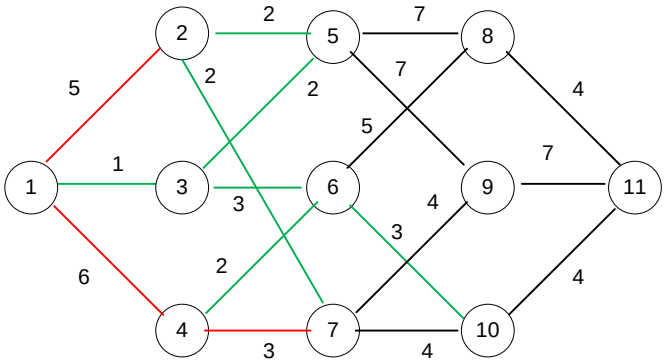
Етап 1: вибираємо ребро(1,3) з найменшою вагою (1) і додаєм до кістякового дерева:



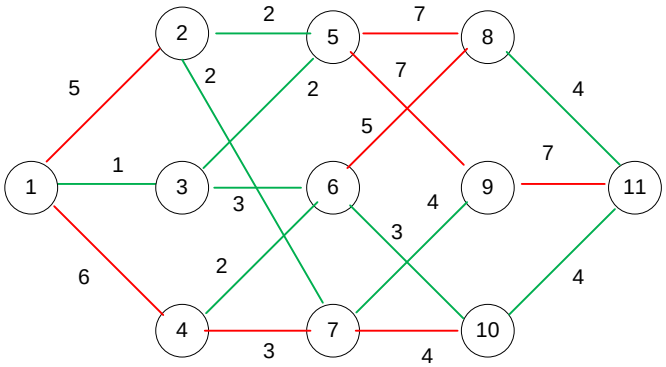
Етап 2: берем ребра з вагою 2 ((2,7), (2,5), (3,5), (4,6)) і додаєм їх до кістякового дерева, але тоді ребро (1,2) нам не підходить, тому що це ребро утворить цикл:



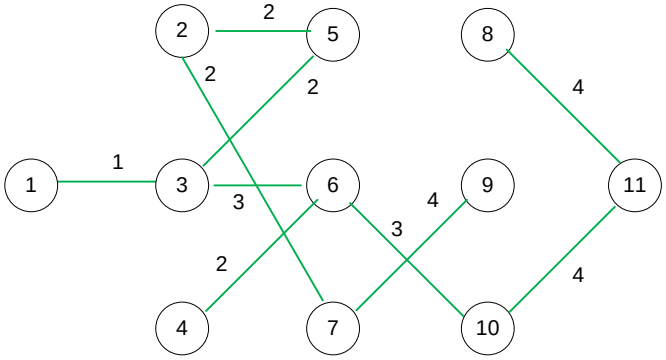
Етап 3: берем ребра з вагою 3 ((3,6),(6,10),(4,7)) і додаєм їх до кістякового дерева, але ребра (4,7) та (1,4) нам не підходить, тому що це ребра утворять цикли:



Етап 4:берем ребра з вагою 4 ((7,9), (8,11), (7,10), (10,11)) і додаєм їх до кістякового дерева, але ребра (7,10), (9,11), (5,9),(6,7), (5,8), тому що вони утворять цикли:

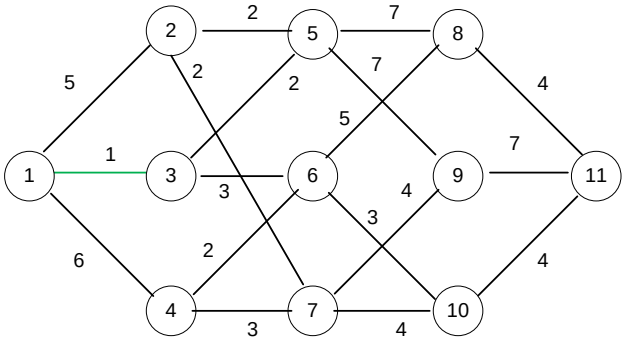


Остове дерево:

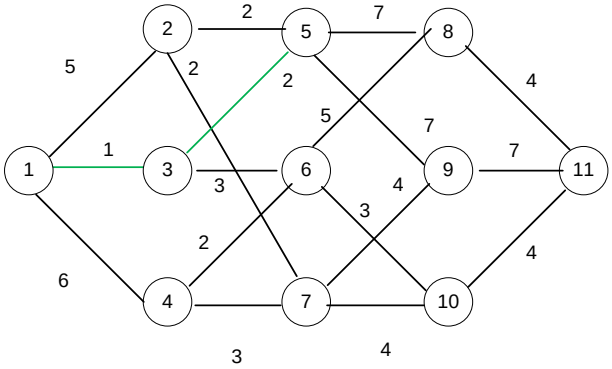


Метод Прима

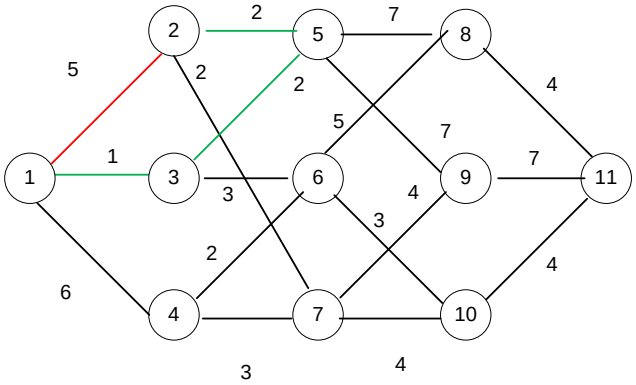
Етап 1: берем ребро (1,3) з вагою 1 і додаєм до кістякового дерева:



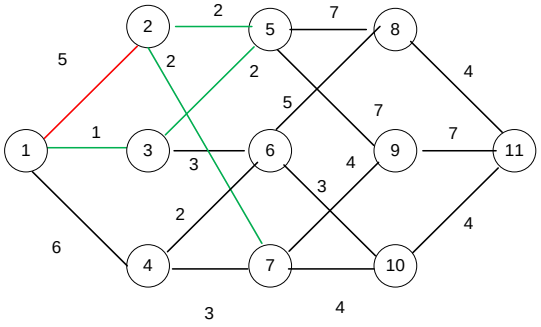
Етап 2: дивимося на вершина 1 та 3 і дивомось яке ребро з найменшою вагою і це є (3,5):



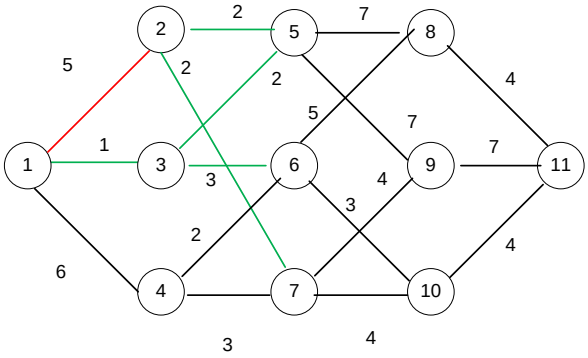
Етап 3: дивимося на вершини 1 і 3 і 5 і дивимося яке ребро найменше з вагою з цих вершин і це є (5,2), а також (1, 2) нам не підходить, тому що це ребро утворює цикл:



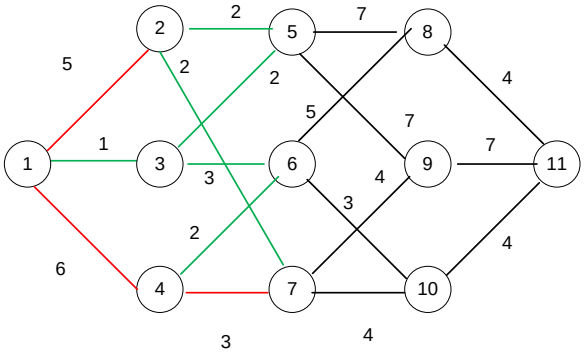
Етап 4: дивимося вершини 1 і 3 і 2 і 5 і дивимося яке ребро найменше з вагою з цих вершин і це є (7,2):



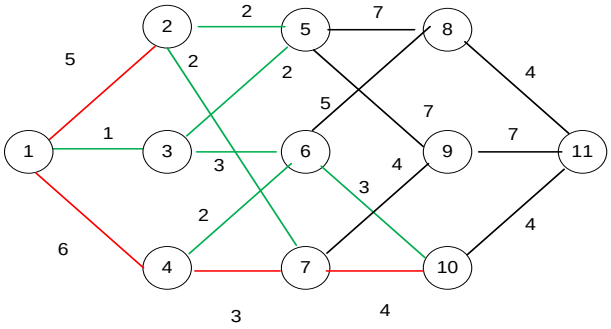
Етап 5: дивимося вершини 1 і 3 і 2 і 5 і 7 і дивимося яке ребро найменше з вагою з цих вершин і це є (3,6):



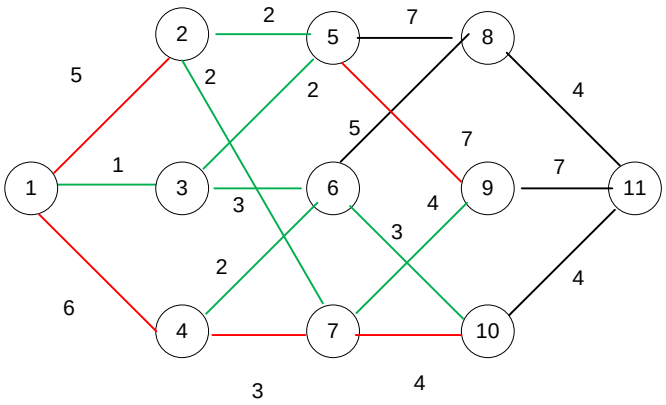
Етап 6: дивимося вершини 1 і 3 і 2 і 5 і 6 і 7 і дивимося яке ребро найменше з вагою з цих вершин і це є (4,6), але ребра (1,4), (4,7) не підходять, тому що вони утворять цикли:



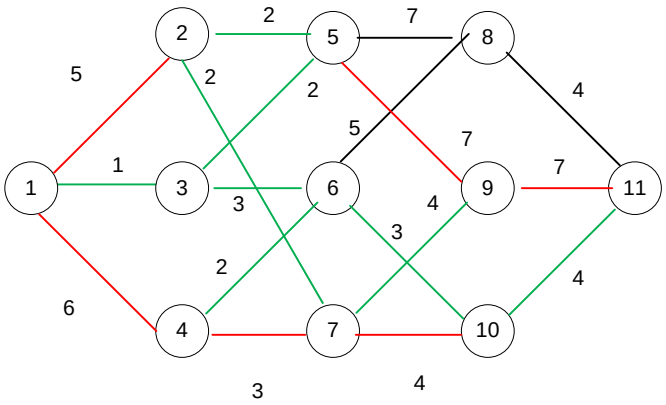
Етап 7: дивимося вершини 1 і 3 і 2 і 5 і 6 і 7 і дивимося яке ребро найменше з вагою з цих вершин і це є (10,6), але ребро (10,7) не підходить, тому що ребро утворить цикл:



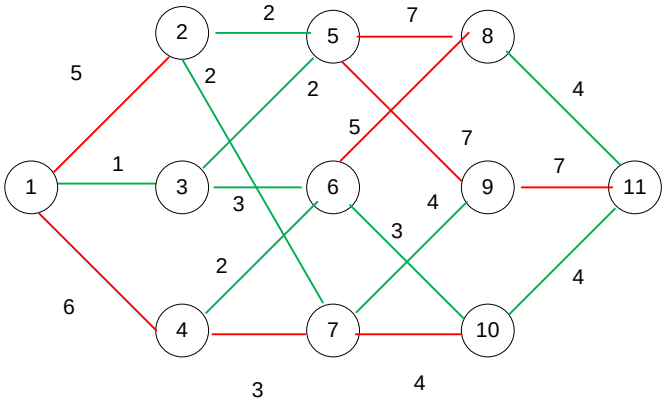
Етап 8: дивимося вершини 1 і 3 і 2 і 5 і 6 і 7 і 10 і дивимося яке ребро найменше з вагою з цих вершин і це є (7,9), але ребро (5,9) не підходить, тому що ребро утворить цикл:



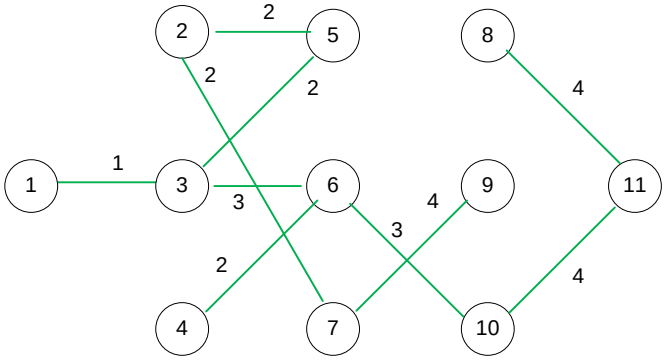
Етап 9: дивимося вершини 1 і 3 і 2 і 5 і 6 і 7 і 9 і 10 і дивимося яке ребро найменше з вагою з цих вершин і це є (10,11), але ребро (9,11) не підходить, тому що ребро утоврить цикл:



Етап 10: дивимося вершини 1 і 3 і 2 і 5 і 6 і 7 і 8 і 9 і 10 і дивимося яке ребро найменше з вагою з цих вершин і це є (8,11), але ребра (5,8) і (8,6)не підходять, тому що ребра утоврять цикли:



Остове дерево:



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

Варіант № 8

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

Код програми:

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   class Edge
5   {
6   public:
7       int src, dest, weight;
8   };
9
10  class Graph
11  {
12  public:
13      int V, E;
14      Edge* edge;
15  };
16
17  Graph* createGraph(int V, int E)
18  {
19      Graph* graph = new Graph;
20      graph->V = V;
21      graph->E = E;
22
23      graph->edge = new Edge[E];
24
25      return graph;
26  }
27
```

```
28  class subset
29  {
30  public:
31      int parent;
32      int rank;
33  };
34
35  int find(subset subsets[], int i)
36  {
37      if (subsets[i].parent != i)
38          subsets[i].parent = find(subsets, subsets[i].parent);
39
40      return subsets[i].parent;
41  }
42
43  void Union(subset subsets[], int x, int y)
44  {
45      int xroot = find(subsets, x);
46      int yroot = find(subsets, y);
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
59  int myComp(const void* a, const void* b)
60  {
61      Edge* a1 = (Edge*)a;
62      Edge* b1 = (Edge*)b;
63      return a1->weight > b1->weight;
64  }
65
66  void KruskalMST(Graph* graph)
67  {
68      int V = graph->V;
69      Edge result[V];
70      int e = 0;
71      int i = 0;
72
73      qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);
74
75      subset *subsets = new subset[( V * sizeof(subset) )];
76
77      for (int v = 0; v < V; ++v)
78      {
79          subsets[v].parent = v;
80          subsets[v].rank = 0;
81      }
```

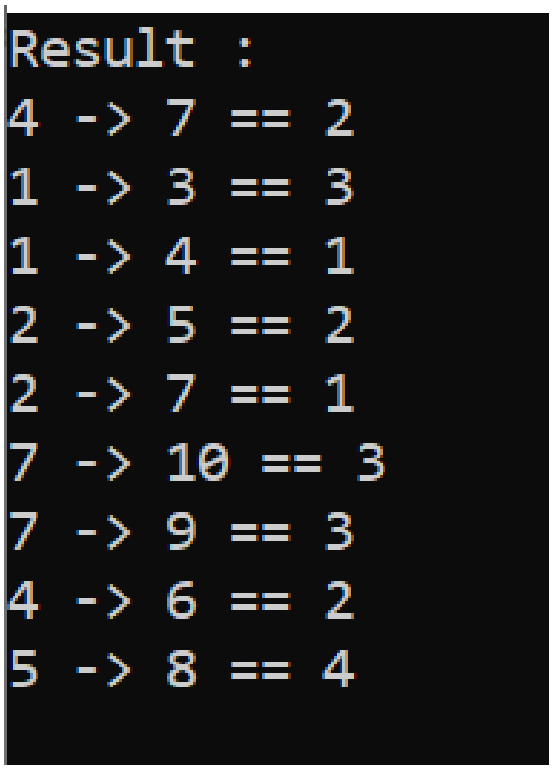
```
82
83      while (e < V - 1 && i < graph->E)
84      {
85          Edge next_edge = graph->edge[i++];
86
87          int x = find(subsets, next_edge.src);
88          int y = find(subsets, next_edge.dest);
89
90          if (x != y)
91          {
92              result[e++] = next_edge;
93              Union(subsets, x, y);
94          }
95      }
96
97      cout<<"Following are the edges in the constructed MST\n";
98      for (i = 0; i < e; ++i)
99          cout<<result[i].src<<" -> "<<result[i].dest<<" == "<<result[i].weight<<endl;
100     return;
101 }
102
103 int main()
104 {
105     int V = 10;
106     int E = 17;
107     Graph* graph = createGraph(V, E);
108
```

```
109
110     graph->edge[0].src = 1;
111     graph->edge[0].dest = 2;
112     graph->edge[0].weight = 4;
113
114     graph->edge[1].src = 1;
115     graph->edge[1].dest = 3;
116     graph->edge[1].weight = 3;
117
118     graph->edge[2].src = 1;
119     graph->edge[2].dest = 4;
120     graph->edge[2].weight = 1;
121
122     graph->edge[3].src = 2;
123     graph->edge[3].dest = 5;
124     graph->edge[3].weight = 2;
125
126     graph->edge[4].src = 2;
127     graph->edge[4].dest = 7;
128     graph->edge[4].weight = 1;
129
130     graph->edge[5].src = 3;
131     graph->edge[5].dest = 5;
132     graph->edge[5].weight = 7;
133
134     graph->edge[6].src = 3;
135     graph->edge[6].dest = 6;
```

```
    graph->edge[6].weight = 7;

    graph->edge[7].src = 4;
    graph->edge[7].dest = 6;
    graph->edge[7].weight = 2;

    graph->edge[8].src = 4;
    graph->edge[8].dest = 7;
    graph->edge[8].weight = 2;

    graph->edge[9].src = 5;
    graph->edge[9].dest = 8;
    graph->edge[9].weight = 4;

    graph->edge[10].src = 5;
    graph->edge[10].dest = 9;
    graph->edge[10].weight = 4;

    graph->edge[11].src = 6;
    graph->edge[11].dest = 8;
    graph->edge[11].weight = 4;

    graph->edge[12].src = 6;
    graph->edge[12].dest = 10;
    graph->edge[12].weight = 5;

     graph->edge[13].src = 7;
```

```
    graph->edge[13].dest = 9;
    graph->edge[13].weight = 3;

    graph->edge[14].src = 7;
    graph->edge[14].dest = 10;
    graph->edge[14].weight = 3;

    graph->edge[15].src = 8;
    graph->edge[15].dest = 11;
    graph->edge[15].weight = 7;

    graph->edge[16].src = 9;
    graph->edge[16].dest = 6;
    graph->edge[16].weight = 11;

    graph->edge[17].src = 10;
    graph->edge[17].dest = 11;
    graph->edge[17].weight = 5;


    KruskalMST(graph);

    return 0;
}
```

Результат програми :



Висновок :

На цій лабораторній роботі я набула практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.