**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”  
  
Кафедра систем штучного інтелекту  
  
  
  
  
  
Лабораторна робота**

**з дисципліни**

**«Дискретна математика»**

**Виконав:  
студент групи КН-110  
Пиріг Богдан  
Викладач:  
Мельникова Н.І.**

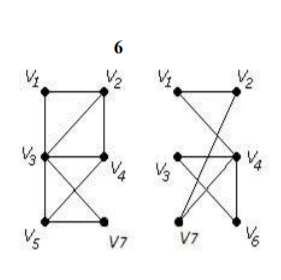
**Львів – 2018 р.Лабораторна робота № 4**

**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.**

**Варіант №6**

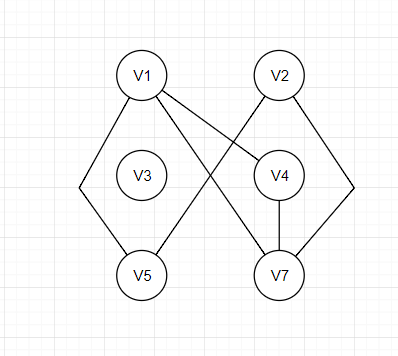
**Завдання №1**



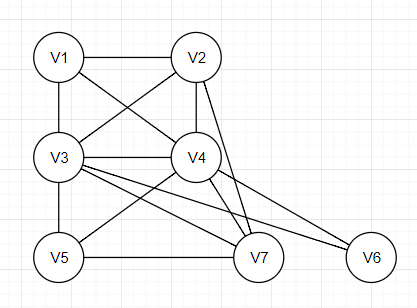
Розв'язати на графах наступні задачі:

1. Виконати наступні операції над графами:

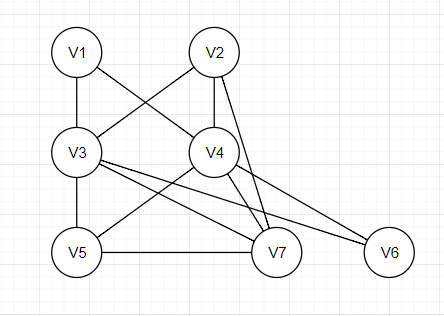
1) знайти доповнення до першого графу



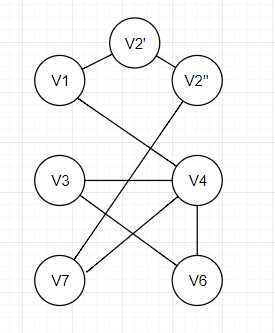
2) об’єднання графів



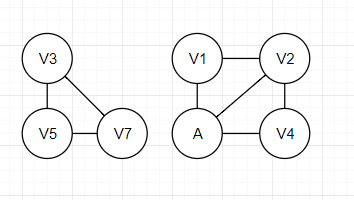
3) кільцеву суму G1 та G2 (G1+G2)



4) розщепити вершину у другому графі

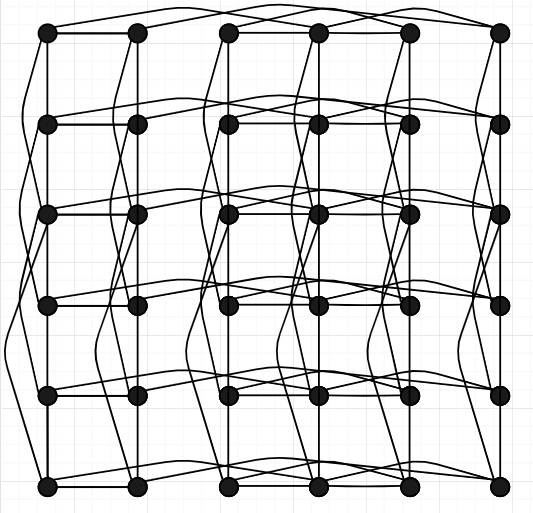


5) виділити підграф А, що складається з 3-х вершин в G1 і знайти стягнення А в G1 (G1\ A)

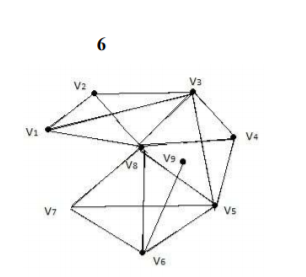


Підграф А

6) добуток графів



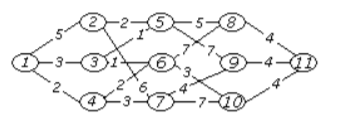
2. Знайти таблицю суміжності та діаметр графа:





Діаметр графа - 3.

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



Метод Краскала:

Вибираємо ребра з найменшою вагою:

3 – 5, 3 – 6 і додаємо їх до дерева,

після бачимо, що найменша вага з-поміж усіх ребер 2.

Шукаємо їх:

1 – 4, 4 – 6, 2 – 5 і додаємо їх до дерева,

Після бачимо, що найменша вага з-поміж усіх ребер .

Шукаємо їх:

1 – 4, але його не додаємо до дерева, тому що утвориться цикл,

4 – 7, 6 – 10, і додаємо їх до дерева.

Після бачимо, що найменша вага з-поміж усіх ребер 4.

Шукаємо їх:

7 – 9, 10 – 11 і додаємо їх до дерева.

Дерево:

3 – 5, 3 – 6, 1 – 4, 4 – 6, 2 – 5, 4 – 7, 6 – 10, 7 – 9, 10 – 11.

Метод Прима:

Сортуємо ребра за вагою:

3 – 6, w = 1;

3 – 5, w = 1;

1 – 4, w = 2;

2 – 5, w = 2;

4 – 6, w = 2;

4 – 7, w = 3;

1 – 3, w = 3;

6 – 10, w = 3;

7 – 9, w = 4;

8 – 11, w = 4;

9 – 11, w = 4;

10 – 11, w = 4;

1 – 2, w = 5;

5 – 8, w = 5;

2 – 7, w = 6;

5 – 9, w = 7;

6 – 8, w = 7;

7 – 10, w = 7;

Додамо ребро 3 – 5 до дерева, бо воно має найменшу вагу.

Точка 6 найближча до точки 3, тому додаємо її до дерева і утворюється ребро

3 – 6.

Точка 4 найближча до точки 6, і точка 2 найближча до точки 5, додамо їх і утворяться ребра 2 – 5 і 4 – 6.

Точка 1 найближча до точки 4, додамо її і утворить ребро 1 – 4.

Точка 9 найближча до точки 9, точка 10 найближчка до точки 6, точка 8 найближча до точки 8, точка 11 найближча до точки 10, утворяться відповідні ребра.

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



Програма :

1. #include <stdio.h>
2. int robytyDereva(int n, int A[n][n]);
3. void VidalimoPovtorenya(int n, int A[n][n]);
4. int vRiznyhDerevah(int n, int A[n][n], int first, int second);
5. void DodatyDoDereva(int n, int A[n][n], int first, int second);
6. int main()
7. {
8. // матриця придатності нашого графа (з вагою)
9. // 1 2 3 4 5 6 7 8 9 10 11
10. int A[11][11] = {
11. /\*1\*/ { 0, 7, 2, 1, 0, 0, 0, 0, 0, 0, 0 },
12. /\*2\*/ { 7, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0 },
13. /\*3\*/ { 2, 0, 0, 0, 7, 4, 0, 0, 0, 0, 0 },
14. /\*4\*/ { 1, 0, 0, 0, 0, 3, 5, 0, 0, 0, 0 },
15. /\*5\*/ { 0, 2, 7, 0, 0, 0, 0, 4, 5, 0, 0 },
16. /\*6\*/ { 0, 0, 4, 3, 0, 0, 0, 6, 0, 2, 0 },
17. /\*7\*/ { 0, 7, 0, 5, 0, 0, 0, 0, 3, 7, 0 },
18. /\*8\*/ { 0, 0, 0, 0, 4, 6, 0, 0, 0, 0, 3 },
19. /\*9\*/ { 0, 0, 0, 0, 5, 0, 3, 0, 0, 0, 4 },
20. /\*10\*/ { 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 4 },
21. /\*11\*/ { 0, 0, 0, 0, 0, 0, 0, 3, 4, 4, 0 }
22. };
24. VidalimoPovtorenya(11, A);
26. /\* Друк листів, сортованих по вазі
27. \*/
28. printf("\nVerticles sorted by weight::");
30. // маса, 7 максимальна маса
31. for (int i = 1; i <= 7; i++)
32. {
33. printf("\n%d: ", i);
34. // перший край
35. for (int j = 1; j <= 11; j++)
36. {
37. // другий край
38. for (int k = 1; k <= 11; k++)
39. {
40. if (A[j - 1][k - 1] == i)
41. {
42. printf("%d-%d; ", j, k);
43. }
44. }
45. }
46. }
48. /\*
49. Перевіряє сортовані вертули і додає один до нашого шляху лише в тому випадку, якщо два краї є різними деревами
50. \*/
52. int B[11][11];
53. robytyDereva(11, B);
55. printf("\n\nOur path: ");
56. // маса, 7 максимальна маса
57. for (int i = 1; i <= 7; i++)
58. {
59. // перший край
60. for (int j = 1; j <= 11; j++)
61. {
62. // другий край
63. for (int k = 1; k <= 11; k++)
64. {
65. if (A[j - 1][k - 1] == i && vRiznyhDerevah(11, B, j, k))
66. {
67. DodatyDoDereva(11, B, j, k);
68. printf("%d-%d; ", j, k);
69. }
70. }
71. }
72. }
73. printf("\n\n");
75. return 0;
76. }
77. int robytyDereva(int n, int A[n][n])
78. {
79. for (int i = 0; i < n; i++)
80. {
81. for (int j = 0; j < n; j++)
82. {
83. A[i][j] = 0;
84. }
85. }
86. for (int i = 0; i < n; i++)
87. {
88. A[i][i] = i + 1;
89. }
91. return A[n][n];
92. }
93. void VidalimoPovtorenya(int n, int A[n][n])
94. {
95. for (int i = 0; i < n; i++)
96. {
97. for (int j = 0; j < n; j++)
98. {
99. if (j < i)
100. {
101. A[i][j] = 0;
102. }
103. }
104. }
105. }
106. int vRiznyhDerevah(int n, int A[n][n], int first, int second)
107. {
108. int temp1;
109. int temp2;
111. // лінія
112. for (int i = 0; i < n; i++)
113. {
114. temp1 = 0;
115. temp2 = 0;
116. // перший елемент
117. for (int j = 0; j < n; j++)
118. {
119. if (A[i][j] == first)
120. {
121. temp1 = 1;
122. }
123. }
124. // другий елемент
125. for (int k = 0; k < n; k++)
126. {
127. if (A[i][k] == second)
128. {
129. temp2 = 1;
130. }
131. }
133. if (temp1 && temp2)
134. {
135. return 0;
136. }
137. }
138. return 1;
139. }
140. void DodatyDoDereva(int n, int A[n][n], int first, int second)
141. {
142. int scndLine;
143. for (int i = 0; i < n; i++)
144. {
145. for (int j = 0; j < n; j++)
146. {
147. if (A[i][j] == second)
148. {
149. scndLine = i;
150. }
151. }
152. }
154. for (int i = 0; i < n; i++)
155. {
156. for (int j = 0; j < n; j++)
157. {
158. if (A[i][j] == first)
159. {
160. for (int k = 0; k < n; k++)
161. {
162. if (A[scndLine][k])
163. {
164. A[i][k] = A[scndLine][k];
165. A[scndLine][k] = 0;
166. }
167. }
168. }
169. }
170. }
171. }

Результат :

Verticles sorted by weight:

1: 1-4; 2-7;

2: 1-3; 2-5; 6-10;

3: 4-6; 7-9; 8-11;

4: 3-6; 5-8; 9-11; 10-11;

5: 4-7; 5-9;

6: 6-8;

7: 1-2; 3-5; 7-10;

Our path: 1-4; 2-7; 1-3; 2-5; 6-10; 4-6; 7-9; 8-11; 5-8; 10-11;