

РОЗРАХУНКОВА-ГРАФІЧНА РОБОТА
з дисципліни
“ДИСКРЕТНА МАТЕМАТИКА”

Виконав: Гончаренко Назар

Студент групи КН-115

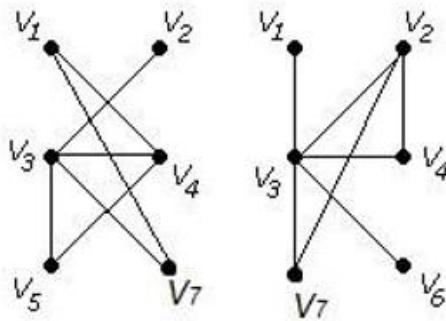
Варіант №12

Варіант №12

Завдання № 1 Виконати наступні операції над графами:

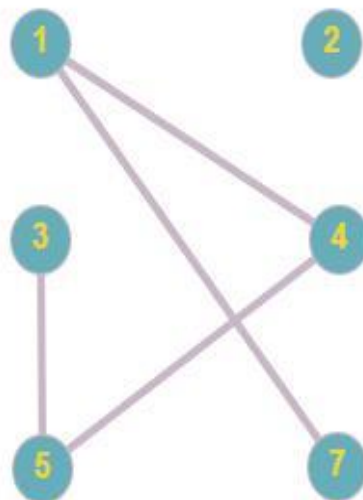
1) знайти доповнення до першого графу, 2) об'єднання графів, 3) кільцеву сумму $G1$ та $G2$ ($G1+G2$), 4) розмножити вершину у другому графі, 5) виділити підграф A - що складається з 3-х вершин в $G1$ 6) добуток графів.

12)

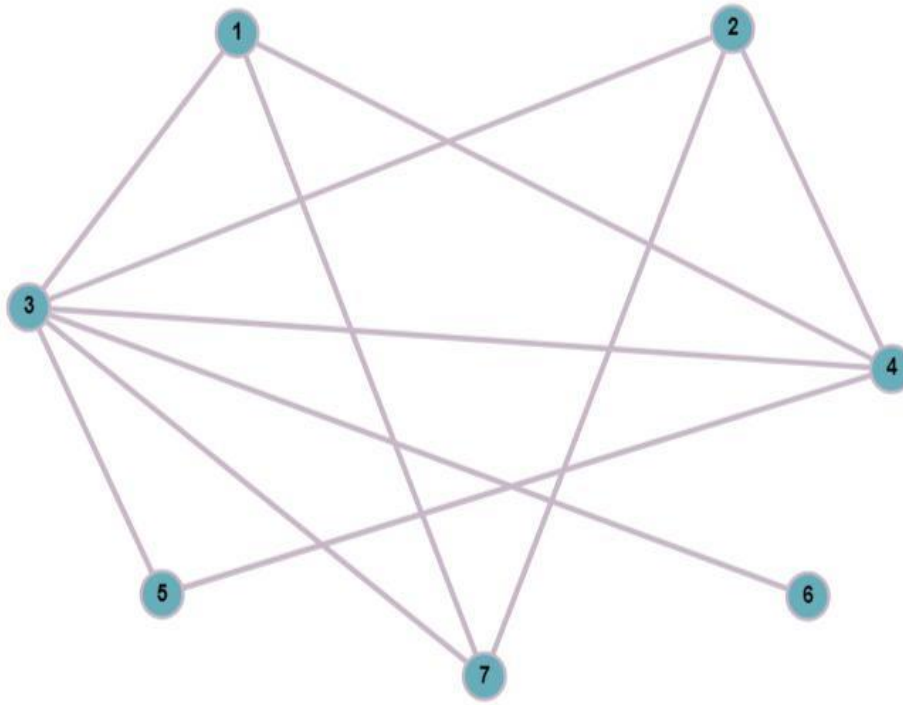


1)Доповнення

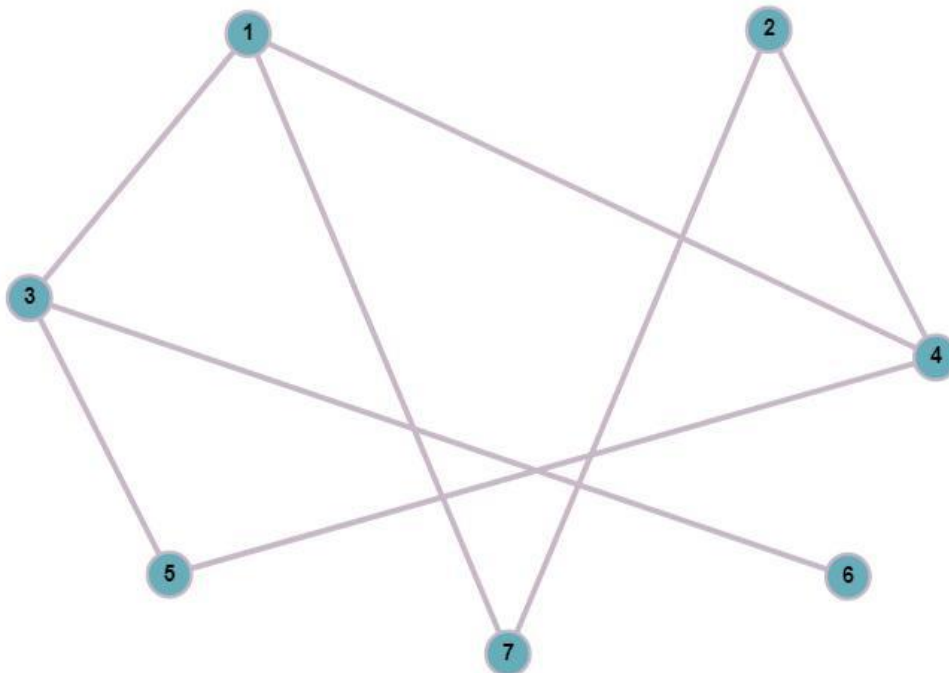
$G1 \setminus G2$:



2)

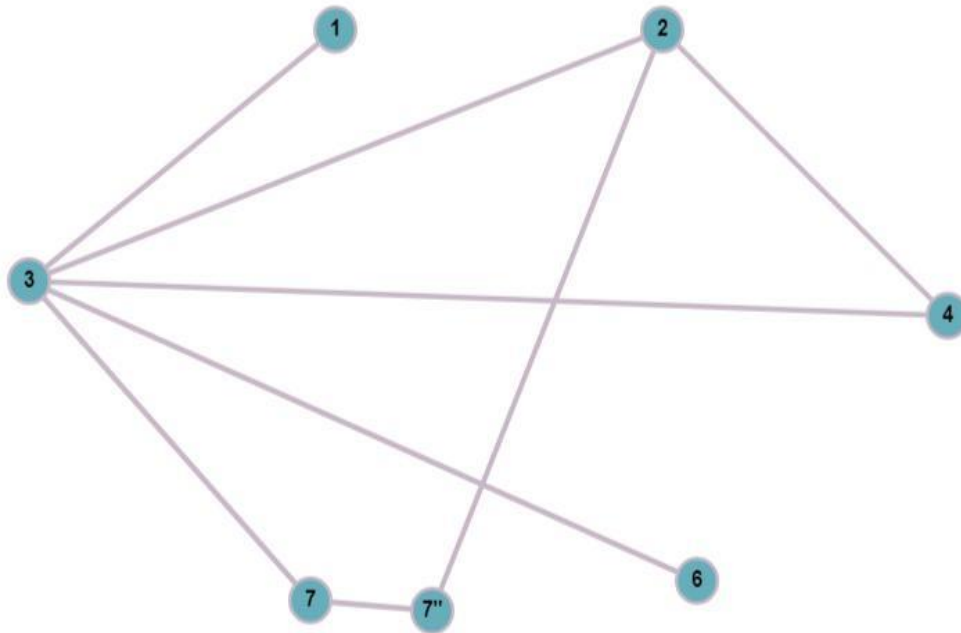


3) Кільцева сума:



4) Розщепити вершину у другому графі:

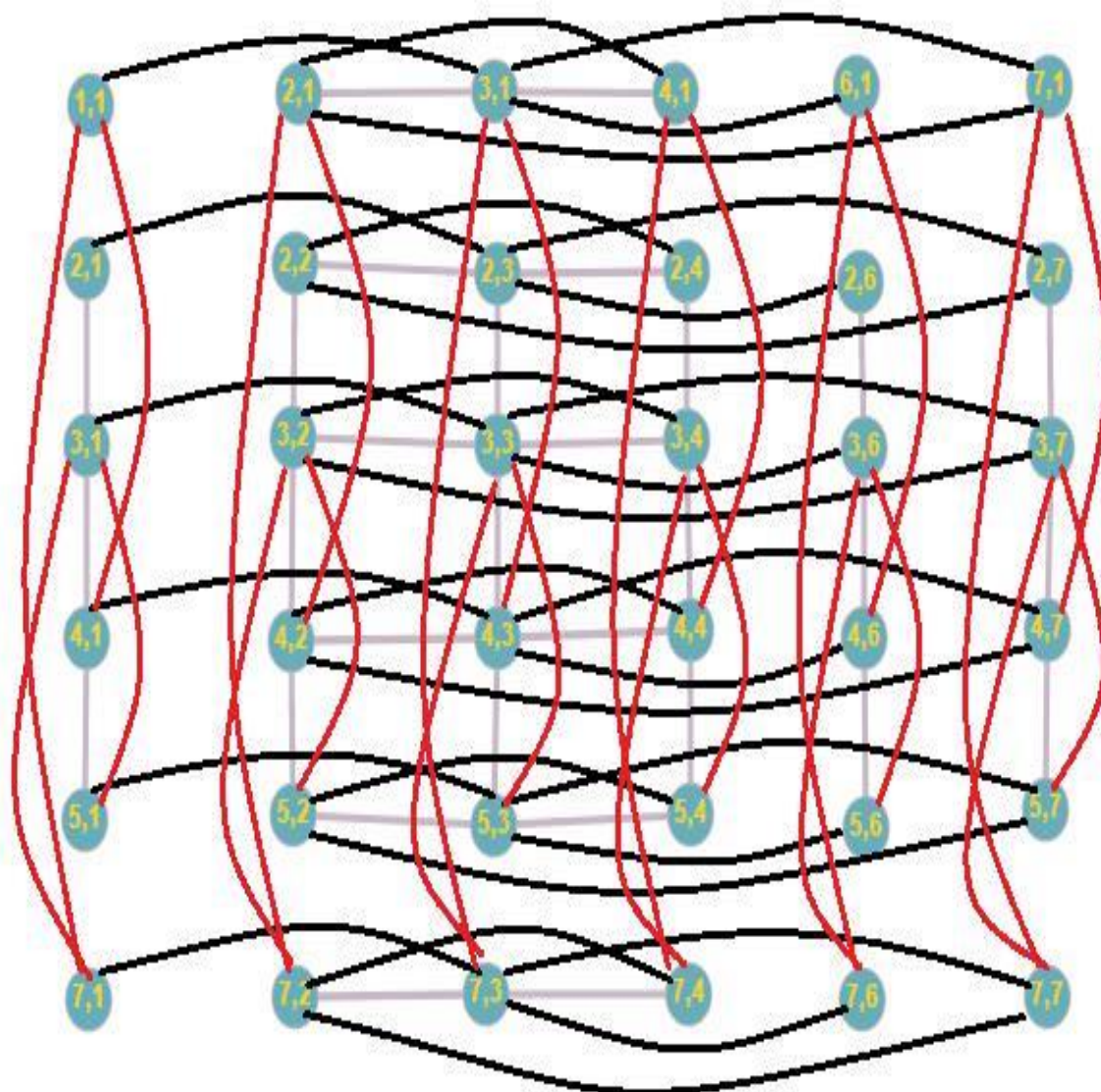
Розщепимо вершину 7



5) Виділити підграф А, що складається з $V=\{1,3,7\}$ в G_1 і знайти стягнення А в G_1



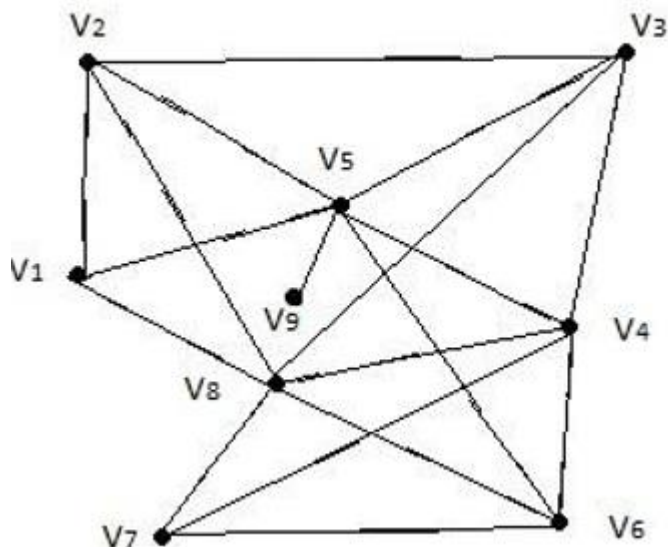
6) Добуток графів



Завдання № 2

Скласти таблицю суміжності для орграфа.

12)



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

Завдання № 3

Для графа з другого завдання знайти діаметр.

Діаметр = 3.

Завдання № 4

Для графа з другого завдання виконати обхід дерева вшир (закінчується на парне число).

№	Додавання вершини	Черга
1	V1	V1
2	V2	V1,V2
3	V5	V1,V2,V5
4	V8	V1,V2,V5,V8
5	-	V2,V5,V8
6	V3	V2,V5,V8,V3
7	-	V2,V5,V8
8	V9	V5,V8,V3,V9
9	V4	V5,V8,V3,V9,V4
10	V6	V5,V8,V3,V9,V4,V6
11	-	V8,V3,V9,V4,V6
12	V7	V8,V3,V9,V4,V6,V7
13	-	V3,V9,V4,V6,V7
14	-	V9,V4,V6,V7
15	-	V4,V6,V7
16	-	V6,V7
17	-	V7
18	-	-
19		

```

1.  #include <iostream>
2.  #include <queue>
3.  using namespace std;
4.  int main() {
5.      int n, v;
6.      cin >> n >> v;
7.      int** matrix = new int* [n];
8.      for (int i = 0; i < n; i++)
9.      {
10.         matrix[i] = new int[n];
11.     }
12.     for (int i = 0; i < n; i++)
13.         for (int j = 0; j < n; j++)
14.             cin >> matrix[i][j];
15.     queue <int> plan;
16.     plan.push(--v);
17.     matrix[v][v] = 1;
18.     int counter = 1;
19.     while (!plan.empty()) {
20.         v = plan.front();
21.         plan.pop();
22.         for (int u = 0; u < n; u++) {

```

```

23.     if (matrix[v][u] and !matrix[u][u]) {
24.         plan.push(u);
25.         matrix[u][u] = 1;
26.         counter++;
27.         cout << plan.back()+1 << endl;
28.     }
29. }
30. }
31. }

```

```

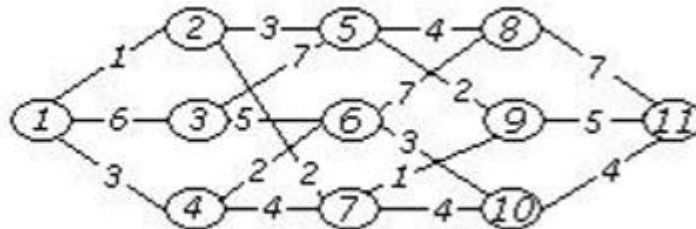
9
1
0 1 0 0 1 0 0 1 0
1 0 1 0 1 0 0 1 0
0 1 0 1 1 0 0 1 0
0 0 1 0 1 1 1 1 0
1 1 1 1 0 1 0 0 1
0 0 0 1 1 0 1 1 0
0 0 0 1 0 1 0 1 0
1 1 1 1 0 1 1 0 0
0 0 0 0 1 0 0 0 0
2
5
8
3
4
6
9
7

```

Завдання № 5

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

12)



Алгоритм Прима:

```

1.  #include <iostream>
2.  #include <cstring>
3.  using namespace std;
4.
5.  #define INF 9999999
6.
7.  // number of vertices in graph
8.  #define V 11
9.
10. // create a 2d array of size 5x5
11. //for adjacency matrix to represent graph
12.
13. int G[V][V] = {
14.     {0, 1, 6, 3, 0, 0, 0, 0, 0, 0, 0},
15.     {1, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0},
16.     {6, 0, 0, 0, 7, 5, 0, 0, 0, 0, 0},
17.     {3, 0, 0, 0, 0, 2, 4, 0, 0, 0, 0},
18.     {0, 3, 7, 0, 0, 0, 0, 4, 2, 0, 0},
19.     {0, 0, 5, 2, 0, 0, 0, 7, 0, 3, 0},
20.     {0, 2, 0, 4, 0, 0, 0, 0, 1, 4, 0},

```



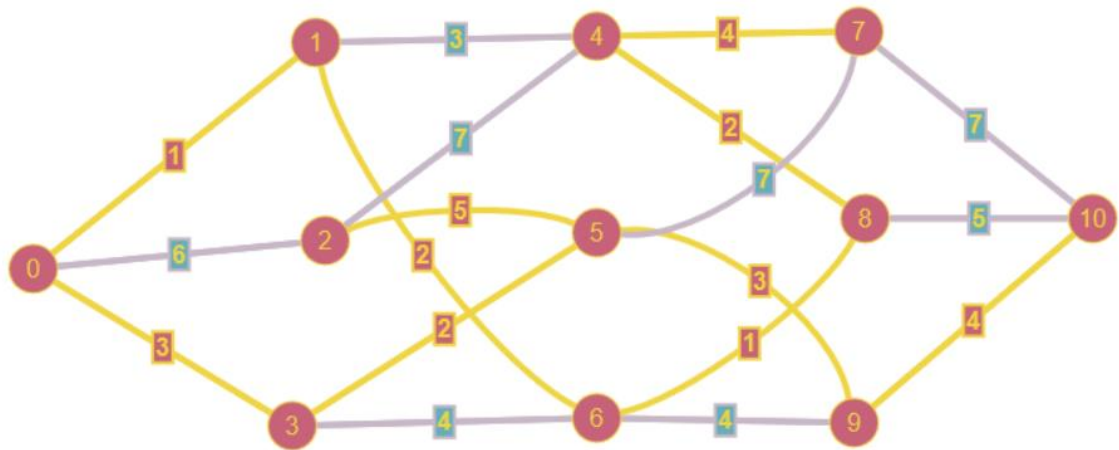
```

21. {0, 0, 0, 0, 4, 7, 0, 0, 0, 0, 7},
22. {0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 5},
23. {0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 4},
24. {0, 0, 0, 0, 0, 0, 0, 7, 5, 4, 0},
25. };
26.
27. int main() {
28.
29.     int no_edge;
30.
31.
32.     int selected[V];
33.
34.
35.     memset(selected, false, sizeof(selected));
36.
37.     no_edge = 0;
38.
39.
40.
41.     selected[0] = true;
42.
43.     int x;
44.     int y;
45.
46.
47.     cout << "Edge" << " : " << "Weight";
48.     cout << endl;
49.     while (no_edge < V - 1) {
50.
51.
52.         int min = INF;
53.         x = 0;
54.         y = 0;
55.
56.         for (int i = 0; i < V; i++) {
57.             if (selected[i]) {
58.                 for (int j = 0; j < V; j++) {
59.                     if (!selected[j] && G[i][j]) {
60.                         if (min > G[i][j]) {
61.                             min = G[i][j];
62.                             x = i;
63.                             y = j;
64.                         }
65.
66.                     }
67.                 }
68.             }
69.         }
70.         cout << x << " - " << y << " : " << G[x][y];
71.         cout << endl;
72.         selected[y] = true;
73.         no_edge++;
74.     }
75.
76.     return 0;
77. }

```

Edge : Weight

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



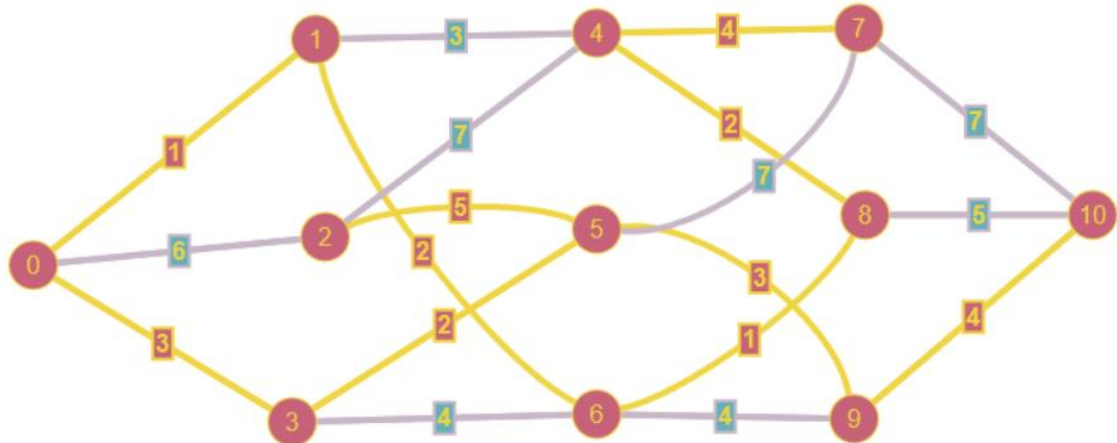
Алгоритм Крассака:

```
1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4.
5. using namespace std;
6. int main()
7. {
8.     int m, n;
9.     cin >> m >> n;
10.    vector < pair < int, pair<int, int> > > g;
11.    for (int i = 0; i < m; i++)
12.    {
13.        int a1, a2, v;
14.        cin >> a1 >> a2 >> v;
15.        g[i] = make_pair(v, make_pair(a1, a2));
16.    }
17.
18.    int cost = 0;
19.    vector < pair<int, int> > res;
20.
21.    sort(g.begin(), g.end());
22.    vector<int> tree_id(n);
23.    for (int i = 0; i < n; ++i)
24.        tree_id[i] = i;
25.    for (int i = 0; i < m; ++i)
```

```

26. {
27.     int a = g[i].second.first, b = g[i].second.second, l = g[i].first;
28.     if (tree_id[a] != tree_id[b])
29.     {
30.         cost += l;
31.         res.push_back(make_pair(a, b));
32.         int old_id = tree_id[b], new_id = tree_id[a];
33.         for (int j = 0; j < n; ++j)
34.             if (tree_id[j] == old_id)
35.                 tree_id[j] = new_id;
36.     }
37. }
38. }

```



Завдання № 6

Розв'язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

12)

	1	2	3	4	5	6	7	8
1	∞	5	6	5	4	4	5	5
2	5	∞	1	5	1	1	1	1
3	6	1	∞	1	1	3	2	1
4	5	5	1	∞	5	5	7	5
5	4	1	1	5	∞	3	2	5
6	4	1	3	5	3	∞	5	6
7	5	1	2	7	2	5	∞	1
8	5	1	1	5	5	6	1	∞

```

1. #include <iostream>

```

```

2. #define min(x, y) x < y ? x : y
3. using namespace std;
4. const int inf = 1E9, NMAX = 16;
5. int n, i, j, k, m, temp, ans, d[NMAX][NMAX], t[1 << NMAX][NMAX];
6. bool get(int nmb, int x)
7. {
8.     return (x & (1 << nmb)) != 0;
9. }
10. int main()
11. {
12.     cin >> n;
13.     for (i = 0; i < n; ++i)
14.         for (j = 0; j < n; ++j) cin >> d[i][j];
15.     t[1][0] = 0; m = 1 << n;
16.     for (i = 1; i < m; i += 2)
17.         for (j = (i == 1) ? 1 : 0; j < n; ++j)
18.         {
19.             t[i][j] = inf;
20.             if (j > 0 && get(j, i))
21.             {
22.                 temp = i ^ (1 << j);
23.                 for (k = 0; k < n; ++k)
24.                     if (get(k, i) && d[k][j] > 0) t[i][j] = min(t[i][j], t[temp][k] + d[k][j]);
25.             }
26.         }
27.     for (j = 1, ans = inf; j < n; ++j)
28.         if (d[j][0] > 0) ans = min(ans, t[m - 1][j] + d[j][0]);
29.     if (ans == inf) cout << -1; else cout << ans;
30. }

```

```

8
100000 5 6 5 4 4 5 5
5 100000 1 5 1 1 1 1
6 1 100000 1 1 3 2 1
5 5 1 100000 5 5 7 5
4 1 1 5 100000 3 2 5
4 1 3 5 3 100000 5 6
5 1 2 7 2 5 100000 1
5 1 1 5 5 6 1 100000
16

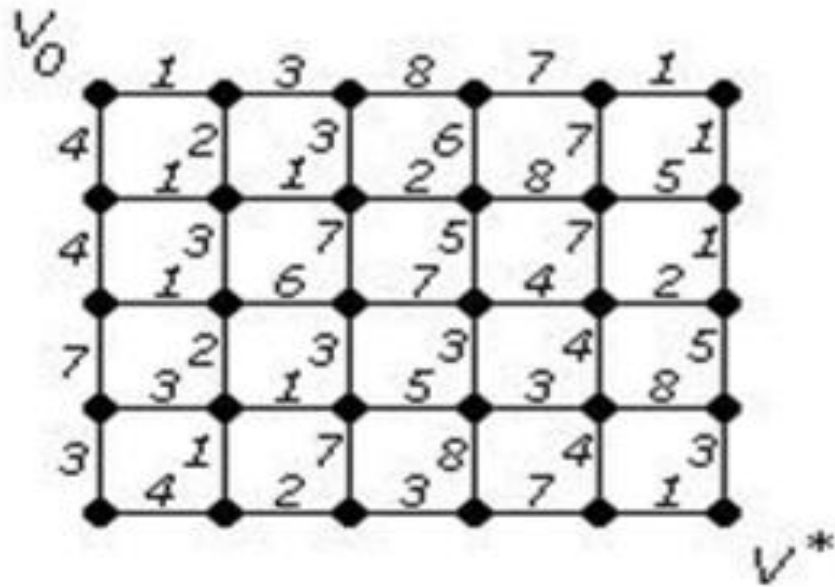
```

Відповідь 16.

Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V_0 і V^* .

12)



```

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, 4, 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. 4, 0, 4, 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,
17. 0, 4, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
18. 0, 0, 7, 0, 3, 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,
19. 0, 0, 0, 3, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
20. 0, 0, 0, 0, 4, 0, 2, 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, 2, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0,
22. 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, 0, 0, 0, 8, 0,
23. 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, 4, 0, 0, 0,
24. 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 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, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0,
26. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,

```

```

27. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
28. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
29. 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
30. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 8, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0,
    0,
31. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 7, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0,
    0,
32. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 7, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
33. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 7, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0,
    0,
34. 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0, 3, 0, 0, 0, 0, 0,
    0,
35. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0, 0, 2, 0, 7, 0, 0, 0, 0, 0,
    0,
36. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 4, 4, 0, 0,
    0,
37. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 8, 0, 2, 0, 0, 0, 0, 5, 0, 0, 0, 0,
    0,
38. 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0,
    2,
39. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 6, 0, 7, 0, 0, 3, 0,
    0,
40. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 5, 0, 7, 0, 0, 3, 0, 0,
    0,
41. 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 3, 0, 0, 0,
    0,
42. 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 0, 5, 0,
    0,
43. 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 5, 0, 0,
    1,
44. 0, 0, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0,
    0,
45. };
46.
47. int cost[N][N] = {
48. 0, 4, 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,
    0,
49. 4, 0, 4, 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,
50. 0, 4, 0, 7, 0, 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,
51. 0, 0, 7, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    3,
52. 0, 0, 0, 3, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
53. 0, 0, 0, 0, 4, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1,
54. 0, 0, 0, 0, 0, 2, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7,
    0,
55. 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, 0, 0, 8, 0,
    0,
56. 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, 4, 0, 0, 0,
    0,
57. 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 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, 0, 3, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0,
    0,
59. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
60. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
61. 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,
62. 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0,

```

```

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

```

```

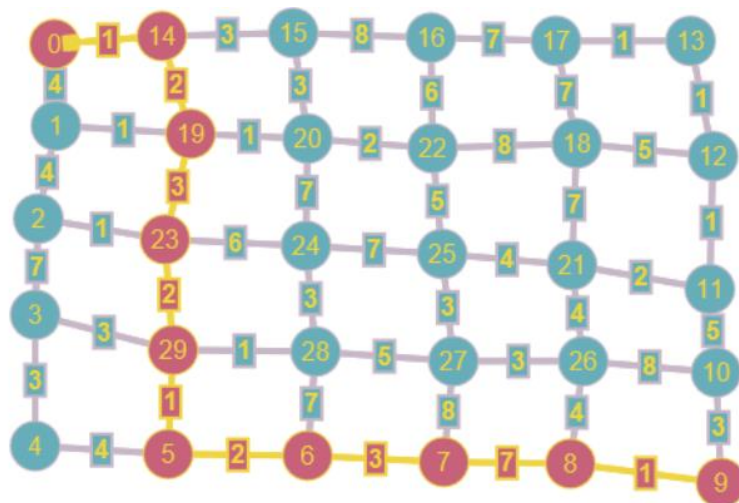
117.     }
118.
119.
120.     int min_dist = INT_MAX;
121.     for (int i = 0; i < N; i++)
122.     {
123.         if (!lin_tree[i] && dist[i] < min_dist)
124.         {
125.             cur = i;
126.             min_dist = dist[i];
127.         }
128.     }
129. }
130. output(start, parent, end);
131. cout << end << "\nWeight: " << dist[end];
132.
133.}
134.
135.
136.void output(int k, int* par, int i)
137.{
138.    if (k == i)
139.    {
140.        return;
141.    }
142.    else
143.    {
144.        output(k, par, par[i]);
145.        cout << par[i] << " --> ";
146.        return;
147.    }
148.}
149.
150.int main()
151.{
152.    deicstra(0 , 9);
153.
154.
155.}

```

```

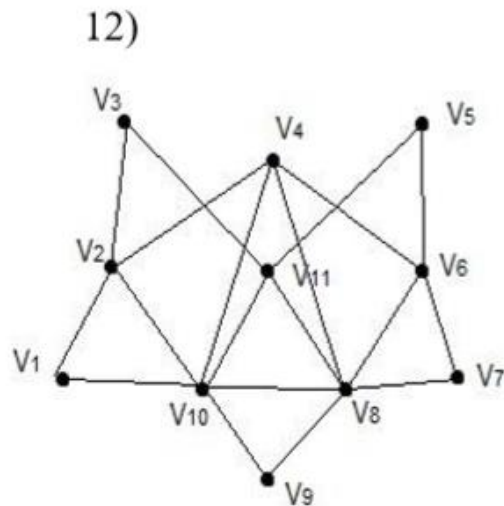
0 --> 14 --> 19 --> 23 --> 29 --> 5 --> 6 --> 7 --> 8 --> 9
Weight: 22

```



Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.



Эйлеровий цикл:

```
1. #include<iostream>
2. #include<vector>
3. #define NODE 11
4. using namespace std;
5. int graph[NODE][NODE] = {
6.     {0,1,0,0,0,0,0,0,0,1,0},
7.     {1,0,1,1,0,0,0,0,0,1,0},
8.     {0,1,0,0,0,0,0,0,0,0,1},
9.     {0,1,0,0,0,1,0,1,0,1,0},
10.    {0,0,0,0,0,1,0,0,0,0,1},
11.    {0,0,0,1,1,0,1,1,0,0,0},
12.    {0,0,0,0,0,1,0,1,0,0,0},
13.    {0,0,0,1,0,1,1,0,1,1,1},
14.    {0,0,0,0,0,0,0,1,0,1,0},
15.    {1,1,0,1,0,0,0,1,1,0,1},
16.    {0,0,1,0,1,0,0,1,0,1,0},
17. };
18. int tempGraph[NODE][NODE];
19. int findStartVert() {
20.     for (int i = 1; i < NODE; i++) {
21.         int deg = 0;
22.         for (int j = 0; j < NODE; j++) {
23.             if (tempGraph[i][j])
24.                 deg++;
25.         }
26.         if (deg % 2 != 0)
27.             return i;
28.     }
29.     return 0;
30. }
31. bool isBridge(int u, int v) {
32.     int deg = 0;
33.     for (int i = 0; i < NODE; i++)
34.         if (tempGraph[v][i])
35.             deg++;
36.     if (deg > 1) {
```

```

37.     return false;
38. }
39. return true;
40. }
41. int edgeCount() {
42.     int count = 0;
43.     for (int i = 0; i < NODE; i++)
44.         for (int j = i; j < NODE; j++)
45.             if (tempGraph[i][j])
46.                 count++;
47.     return count;
48. }
49. void fleuryAlgorithm(int start) {
50.     static int edge = edgeCount();
51.     for (int v = 0; v < NODE; v++) {
52.         if (tempGraph[start][v]) {
53.             if (edge <= 1 || !isBridge(start, v)) {
54.                 cout << start + 1 << "--" << v + 1 << " ";
55.                 tempGraph[start][v] = tempGraph[v][start] = 0;
56.                 edge--;
57.                 fleuryAlgorithm(v);
58.             }
59.         }
60.     }
61. }
62. int main() {
63.     for (int i = 0; i < NODE; i++)
64.         for (int j = 0; j < NODE; j++)
65.             tempGraph[i][j] = graph[i][j];
66.     cout << "Euler Path Or Circuit: ";
67.     fleuryAlgorithm(findStartVert());
68. }

```

```
Euler Path Or Circuit: 1--2 2--3 3--11 11--5 5--6 6--4 4--2 2--10 10--4 4--8 8--6 6--7 7--8 8--9 9--10 10--8 8--11 11--10 10--1
```

**1—2, 2—3, 3—11, 11—5, 5—6, 6—4, 4—2, 2—10, 10—4, 4—8, 8—6,
6—7, 7—8, 8—9, 9—10, 10—8, 8—11, 11—10, 10—1.**

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

12. $\bar{x}y \vee x\bar{y}\bar{z}$

Відповідь: $\bar{x}y \vee x\bar{y}\bar{z}$.