

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ В
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА
ПОЛІТЕХНІКА”**

Кафедра систем штучного інтелекту

Лабораторна робота №5
з дисципліни
«Дискретна математика»

Виконав:
студент групи КН-115
Гончаренко Н.
Викладач:
Мельникова Н.І.

Львів – 2019 р.

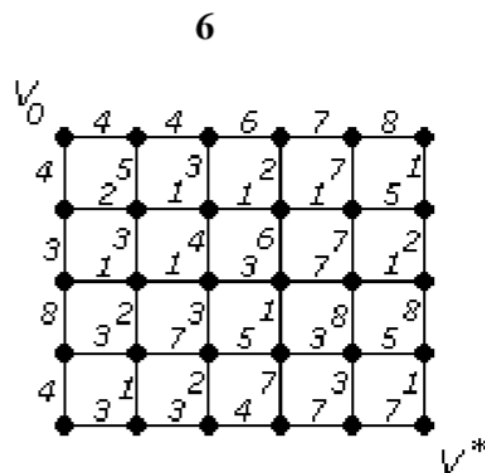
Лабораторна робота № 5.

Тема: Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи

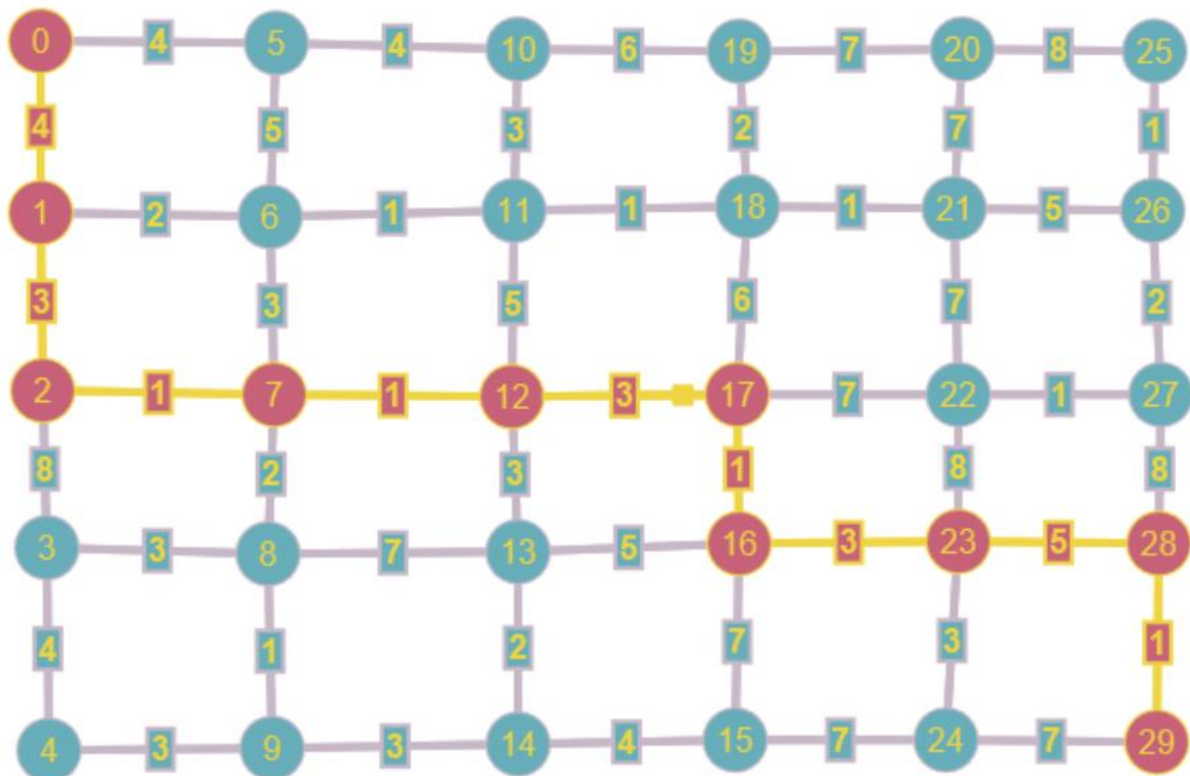
Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

Варіант 6.

Завдання № 1. Розв'язати на графах наступні 2 задачі: 1. За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі поміж парою вершин V_0 і V^* .

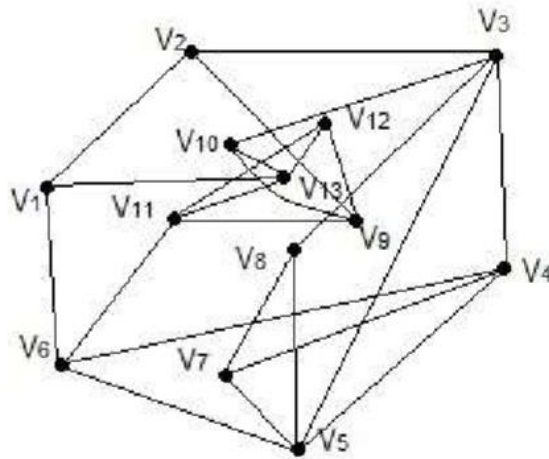


Пройшовши усі вершини за алгоритмом отримаємо:

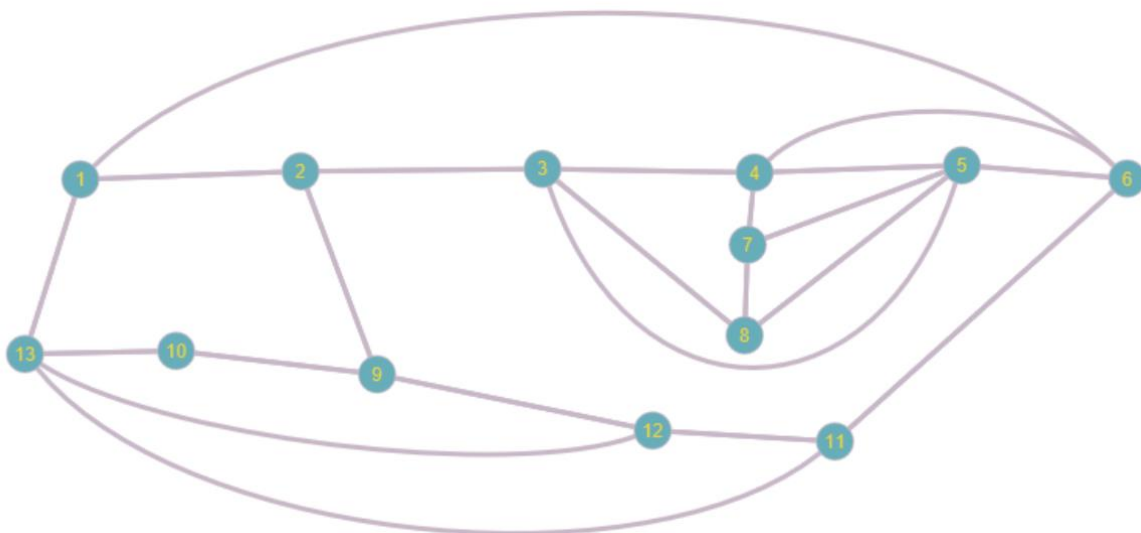


2. За допомогою у-алгоритма зробити укладку графа у площині, або довести що вона неможлива.

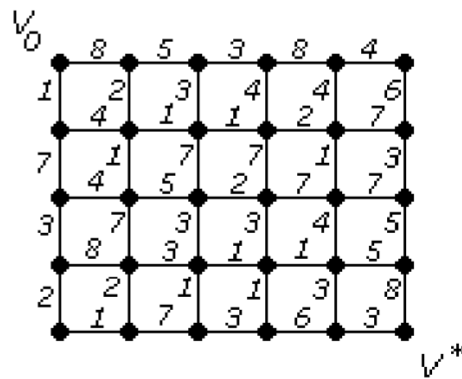
6



Скориставшись алгоритмом отримаємо:



Завдання №2. Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту.



Реалізуємо програмний код.

```

1. #include <cmath>
2. #include <iostream>
3.
4. using namespace std;
5.
6.
7.
8. void output(int i, int* par, int k);
9.
10.
11. const int N = 30;
12.
13.
14. int matrix[N][N] = {
15. 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
16. 8, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
17. 0, 5, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
18. 0, 0, 3, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
19. 0, 0, 0, 8, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
20. 0, 0, 0, 0, 4, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
21. 0, 0, 0, 0, 0, 6, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0,
22. 0, 0, 0, 0, 0, 0, 3, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0,
23. 0, 0, 0, 0, 0, 0, 0, 5, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0,
24. 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
25. 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
26. 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
27. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
28. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
29. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
30. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8,
31. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0,
32. 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
33. 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
34. 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0,
35. 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0,
36. 0, 0, 0, 0, 4, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
37. 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 4, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0,
38. 0, 0, 0, 0, 0, 0, 0, 5, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
39. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0,
40. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 7, 0, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
41. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 2, 0, 3, 5, 0, 0, 0, 0, 0,
42. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 0, 0, 3, 0, 0, 0, 0, 0,
43. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 0, 5, 0, 0, 7, 0, 0, 0, 0,
44. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 7, 0, 0, 0, 0,
45. };
46.
47. int cost[N][N] = {

```

```

48. 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
49. 8, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
50. 0, 5, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
51. 0, 0, 3, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
52. 0, 0, 0, 8, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
53. 0, 0, 0, 0, 4, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
54. 0, 0, 0, 0, 0, 6, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
55. 0, 0, 0, 0, 0, 0, 3, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
56. 0, 0, 0, 0, 0, 0, 0, 5, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0,
57. 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
58. 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0,
59. 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
60. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
61. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
62. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
63. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8,
64. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0,
65. 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
66. 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
67. 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0,
68. 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
69. 0, 0, 0, 0, 4, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
70. 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 4, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
71. 0, 0, 0, 0, 0, 0, 0, 5, 0, 3, 0, 0, 0, 0, 0, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
72. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
73. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 7, 0, 3, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
74. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 2, 0, 3, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0,
75. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 3,
76. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 0, 5, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0,
77. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 7, 0, 0, 0, 0, 0, 0, 0,
78. };
79.
80.
81. int dist[N];
82.
83. int parent[N];
84. void dijkstra(int start, int end)
85. {
86.
87.     bool in_tree[N] = { false };
88.
89.     for (int i = 0; i < N; i++)
90.         dist[i] = INT_MAX;
91.
92.
93.     dist[start] = 0;
94.
95.     int cur = start;
96.
97.
98.     while (!in_tree[cur])
99.     {
100.         in_tree[cur] = true;
101.
102.         for (int i = 0; i < N; i++)
103.         {
104.             if (matrix[cur][i] != 0)
105.             {
106.
107.                 int d = dist[cur] + cost[cur][i];
108.
109.                 if (d < dist[i])
110.                 {
111.                     dist[i] = d;
112.                     parent[i] = cur;
113.                 }
114.             }
115.         }
116.

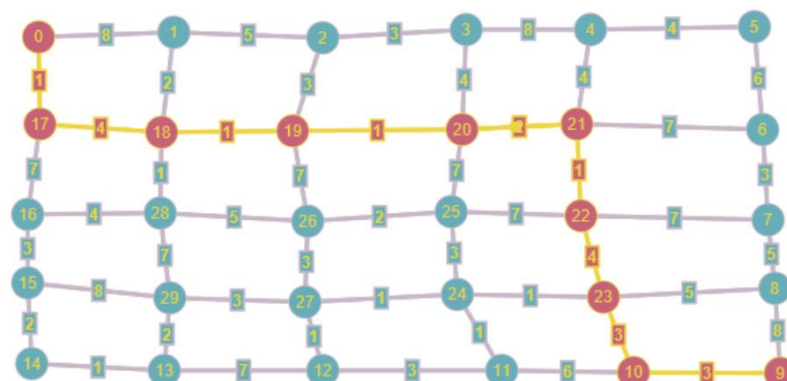
```

```

117.
118.     int min_dist = INT_MAX;
119.     for (int i = 0; i < N; i++)
120.     {
121.         if (!in_tree[i] && dist[i] < min_dist)
122.         {
123.             cur = i;
124.             min_dist = dist[i];
125.         }
126.     }
127. }
128. output(start, parent, end);
129. cout << end << "\nWeight: " << dist[end];
130.
131.}
132.
133.
134. void output(int k, int* par, int i)
135. {
136.     if (k == i)
137.     {
138.         return;
139.     }
140.     else
141.     {
142.         output(k, par, par[i]);
143.         cout << par[i] << " --> ";
144.         return;
145.     }
146. }
147.
148. int main()
149. {
150.     dijkstra(0 , 9);
151.
152.
153. }

```

Результат виконання програми:



Висновок: я набув практичних вмінь та навичок з використання алгоритму Дейкстри.