МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконала:

студентка групи КН-112

Казьоннікова Ніколетта

Викладач:

Мельникова Н.І.

Львів – 2019 р.

Напишіть алгоритм

*Обхід графа **вглиб** або вшир*

```
#include <iostream>
#include <stack>
#include <vector>
using namespace std;
int n, v;
vector <vector <int> > M;
vector <int> visited;
vector <int> colours;
vector <int> cyclicnodes;
bool dfs(int v)
{
   colours[v] = 2;
   visited[v]++;
   for (int i = 0; i < M[v].size(); i++)
   {
      if (colours[M[v][i]] != 3 and visited[M[v][i]] != 2)
      {
         if (colours[M[v][i]] == 1)
         {
            if (dfs(M[v][i])) return true;
         }
         else
         {
            if (colours[M[v][i]] == 2) return true;
```

```
            }
        }
        if (colours[v] == 3) return false;
        if (visited[v] == 2) return false;
    }
    colours[v] = 3;
    return false;
}
void cyclic(int v)
{
    for (int i = 0; i < M[v].size(); i++)
    {
        cyclicnodes.push_back(M[v][i]);
        if (cyclicnodes[0] == cyclicnodes[cyclicnodes.size() - 1] and
cyclicnodes.size() > 1)
            break;
        cyclic(M[v][i]);
    }
}
int main()
{
    cout << "Enter the value of lines and vertices:" << endl;
    cin >> n >> v;
    M.resize(n);
    visited.resize(n);
    colours.resize(n);
    int v1, v2;
```

```
colours.assign(n, 1);
visited.assign(n, 0);
cout << "Enter vertices: " << endl;
for (int i = 0; i < v; i++)
{
   cin >> v1 >> v2;
   M[v1 - 1].push_back(v2 - 1);
}
int t = 0;
for (int i = 0; i < n; i++)
{
   if (dfs(i)) t = 1;
}
if (t == 0) cout << "There are no cycles.";
else
{
   cout << "Result:" << endl;
   vector <int> iscyclic;
   for (int i = 0; i < colours.size(); i++) {
      if (colours[i] == 2) iscyclic.push_back(i);
   }
   cyclic(iscyclic[0]);
   for (int i = 1; i < cyclicnodes.size(); i++)
      cout << cyclicnodes[i] + 1 << " ";
}
return 0;
```

```
}
```

Прима *знаходження найменшого остову.*

```
#include <iostream>
using namespace std;
#define V 5
int minKey(int key[], bool mstSet[])
{
   int min = INT_MAX, min_index;
   for (int v = 0; v < V; v++)
      if (mstSet[v] == false && key[v] < min)
         min = key[v], min_index = v;
   return min_index;
}
void printMST(int parent[], int graph[V][V])
{
   cout << "Edge \tWeight\n";
   for (int i = 1; i < V; i++)
      cout << parent[i] << " - " << i << " \t" << graph[i][parent[i]] << " \n";
}
void primMST(int graph[V][V])
{
   int parent[V];
   int key[V];
   bool mstSet[V];
   for (int i = 0; i < V; i++)
      key[i] = INT_MAX, mstSet[i] = false;
```

```
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < V - 1; count++)
    {
        int u = minKey(key, mstSet);
        mstSet[u] = true;
        for (int v = 0; v < V; v++)
            if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
    printMST(parent, graph);
}
int main()
{
    int graph[V][V] = { { 0, 2, 0, 6, 0 },
            { 2, 0, 3, 8, 5 },
            { 0, 3, 0, 0, 7 },
            { 6, 8, 0, 0, 9 },
            { 0, 5, 7, 9, 0 } };
    primMST(graph);
    return 0;
}
```

Краскала *знаходження найменшого остову.*

```
#include <iostream>
using namespace std;
```

```
const int ver = 11;
const int reb = 18;
void obchod(int v, int matrix[][ver], int weight[], int proydenirebra[]);
int main()
{
   int matrix [ver][ver]  =
   {
      {0,4,3,1,0,0,0,0,0,0,0},
      {4,0,0,0,2,0,2,0,0,0,0},
      {3,0,0,0,7,7,0,0,0,0,0},
      {1,0,0,0,0,2,2,0,0,0,0},
      {0,2,7,0,0,0,0,4,4,0,0},
      {0,0,7,2,0,0,0,4,0,5,0},
      {0,1,0,2,0,0,0,0,3,3,0},
      {0,0,0,0,4,4,0,0,0,0,7},
      {0,0,0,0,4,0,3,0,0,0,6},
      {0,0,0,0,0,5,3,0,0,0,5},
      {0,0,0,0,0,0,0,0,7,6,5,0}
   };
   int weight[] = { 1,2,3,4,5,6,7 };
   int value = sizeof(weight) / 4;
   int proydenirebra[ver];
   for (int i = 0; i < value; i++)
   {
      proydenirebra[i] = 0;
   }
```

```
    int dovzh;
    for (int wei = 0; wei < value; wei++)
    {
        obchod(wei, matrix, weight, proydenirebra);
     }
}
void obchod(int v, int matrix[][ver], int weight[], int proydenirebra[])
{
    int counter1 = 0, counter2 = 0;
    bool flag1=false, flag2=false;
    for (int i = 0; i < ver; i++)
    {
        for (int j = 0; j < ver; j++)
        {
            if (matrix[i][j] == weight[v])
            {
                for (int k = 0; k < ver; k++)
                {
                    if (proydenirebra[k] != i)counter1++;
                    if (proydenirebra[k] != j)counter2++;
                }
            }
            if (counter1 == 11)
            {
                proydenirebra[i] = i;
                flag1 = true;
```

```
        }
        if (counter2 == 11)
        {
           proydenirebra[j] = j;
           flag2 = true;
        }
        if (flag1 == false && flag2 == false) {}
        else
        {
           cout << "\nLine:{" << proydenirebra[i] + 1 << ";" << proydenirebra[j] +
1 << "}\n";
           cout << "Line weight:" << v + 1 << endl;
        }
        counter1 = 0;
        counter2 = 0;
        flag1 = false;
        flag2 = false;
      }
   }
}
```

Дейкстра знаходження найкоротшого ланцюга між парою вершин.

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <cstdio>
#include<sstream>
using namespace std;
```

```
const int number = 30;
const int inf = 1000;
struct vershuna
{
   int weight = inf;
   int from = 0;
   bool fixed = false;
};
int main()
{
   int matrix[number][number];
   string rebro;
   char rebroch[100];
   char* rch;
   vershuna ver[number];
   ver[0].weight = 0;
   ver[0].fixed = true;
   int currver = 0;
   for (int i = 0; i < number; i++)
   {
      for (int j = 0; j < number; j++)
      {
         matrix[i][j] = 0;
      }
   }
   int i = 0; int j = 0; int row = 0;
```

```
fstream file;
file.open("MATRIX.txt", ios_base::in);
while (getline(file, rebro))
{
   strcpy(rebroch, rebro.c_str());
   char* rch = strtok(rebroch, " ");
   while (rch != NULL)
   {
      if (row % 2 == 0)
      {
         matrix[i][i + 1] = (int)*rch - 48;
         matrix[i + 1][i] = (int)*rch - 48;
         i++;
      }
      else
      {
         matrix[j][j + 6] = (int)*rch - 48;
         matrix[j + 6][j] = (int)*rch - 48;
         j++;
      }
      rch = strtok(NULL, " ");
   }
   row++;
   if (row % 2 == 0)i++;
}
for (int i = 0; i < number; i++)
```

```
    {
        for (int j = 0; j < number; j++)
        {
            cout << matrix[i][j] << "  ";   //druk matrix
        }
        cout << endl;
    }
    while (ver[number - 1].fixed != true)
    {
        int conected[number - 1];
        int index = 0;
        for (int i = 0; i < number; i++)
        {
            if (matrix[currver][i] != 0)
            {
                conected[index] = i;
                index++;
            }
        }
        for (int i = 0; i < index; i++)
        {
            if ((ver[currver].weight + matrix[currver][conected[i]]) <
ver[conected[i]].weight)
            {
                ver[conected[i]].weight = ver[currver].weight +
matrix[currver][conected[i]];
                ver[conected[i]].from = currver;
```

```
        }
    }
    int min = inf;  //znach ver iz min vaha i fix ii
    int ind = 0;
    for (int i = 0; i < number; i++)
    {
        if ((ver[i].weight < min) && (ver[i].fixed != true))
        {
            min = ver[i].weight;
            ind = i;
        }
        ver[ind].fixed = true;
        currver = ind;
    }
}
   int track[number];
int trindex = 0;
currver = number - 1;
track[trindex] = currver;
trindex++;
while (currver > 0)
{
    track[trindex] = ver[currver].from;
    trindex++;
    currver = ver[currver].from;
}
```

```
    cout << "\nMin track(" << ver[number - 1].weight << "): ";
    for (int i = trindex - 1; i >= 0; i--)
    {
        cout << track[i] + 1;
        if (i != 0)cout << "-";
        else cout << endl;
    }
    file.close();
}
```

«Іди в найближчий» *для розв'язання задачі комівояжера.*

```
#include <iostream>
using namespace std;
#define inf 1E9
#define NMAX 8
#define min(x, y) x < y ? x : y

int n,m, temp, ans, d[NMAX][NMAX], t[1 << NMAX][NMAX];
int get(int nmb, int x)
{
    return (x & (1 << nmb)) != 0;
}
int main()
{
    cout << "Enter the value of vertices:" << endl;
    cin >> n;
    cout << "Enter tracks:" << endl;
```

```
   for (int i = 0; i < n; ++i)
      for (int j = 0; j < n; ++j) cin>>d[i][j];
   t[1][0] = 0; m = 1 << n;
   for (int i = 1; i < m; i += 2)
      for (int j = (i == 1) ? 1 : 0; j < n; ++j)
      {
         t[i][j] = inf;
         if (j > 0 && get(j, i))
         {
            temp = i ^ (1 << j);
            for (int k = 0; k < n; ++k)
               if (get(k, i) && d[k][j] > 0) t[i][j] = min(t[i][j], t[temp][k] + d[k][j]);
         }
      }
   for (int j = 1, ans = inf; j < n; ++j)
      if (d[j][0] > 0) ans = min(ans, t[m - 1][j] + d[j][0]);
   cout << "Result: " << endl;
   if (ans == inf) cout<<-1; else cout<<ans;
}
```

Флері *та* ***елементарних циклів*** *знаходження ейлерового ланцюга в ейлеровому графі.*

1). #include <conio.h>

```
#include <stdlib.h>
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
```

```
int main()
{
    cout << "Enter matrix:" << endl;
    const int n = 9;
    int D[9][9];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cin >> D[i][j];
    }
    cout << endl << endl;
    int stV[9] = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };
    int odd_stV[6] = { 0, 0, 0, 0, 0, 0 };
    int r = 6;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (D[i][j] != 0)
                stV[i] = stV[i] + 1;
    int b = 0;
    for (int i = 0; i < n; i++)
    {
        if (stV[i] % 2 == 1)
        {
            stV[i] = 1;
            odd_stV[b] = i;
            b++;
```

```
    }
    else
        stV[i] = 0;
}
for (int q = 0; q < n; q++)
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (i != j && D[i][q] != 0 && D[q][j] != 0)
            {
                if (D[i][j] == 0)
                {
                    D[i][j] = D[i][q] + D[q][j];
                }
                else if (D[i][j] > D[i][q] + D[q][j])
                {
                    D[i][j] = D[i][q] + D[q][j];
                }
            }
cout << "D9:" << endl;
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < r; j++)
        cout << D[i][j] << "\t";
    cout << endl;
}
cout << endl;
```

```
int     y = 0;
b = 0;
int D1[6][6];
for (int i = 0; i < r; i++)
   for (int j = 0; j < r; j++)
      D1[i][j] = 0;
for (int i = 0; i < n; i++)
   if (stV[i] == 1)
   {
      for (int j = 0; j < n; j++)
         if (stV[j] == 1)
         {
            D1[y][b] = D[i][j];
            b++;
         }
      b = 0;
      y++;
   }
cout << "Matrix of odd vertices" << endl;
for (int i = 0; i < r; i++)
{
   for (int j = 0; j < r; j++)
      cout << D1[i][j] << " ";
   cout << endl;
}
```

```
cout << endl << endl;

int D2[15]; string D2_1[15]; string D2_2[15];
D2[0] = D1[0][1] + D1[2][3] + D1[4][5]; D2_1[0] = "012345";
D2[1] = D1[0][1] + D1[2][4] + D1[3][5]; D2_1[1] = "012435";
D2[2] = D1[0][1] + D1[2][5] + D1[3][4]; D2_1[2] = "012534";
D2[3] = D1[0][2] + D1[1][3] + D1[4][5]; D2_1[3] = "021345";
D2[4] = D1[0][2] + D1[1][4] + D1[3][5]; D2_1[4] = "021435";
D2[5] = D1[0][2] + D1[1][5] + D1[3][4]; D2_1[5] = "021534";
D2[6] = D1[0][3] + D1[1][2] + D1[4][5]; D2_1[6] = "031245";
D2[7] = D1[0][3] + D1[1][4] + D1[2][5]; D2_1[7] = "031425";
D2[8] = D1[0][3] + D1[1][5] + D1[2][4]; D2_1[8] = "031524";
D2[9] = D1[0][4] + D1[1][2] + D1[3][5]; D2_1[9] = "041235";
D2[10] = D1[0][4] + D1[1][3] + D1[2][5]; D2_1[10] = "041325";
D2[11] = D1[0][4] + D1[1][5] + D1[2][3]; D2_1[11] = "041523";
D2[12] = D1[0][5] + D1[1][2] + D1[3][4]; D2_1[12] = "051234";
D2[13] = D1[0][5] + D1[1][3] + D1[2][4]; D2_1[13] = "051324";
D2[14] = D1[0][5] + D1[1][4] + D1[2][3]; D2_1[14] = "051423";
int z = 1000;
for (int i = 0; i < 15; i++)
   if (D2[i] < z)
      z = D2[i];
int Z[9][9];
for (int i = 0; i < n; i++)
   for (int j = 0; j < n; j++)
      Z[i][j] = 0;
```

```
y = 0;
string a[6];
string c[6];
for (int i = 0; i < 15; i++)
   if (z == D2[i])
   {
      y = i;
      for (int j = 0; j < r; j++)
      {
         a[j] = D2_1[i][j];
      }
   }
int o = 0;
for (int i = 0; i < r; i++)
   for (int j = 0; j < r; j++)
      if (atoi(a[i].c_str()) == j)
      {
         c[o] = to_string(odd_stV[j]);
         o++;
      }
b = 0;
int q;
int e;
for (int i = 0; i < n; i++)
{
   if (b == 6)
```

```
        break;
    for (int j = 0; j < n; j++)
    {
        if (b >= 6)
            break;
        if ((i == atoi(c[b].c_str())) && (j == atoi(c[b + 1].c_str())))
        {
            q = atoi(a[b].c_str());
            e = atoi(a[b + 1].c_str());
            Z[i][j] = D1[q][e];
            Z[j][i] = D1[q][e];
            b = b + 2;
        }
    }
}
cout << "Additional matrix:" << endl;
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
        cout << Z[i][j] << "  ";
    cout << endl;
}
cout << endl << endl;
string M;
M = M + '0';
int k = 0;
```

```
int g = 0;
int d = 0;
int u = 0;
for (int i = 0; i <= 100; i++)
{
   for (int j = 0; j < n; j++)
   {
      if (D[k][j] != 0)
      {
         g = j;
         d = 1;
         u = 1;
      }
      if ((Z[k][j] != 0) && (u == 0))
      {
         g = j;
         d = 2;
      }
      u = 0;
   }
   if ((D[k][g] != 0) && (d == 1))
   {
      D[k][g] = 0;
      D[g][k] = 0;
      k = g;
      M = M + '-';
```

```
            string h = to_string(k);
            M = M + h;

        }
        if ((Z[k][g] != 0) && (d == 2))
        {
            Z[k][g] = 0;
            Z[g][k] = 0;
            k = g;
            M = M + '-';
            string h = to_string(k);
            M = M + h;
        }
        d = 0;
    }
    cout << "Result: " << endl << M << endl;
    return 0;
}
```

2).

```
#include <iostream>
using namespace std;
int** a;
int n = 16;
int* was;
bool dfs(int key)
{
    was[key] = 1;
```

```
    for (int i = 0; i < n; i++)
    {
        if (was[i] == 0) { if (dfs(i)) return true; }
        else if (was[i] == 1) { return true; }
    }
    was[key] = 2;
    return false;
}
int main()
{
    cout << "Enter the number of vertices:" << endl;
    cin >> n;
    a = new int* [n];
    was = new int[n];
    cout << "Enter matrix:" << endl;
    for (int i = 0; i < n; i++)
    {
        a[i] = new int[n];
        was[i] = 0;
        for (int j = 0; j < n; j++)
        {
            cin >> a[i][j];
        }
    }
    cout << "Result:" << endl;
    for (int i = 0; i < n; i++)
```

```
    {

        if (!was[i] && dfs(i)) { cout << "1\n"; return 0; }
    }

    cout << "0\n";
    return 0;
}
```

Індивідуальні завдання

Варіант 15

Завдання № 1

Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

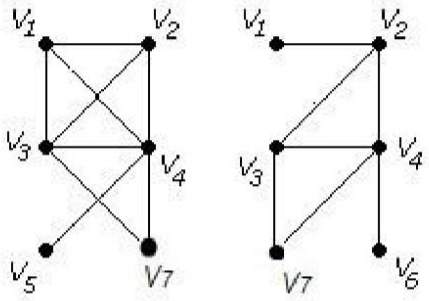
2) об’єднання графів,

3) кільцеву сумму G1 та G2 (G1+G2),

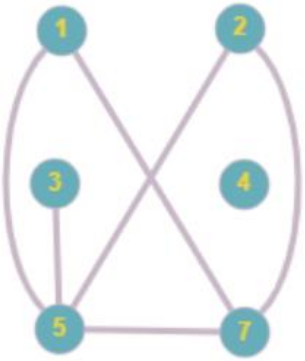
4) розмножити вершину у другому графі,

5) виділити підграф А - що складається з 3-х вершин в G1

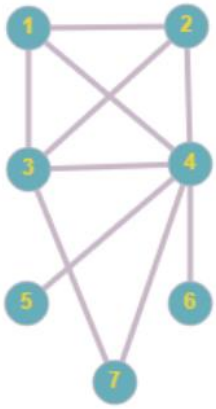
6) добуток графів.



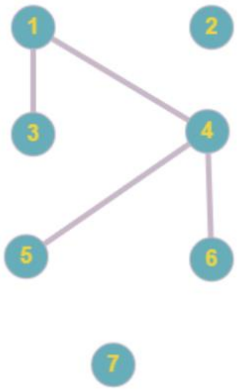
1) 2)


1
2
3
4
5
7

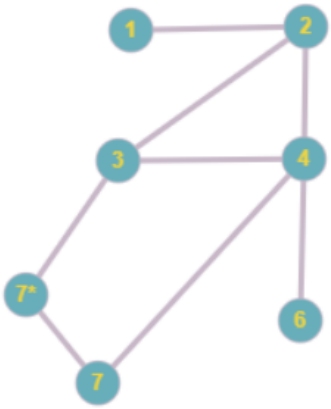
3)

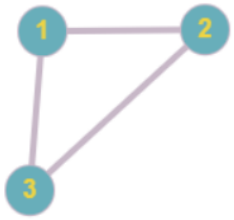

1
2
3
4
5
6
7

4)

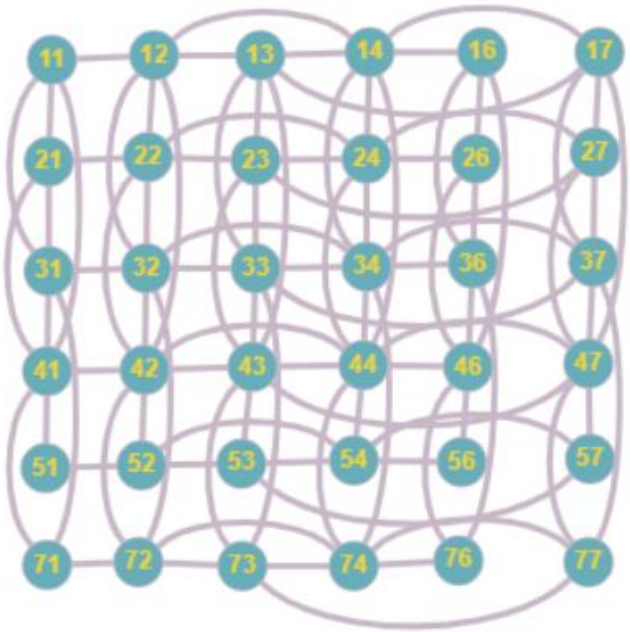

1
2
3
4
5
6
7

5)


1
2
3
4
7*
6
7

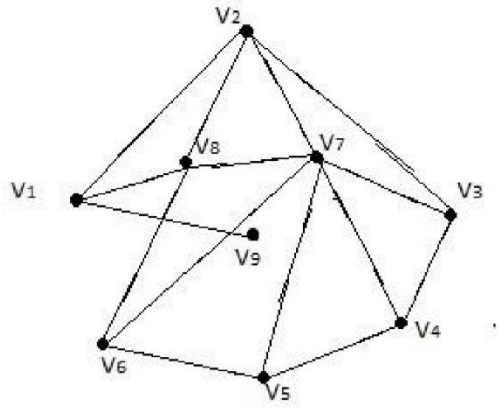

1
2
3

6)


11
12
13
14
16
17
21
22
23
24
26
27
31
32
33
34
36
37
41
42
43
44
46
47
51
52
53
54
56
57
71
72
73
74
76
77

Завдання № 2

Скласти таблицю суміжності для неографа.



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

Завдання № 3

Для графа з другого завдання знайти діаметр

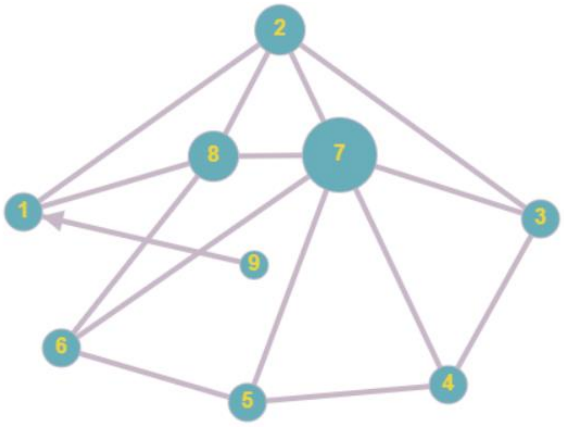
Діаметр графа=4

Завдання № 4

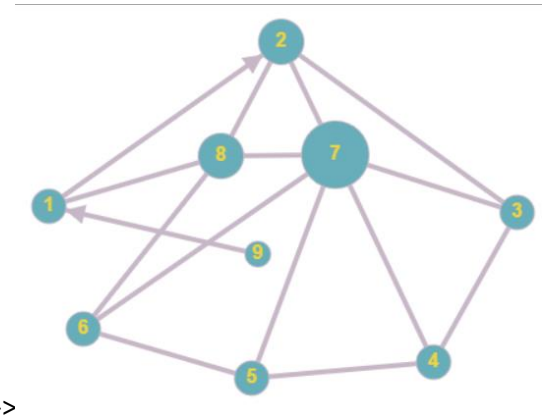
Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).

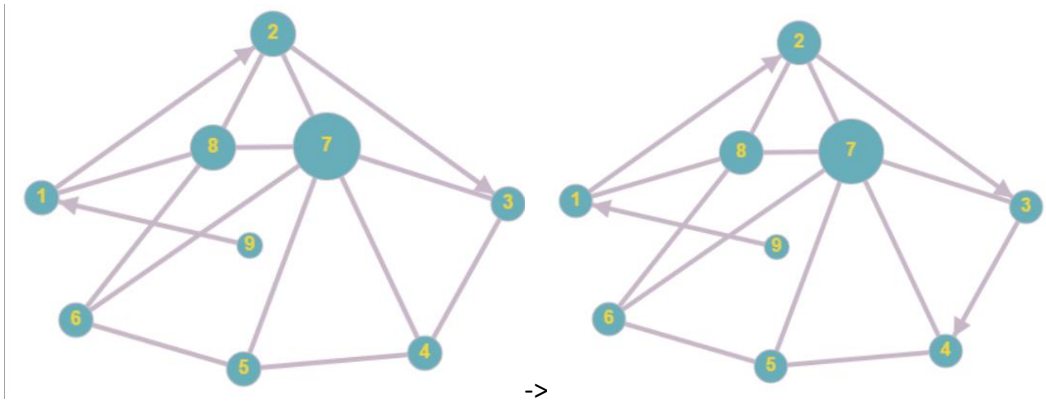
Обхід вглиб.

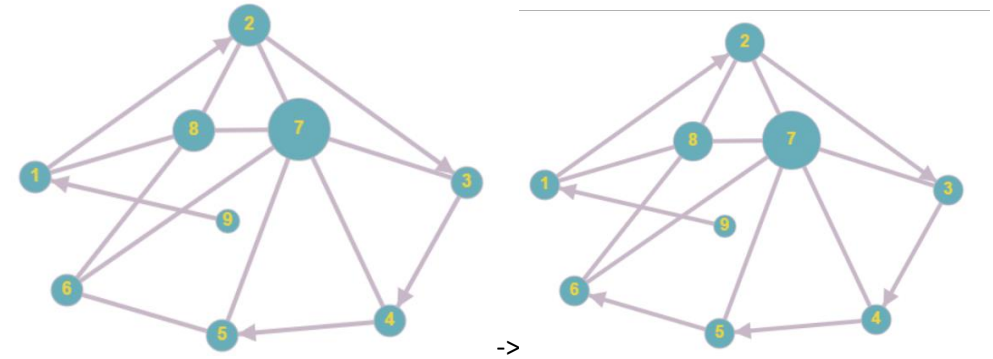
Початкова вершина-9;

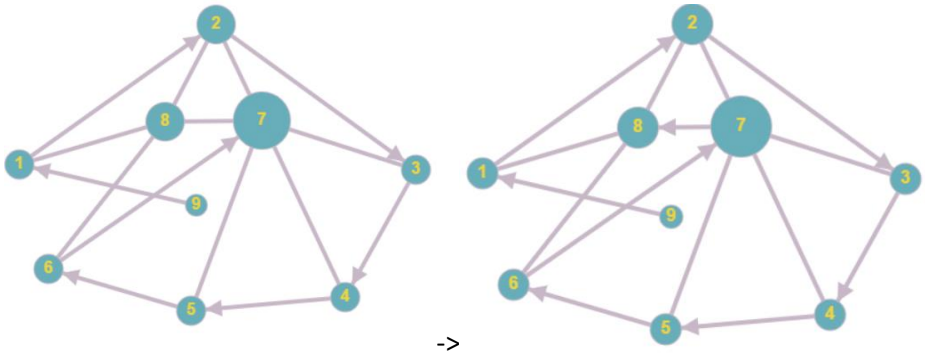


->





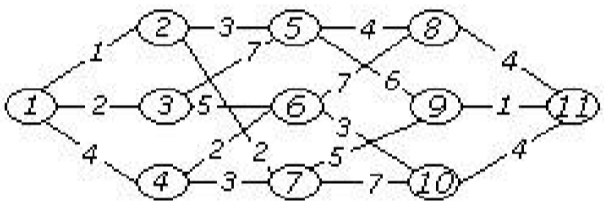




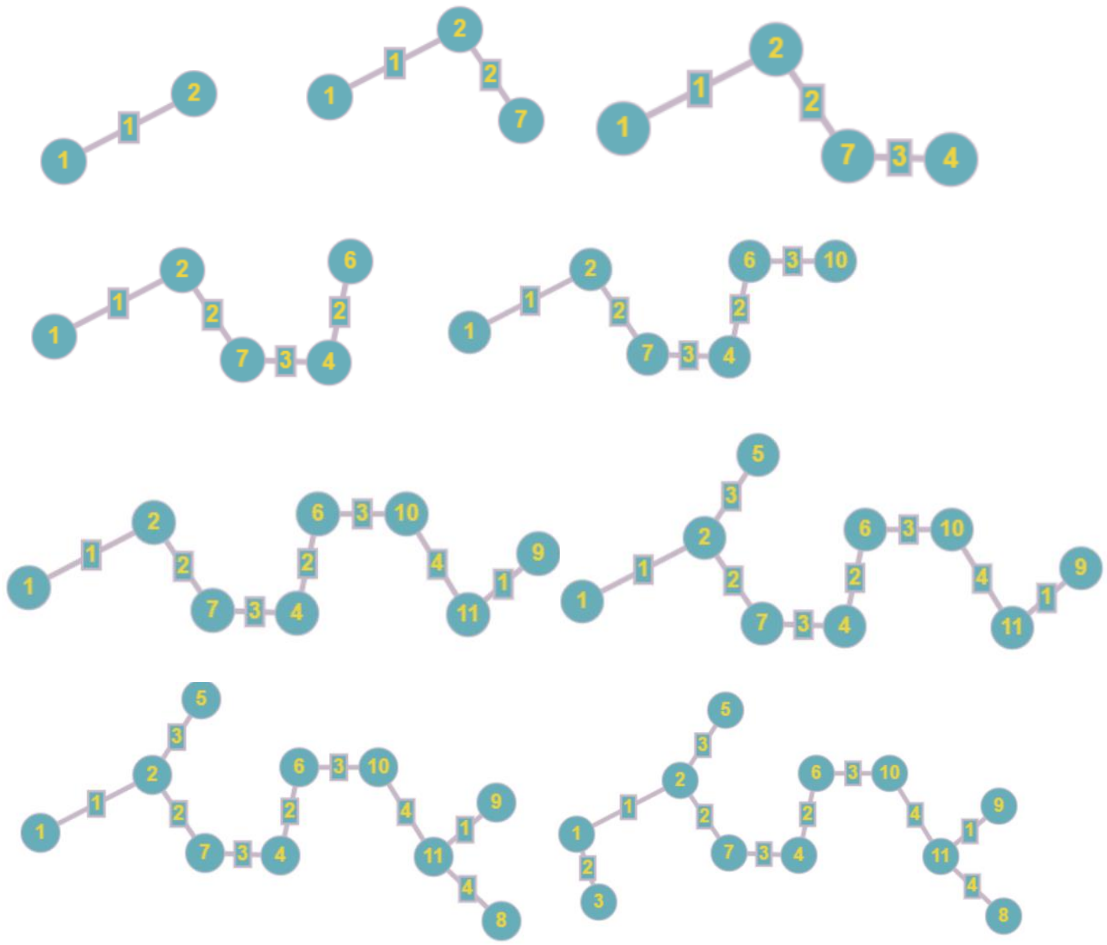
Порядок обхода: 9 1 2 3 4 5 6 7 8

Завдання № 5

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



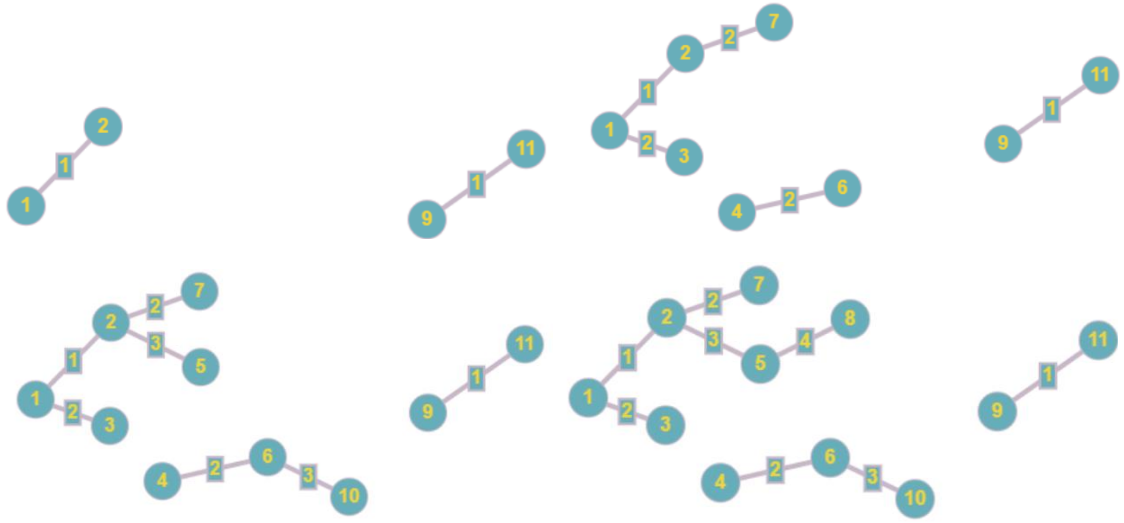
Прима:



V={1;2;7;4;6;10;11;9;5;8;3}

E={(1;2);(2;7);(7;4);(4;6);(6;10);(10;11);(11;9);(2;5);(11;8);(1;3)}

Краскала:



V={1;2;9;11;3;4;6;7;5;10;8}

E={(1;2);(9;11);(1;3);(4;6);(2;7);(2;5);(6;10);(5;8)}

Завдання № 6

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

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

Порядок:1 4 6 3 5 7 2 8 1

1+1+1+2+2+1+2+2=12

1 4 6 3 5 7 2 8 {12}

1 4 8 2 7 5 3 6 1 {12}

1 4 8 2 7 5 6 3 1 {12}

2 7 5 3 6 4 1 8 2 {12}

2 7 5 6 3 1 4 8 2 {12}

3 6 4 1 8 2 7 5 3 {12}

3 6 4 8 2 7 5 1 3 {12}

4 1 3 6 5 7 2 8 4 {12}

4 1 6 3 5 7 2 8 4 {12}

4 1 8 2 7 5 3 6 4 {12}

4 6 3 1 5 7 2 8 4 {12}

4 6 3 5 7 2 8 1 4 {12}

4 8 2 7 5 1 3 6 4 {12}

4 6 3 5 7 2 8 1 4 {12}

5 3 6 4 1 8 2 7 5 {12}

5 6 3 1 4 8 2 7 5 {12}

5 7 2 8 4 1 3 6 5 {12}

5 7 2 8 4 1 6 3 5 {12}

5 7 2 8 4 6 3 1 5 {12}

6 3 1 4 8 2 7 5 6 {12}

6 3 5 7 2 8 4 1 6 {12}

6 4 1 8 2 7 5 3 6 {12}

6 4 8 2 7 5 1 3 6 {12}

7 2 8 4 1 3 6 5 7 {12}

7 2 8 4 1 6 3 5 7 {12}

7 2 8 4 6 3 1 5 7 {12}

8 4 1 3 6 5 7 2 8 {12}

7 2 8 4 6 3 1 5 7 {12}

4 8 2 7 5 1 3 6 4 {12}

4 8 2 7 5 3 6 1 4 {12}

4 8 2 7 5 6 3 1 4 {12}

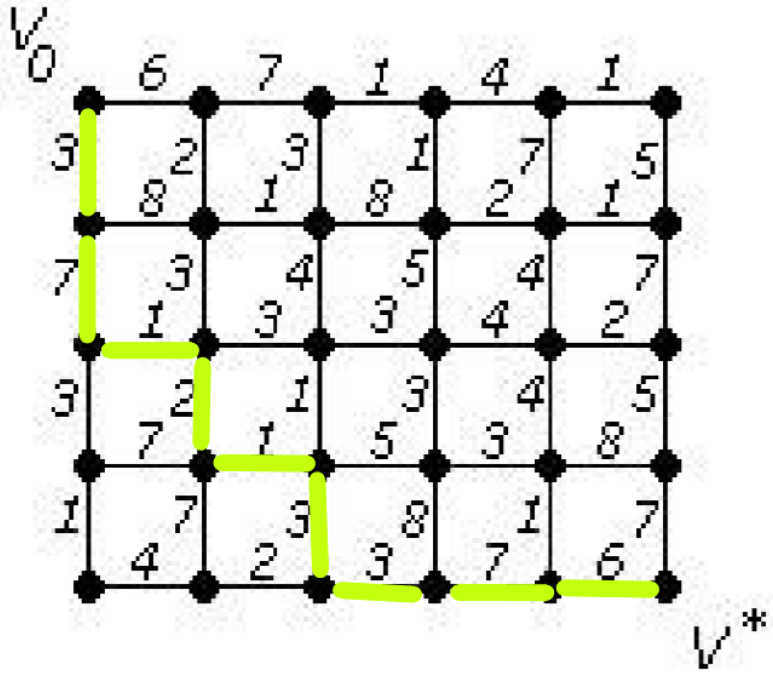
8 4 1 3 6 5 7 2 8 {12}

8 4 1 6 3 5 7 2 8 {12}

8 4 6 3 1 5 7 2 8 {12}

Завдання № 7

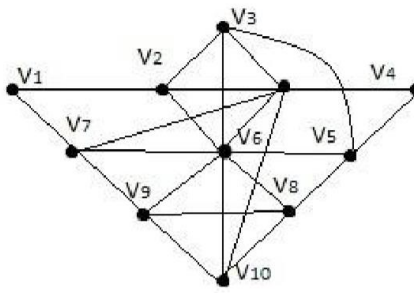
За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V_0 і V*



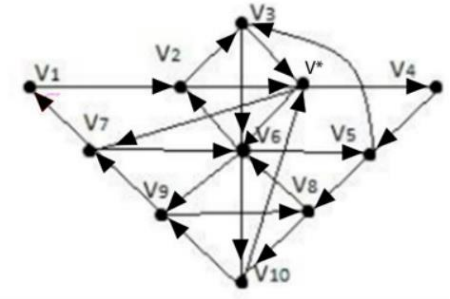
Мінімальна довжина шляху:31

Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами:



а) Флері



(v1,v2);(v2,v3);(v3,v*);(v*;v6);(v6,v2);(v2,v*);(v*,v4);(v4,v5);(v5;v8);(v8,v10);(v10;v9);(v9,v8);(v8,v6);(v6,v9);(v9,v7);(v7,v6);(v6;v5);(v5,v3);(v3,v6);(v6,v10);(v10,v*);(v*,v7);(v7,v1)

v1->v2->v3->v*->v6->v2->v*->v4->v5->v8->v10->v9->v8->v6->v9->v7->v6->v5->v3->v6->v10->v*->v7->v1

б) елементарних циклів.

1)v2->v3->v*->v6->v2

2)v8->v10->v9->v8

3)v6->v5->v3->v6

4)v6->v9->v7->v6

V1->(1)->v*->v4->v5->(2)->(v6->v9->v7-(v6)->v5->v3->v6)->v10->v*->v7->v1

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

15. $x\bar{z} \vee xy \vee yz$

x ¬ z v yz=xy;

xy v xy=xy;