МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

з дисципліни

«Дискретна математика»

Виконала:

студентка групи КН-114

Олескевич Софія

Викладач:

Мельникова Н.І.

Львів – 2019р.

Індивідуальний варіант за номером залікової книги №8

Завдання № 1

Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

2) об’єднання графів,

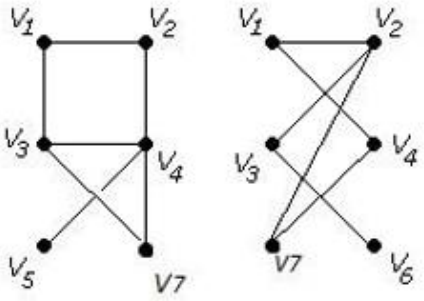
3) кільцеву суму G1 та G2 (G1+G2),

4) розмножити вершину у другому графі,

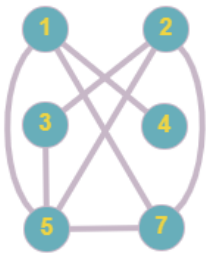
5) виділити підграф А - що складається з 3-х вершин в G1

6) добуток графів.

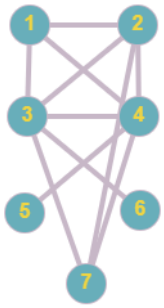
8)



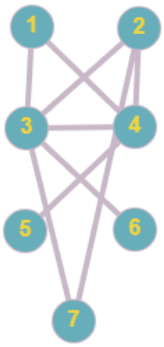
1) Доповнення до графа G1



2) Об’єднання G1 та G2

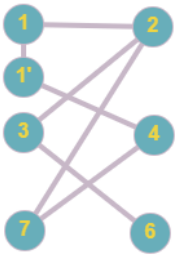


3) Кільцева сума G1⊕G2

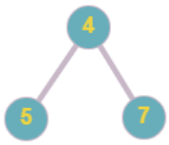


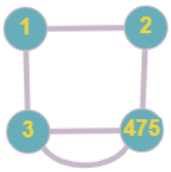
4) Розщепити вершину в G2

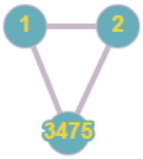
Розщепаю вершину 1:



5) Стягнути підграф A, який складається з трьох вершин



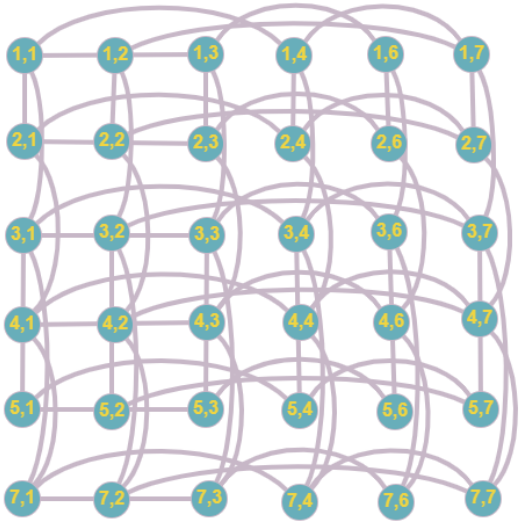




6) Добуток графів

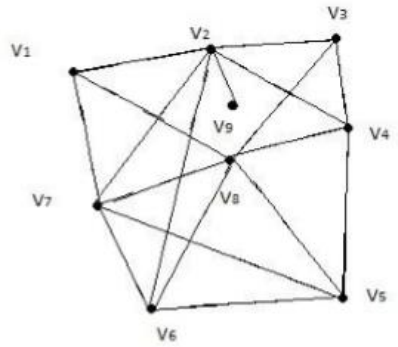
G1:

G2:



Завдання № 2

Скласти таблицю суміжності для неографа



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

Завдання № 3

Для графа з другого завдання знайти діаметр.

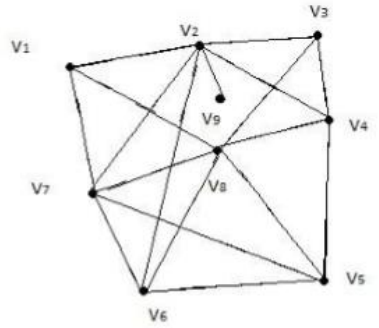
Почну з вершини 1

Діаметр графа = 3 (V5 V7 V2 V9)

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

Завдання № 4

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



Почну пошук з вершини V1 і зроблю таблицю.

Вершина	BFS номер	Черга
V1	0	V1
V2	1	V1 V2
V7	2	V1 V2 V7
V8	3	V1 V2 V7 V8
-	-	V2 V7 V8
V3	4	V2 V7 V8 V3
V4	5	V2 V7 V8 V3 V4
V6	6	V2 V7 V8 V3 V4 V6
V9	7	V2 V7 V8 V3 V4 V6 V9
-	-	V7 V8 V3 V4 V6 V9
V5	8	V7 V8 V3 V4 V6 V9 V5
-	-	V8 V3 V4 V6 V9 V5
-	-	V3 V4 V6 V9 V5
-	-	V4 V6 V9 V5
-	-	V6 V9 V5
-	-	V9 V5
-	-	V5
-	-	Ø

Програмна реалізація

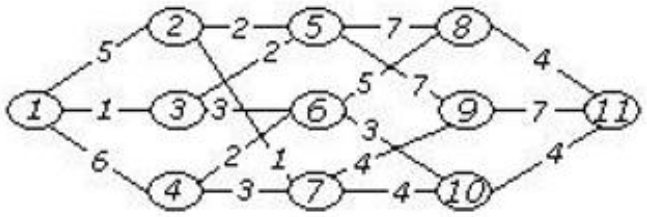
```cpp
#include <iostream>
#include <queue> // черга
using namespace std;
int main()
{
  queue <int> Queue;
  int mas[9][9] = { { 0, 1, 0,0,0,0,1,1,0 }, // матриця суміжності
  { 1,0,1,1,0,1,1,0,1 },
  { 0,1,0,1,0,0,0,1,0 },
  { 0,1,1,0,1,0,0,1,0 },
  { 0,0,0,1,0,1,1,1,0 },
  { 0,1,0,0,1,0,1,1,0 },
  { 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 } };
  int nodes[9]; // вершини графа
  for (int i = 0; i < 9; i++)
  nodes[i] = 0; //  всі вершини рівні 0
  Queue.push(0); // ставлю в чергу першу вершину
  while (!Queue.empty())//поки черга не пуста
  {
    int node = Queue.front(); // витягую вершину
    Queue.pop();
    nodes[node] = 2; // помічаю її як пройдену
    for (int j = 0; j < 9; j++)
        { // перевіряю для неї всі суміжні вершини
      if (mas[node][j] == 1 && nodes[j] == 0)
      { // якщо вершина суміжна і ще не в черзі
        Queue.push(j); // добавляємо її в чергу
        nodes[j] = 1; // помічаю вершину як пройдену
      }
    }
    cout << node + 1 << endl; // виводжу номер вершини
}
  cin.get();
  return 0;
}
```

```
D:\code\т°шЁ\bin\Debug\т°шЁ.exe
1
2
7
8
3
4
6
9
5
```

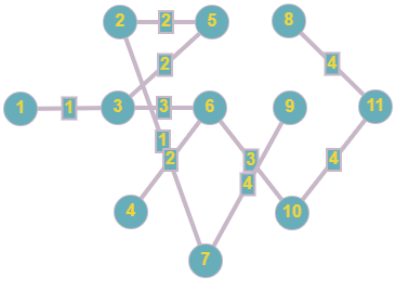
Завдання № 5

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

8)



Метод Краскала



Пошук ребер з найменшою довжиною 1: 1-3, 2-7

Ребра з довжиною 2: 2-5, 3-5, 4-6 і тд шукаємо ребра з більшою довжиною ніж попередня, але слідкувати щоб не утворювався цикл.

V: {1,3,2,7,5,4,6,10,9,8,11}

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

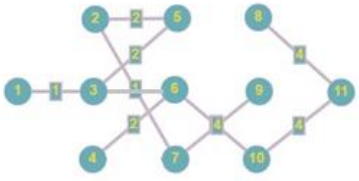
Сума мінімального остового дерева дорівнює 26

Метод Прима

З вершини 1 шукаю ребра з мінімальною вагою, які виходять з цієї точки, або входять в неї. Позначу ребро з мінімальною вагою(наразі з вагою 1). Оскільки з вершини 1 нема більше ребер з малою вагою, то переходимо до вершини 3 і з неї так само шукаємо ребра з мінімальною вагою

V: {1,3,5,2,7,4,6,10,9,11,8}

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



Сума мінімального остового дерева дорівнює 26.

Краскала:

```
#include <algorithm>
```

```
Enter E: 18
1 2 5
1 3 1
1 4 6
2 7 1
2 5 2
3 5 2
3 6 3
4 7 3
4 6 2
7 9 4
7 10 4
10 6 3
6 8 5
5 8 7
5 9 7
10 11 4
9 11 7
11 8 4

Min ostov tree:
1 3 1
2 7 1
2 5 2
3 5 2
4 6 2
3 6 3
6 10 3
8 11 4
10 11 4
7 9 4
```

in.cpp

```cpp
#include <iostream>
#include <algorithm>
#include <functional>
#include <iostream>
#include <iterator>
using namespace std;
class Edge{
public:
    int sourse;
    int dest;
    int weight;
};

int compare(Edge e1, Edge e2){
return e1.weight< e2.weight;
}

int findParent(int v, int* parent){
if (parent[v]==v){
    return v;
}
return findParent(parent[v], parent);
}

void kruskal(Edge *input , int n, int E)//передаю кількість вершин, ребер
{
sort(input, input+E, compare);//сортує за зростанням і порівнює

Edge *output=new Edge[n-1];//ост дерево скл з n-1 ел

int *parent=new int [n];
for(int i=0; i<n; i++){
    parent[i]=i;
}
int count=0, i=0;//слідкую яке ребро розглядається
while(count!=n-1){
Edge currentEdge=input[i];//ребро
//checking if can add currentEdge in MST
int sourseParent = findParent (currentEdge.sourse, parent ) ;
int destParent = findParent (currentEdge.dest, parent ) ;

if (sourseParent!=destParent){

output[count]=currentEdge;// на місце count ставлю поточне ребро
count++;
parent[sourseParent]=destParent;
}
i++;}
cout<<"\nMin ostov tree: "<<endl;
for(int i=0; i<n-1; i++){
    if(output[i].sourse< output[i].dest){
    cout<<output[i].sourse<<" "<<output[i].dest<<" "<<output[i].weight<<endl;}
else {
    cout<<output[i].dest<<" "<<output[i].sourse<<" "<<output[i].weight<<endl;}}

for(int i=0; i<n-1; i++){
int sum=0;
sum+=output[i].weight;
//int sum;
cout<<"\nSum: "<<sum ;}}
int main()
{
    int n, E, sum;
    cout<<"Enter n: ";
    cin>>n;
    cout<<"Enter E: ";
    cin>>E;
     Edge *input= new Edge[E];
        for(int i=0; i<E; i++){
        int s,d,w;
        cin>>s>>d>>w;
        input[i].sourse=s;
        input[i].dest=d;
        input[i].weight=w;}
kruskal(input, n, E);
return 0;}
```

Прима:

```cpp
#include<iostream>
using namespace std;
int main()
{
    int a,b,u,v,n,i,j,ne=1;
    int visited[20]={0},min,mincost=0;
   int cost[20][20];
    int path[100]={0};
    int path_index=0;

    cout<<"Enter the number of vertices: ";
    cin>>n;
    cout<<"Enter the adjacency matrix: " << endl;
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++)
        {
            cin>>cost[i][j];
            if(cost[i][j]==0)
                cost[i][j]=99;
        }
    }
    visited[1]=1;
    cout<<endl;;

    while(ne < n)
    {
        for(i=1,min=99;i<=n;i++)
            for(j=1;j<=n;j++)
                if(cost[i][j]< min)
                    if(visited[i]!=0)
                    {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;
                    }
        if(visited[u]==0 || visited[v]==0)
        {
            path[path_index]=b;
```

```cpp
    visited[1]=1;
    cout<<endl;;

    while(ne < n)
    {
        for(i=1,min=99;i<=n;i++)
            for(j=1;j<=n;j++)
                if(cost[i][j]< min)
                    if(visited[i]!=0)
                    {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;
                    }
        if(visited[u]==0 || visited[v]==0)
        {
            path[path_index]=b;
            path_index++;
            ne++;
            mincost+=min;
            visited[b]=1;
        }
        cost[a][b]=cost[b][a]=99;
    }
    cout<<endl;
    cout<<1<<" -> ";
    for (int i=0;i<n-1;i++)
    {
        cout<<path[i];
        if (i<n-2) cout<<" -> ";
    }
    cout<< endl << "The least way is: "<<mincost;
    cin.get();
    cin.get();
    return 0;
}
```

```
Enter the number of vertices: 11
Enter the adjacency matrix:
0 5 1 6 0 0 0 0 0 0 0
5 0 0 0 2 0 1 0 0 0 0
1 0 0 0 2 3 0 0 0 0 0
6 0 0 0 0 2 3 0 0 0 0
0 2 2 0 0 0 0 7 7 0 0
0 0 3 2 0 0 0 5 0 3 0
0 1 0 3 0 0 0 0 4 4 0
0 0 0 0 7 5 0 0 0 0 4
0 0 0 0 7 0 4 0 0 0 7
0 0 0 0 0 3 4 0 0 0 4
0 0 0 0 0 0 0 4 7 4 0


1 -> 3 -> 5 -> 2 -> 7 -> 6 -> 4 -> 10 -> 9 -> 11 -> 8
The least way is: 26
Process returned 0 (0x0)   execution time : 3.837 s
Press any key to continue.
```

Завдання № 6

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

Розглядаю 1 стовбець, мінімальне значення=2, перший стовбець та рядок викреслюю.

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

	2	3	4	5	6	17	8
2	∞	6	1	5	1	1	2
3	6	∞	5	1	7	5	5
4	1	5	∞	3	3	2	3
5	5	1	3	∞	2	2	3
6	1	7	3	2	∞	5	7
17	1	5	2	2	5	∞	5
8	2	5	3	3	7	5	∞

	172	3	4	5	6	8
172	∞	6	1	5	1	2
3	6	∞	5	1	7	5
4	1	5	∞	3	3	3
5	5	1	3	∞	2	3
6	1	7	3	2	∞	7
8	2	5	3	3	7	∞

	3	1724	5	6	8
3	∞	5	1	7	5
1724	5	∞	3	3	3
5	1	3	∞	2	3
6	7	3	2	∞	7
8	5	3	3	7	∞

	3	17245	6	8
3	∞	1	7	5
17245	1	∞	2	3
6	7	2	∞	7
8	5	3	7	∞

	172453	6	8
172453	∞	7	5
6	7	∞	7
8	5	7	∞

	6	1724538
6	∞	7
1724538	7	∞

Шлях 1724538. Вага 20. Аналогічно можна зробити обхід з інших вершин. Для кращого розуміння розпочну з вершини 2

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

	1	3	24	5	6	7	8
1	∞	3	5	4	6	2	3
3	3	∞	5	1	7	5	5
24	5	5	∞	3	3	2	3
5	4	1	3	∞	2	2	3
6	6	7	3	2	∞	5	7
7	2	5	2	2	5	∞	5
8	3	5	3	3	7	5	∞

	1	3	5	6	247	8
1	∞	3	4	6	2	3
3	3	∞	1	7	5	5
5	4	1	∞	2	2	3
6	6	7	2	∞	5	7
247	2	5	2	5	∞	5
8	3	5	3	7	5	∞

	2471	3	5	6	8
2471	∞	3	4	6	3
3	3	∞	1	7	5
5	4	1	∞	2	3
6	6	7	2	∞	7
8	3	5	3	7	∞

	24713	5	6	8
24713	∞	1	7	5
5	1	∞	2	3
6	7	2	∞	7
8	5	3	7	∞

	247135	6	8
247135	∞	2	3
6	2	∞	7
8	3	7	∞

	2471356	8
2471356	∞	7
8	7	∞

Bara= 1+2+2+3+1+2+7=11+7=18

```cpp
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
int formation(int vershunu, int **arr);
void print(int vershunu, int **arr);
bool check_node(int *V, int vershunu, int Node);
int build(int vershunu, int **arr, int *V, int *counter_of_nodes);
void del(int vershunu, int **arr);
int min_numb(int vershunu, int min, int **arr, int *V, int i);

int main() {
    int vershunu, counter_of_nodes = 1, min = 999, rez = 0;
    cout << "Input size: ";
    cin >> vershunu;
    int *V = new int[vershunu];
    int *Vcon = new int[vershunu];

    int **arr = new int *[vershunu];
    for (int i = 0; i < vershunu; i++)
        arr[i] = new int[vershunu];
    cout << "Input matrix=";
    if (formation(vershunu, arr) == -1) {
        cout << "Program error" << endl;
    } else {
        print(vershunu, arr);
        cout << endl;
        delete[] V;
        for (int i = 0; i < vershunu; i++) {
            int *V = new int[vershunu];
            V[0] = i;
            counter_of_nodes = 1;
            rez = build(vershunu, arr, V, &counter_of_nodes);
        }
        cout << endl << "The shortest way:" << rez << endl;
        for (int i = 0; i < vershunu; i++) {

            if (i == vershunu - 1)
```

```
38              if (i == vershunu - 1)
39                  cout << V[i] + 1;
40              else cout << V[i] + 1 << " -> ";
41          }
42      del(vershunu, arr);
43      delete[] Vcon;
44      return 0;
45  }
46  }
47  int formation(int vershunu, int **arr) {
48      for (int i = 0; i < vershunu; i++) {
49          for (int j = 0; j < vershunu; j++) {
50              cin >> arr[i][j];
51              if (cin.fail()) {
52                  cout << "Input error" << endl;
53                  return -1;
54              }
55          }
56      }
57      return 1;
58  }
59  void print(int vershunu, int **arr) {
60      for (int i = 0; i < vershunu; i++) {
61          for (int j = 0; j < vershunu; j++) {
62              cout << setw(4) << arr[i][j];
63          }
64          cout << endl;
65      }
66  }
67  void del(int vershunu, int **arr) {
68      for (int i = 0; i < vershunu; i++) {
69          delete[] arr[i];
70      }
71      delete[] arr;
72
73  }
74  bool check_node(int *V, int vershunu, int Node) {
75      for (int i = 0; i < vershunu; i++)
```

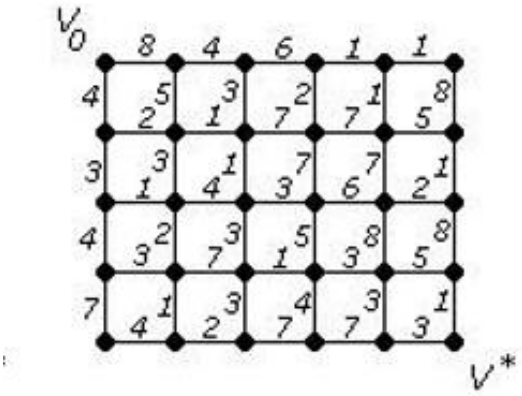
```
62              cout << setw(4) << arr[i][j];
63          }
64          cout << endl;
65      }
66  }
67  void del(int vershunu, int **arr) {
68      for (int i = 0; i < vershunu; i++) {
69          delete[] arr[i];
70      }
71      delete[] arr;
72
73  }
74  bool check_node(int *V, int vershunu, int Node) {
75      for (int i = 0; i < vershunu; i++)
76          if (V[i] == Node)return false;
77      return true;
78  }
79
80  int build(int vershunu, int **arr, int *V, int *counter_of_nodes) {
81      int sum = 0, min;
82      for (int i = 0; i < *counter_of_nodes; i++) {
83          min = min_numb(vershunu, 999, arr, V, i);
84          for (int j = 0; j < vershunu; j++)
85              if (arr[V[i]][j] == min && check_node(V, vershunu, j)) {
86                  V[*counter_of_nodes] = j;
87                  (*counter_of_nodes)++;
88                  sum += arr[V[i]][j];
89                  break;
90              }
91      }
92      return sum;
93  }
94  int min_numb(int vershunu, int min, int **arr, int *V, int i) {
95      for (int j = 0; j < vershunu; j++)
96          if (min > arr[V[i]][j] && arr[V[i]][j] != 0 && check_node(V, vershunu, j)) min = arr[V[i]][j];
97      return min;}
98
```

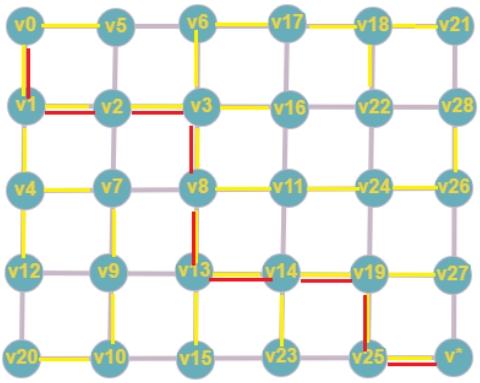
```
Input size: 8
Input matrix= 0 7 3 5 4 6 2 3
7 0 6 1 5 1 1 2
3 6 0 5 1 7 5 5
5 1 5 0 3 3 2 3
4 5 1 3 0 2 2 3
6 1 7 3 2 0 5 7
2 1 5 2 2 5 0 5
3 2 5 3 3 7 5 0
   0   7   3   5   4   6   2   3
   7   0   6   1   5   1   1   2
   3   6   0   5   1   7   5   5
   5   1   5   0   3   3   2   3
   4   5   1   3   0   2   2   3
   6   1   7   3   2   0   5   7
   2   1   5   2   2   5   0   5
   3   2   5   3   3   7   5   0


The shortest way:15
1 -> 7 -> 2 -> 4 -> 5 -> 3 -> 8 -> 6
Process returned 0 (0x0)   execution time : 104.388 s
Press any key to continue.
```

Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V* .

8)





Розпишу найменші відстані до кожної вершини

L(v1)=4

L(v2)=6

L(v3)=7

L(v4)=7

L(v5)=8

L(v6)=8

l(v7)=8

L(v8)=8

L(v9)=10

L(v10)=11

L(v11)=11

L(v12)=11

L(v13)=11

L(v14)=12

L(v15)=13

L(v16)=14

L(v17)=14

L(v18)=15

L(v19)=15

L(v20)=15

L(v21)=16

L(v22)=16

L(v23)=16

L(v24)=17

L(v25)=18

L(v26)=19

L(v27)=20

L(v28)=20

L(v*)=21

Мінімальна відстань від v0 до v* дорівнює 21.

Дейкстра:

```
D:\code\DM5\bin\Debug\DM5.exe
top11 to top12 :5
top11 to top17 :7
top12 to top13 :0
top12 to top18 :1
top13 to top14 :1
top13 to top19 :4
top14 to top15 :4
top14 to top20 :2
top15 to top16 :3
top15 to top21 :3
top16 to top17 :6
top16 to top22 :5
top17 to top18 :2
top17 to top23 :8
top18 to top19 :0
top18 to top24 :8
top19 to top20 :3
top19 to top25 :7
top20 to top21 :7
top20 to top26 :1
top21 to top22 :1
top21 to top27 :3
top22 to top23 :3
top22 to top28 :4
top23 to top24 :5
top23 to top29 :3
top24 to top25 :0
top24 to top30 :1
top25 to top26 :4
top26 to top27 :2
top27 to top28 :7
top28 to top29 :7
top29 to top30 :3
Enter the Source Node : 1
The least way is: 21
The way is: Top -> Top7 -> Top8 -> Top9 -> Top15 -> Top21 -> Top22 -> Top23 -> Top29 -> Top30 -> finish

Process returned 0 (0x0)   execution time : 121.316 s
Press any key to continue.
```

```cpp
#include <iostream>
using namespace std;
int n;
int g[50][50];

bool visited[50];
int dist[50];
int pred[50];

void graf(int g[50][50])
{
    //int g[50][50];
    cout<<"Number of tops?";
    cin>>n;
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            g[i][j]=0;
        }
    }

int t1, t2;
cout<<"Number of rows and columns?"; //6 5
cin>>t1>>t2;

for (int i=0; i<n; i++){
    for(int j=i+1; j<n; j++){
      if(j==i+1 || j==i+t1){
        cout<<"top"<<i+1<<" to top"<<j+1<<" :";
        cin>>g[i][j];
      }else { g[i][j]=0; }
    }
 }
}

int distance()
{

  int minimum = 10000, minDist;
```

```cpp
int distance()
{

  int minimum = 10000, minDist;
  for (int z = 0; z < n; z++)
    if (visited[z] == false && dist[z] <= minimum) //якщо візітет не пройдена і відстань до неї менша за мінімум, то вона стає мінімальною
    {
     minimum = dist[z];
     minDist = z;
    }
  return minDist;
}


void printPath(int j)
{
  if (pred[j] == -1)
    return;
  printPath(pred[j]);
  cout << "Top" << j+1 << " -> ";
}

void dijkstra(int g[50][50])
{
    //int g[50][50];
    int src;
    cout << "Enter the Source Node : ";
    cin >> src;
    for (int i = 0; i < n; i++)
    {
      pred[0] = -1;
      dist[i] = 10000;
      visited[i] = false;
    }
    dist[src-1] = 0;
    for (int count = 0; count < n - 1; count++)
    {
      int u = distance();
```

```cpp
    for (int i = 0; i < n; i++)
    {
      pred[0] = -1;
      dist[i] = 10000;
      visited[i] = false;
    }
    dist[src-1] = 0;
    for (int count = 0; count < n - 1; count++)
    {
      int u = distance();
      int q;
      visited[u] = true;
      for (int z = 0; z < n; z++)
        if (!visited[z] && g[u][z] && dist[u] + g[u][z] < dist[z])// якщо відстань менша ніж була раніше
        {
          pred[z] = u; //стає вершиною яка входить в найкоротший шлях
          q++;
          dist[z] = dist[u] + g[u][z]; //+ вершину від якої йшли до u
        }
    }
    cout << "The least way is: ";
    cout << dist[29] << endl;
    cout << "The way is: ";
    cout << "Top -> ";
      printPath(29);
      cout << "finish" << endl;}

int main()
{

int g[50][50];
graf(g);
dijkstra(g);
return 0;
```

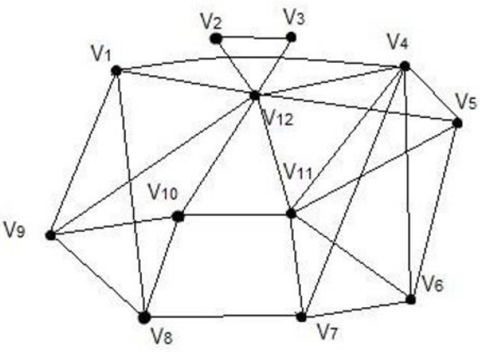
Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами:

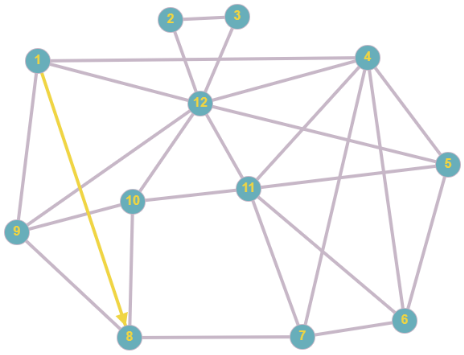
а) Флері;

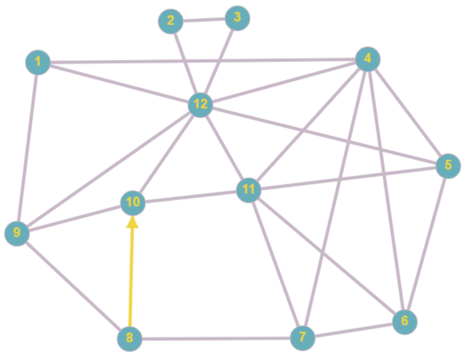
б) елементарних циклів.

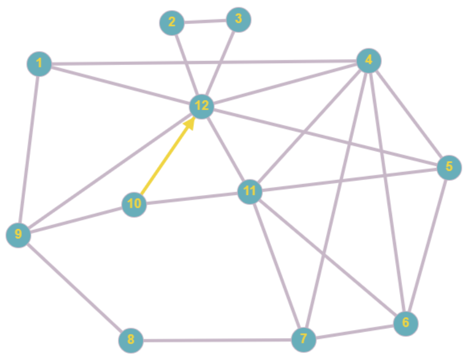
8)

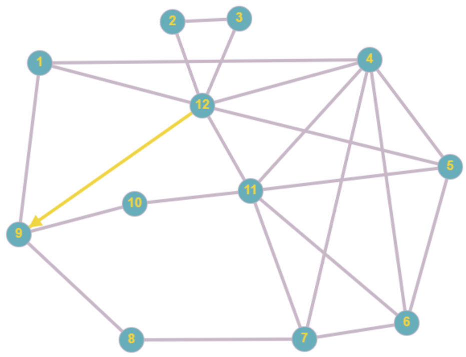


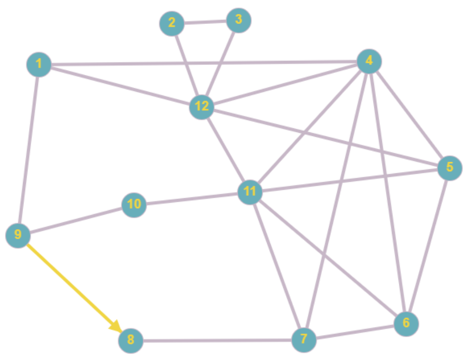
Почну з вершини V1

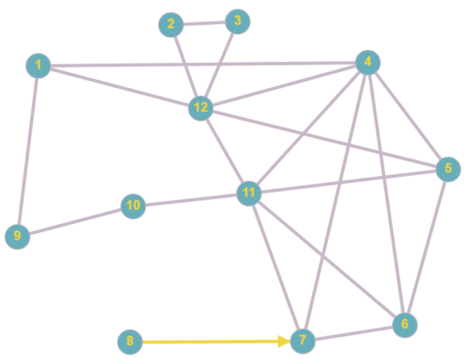


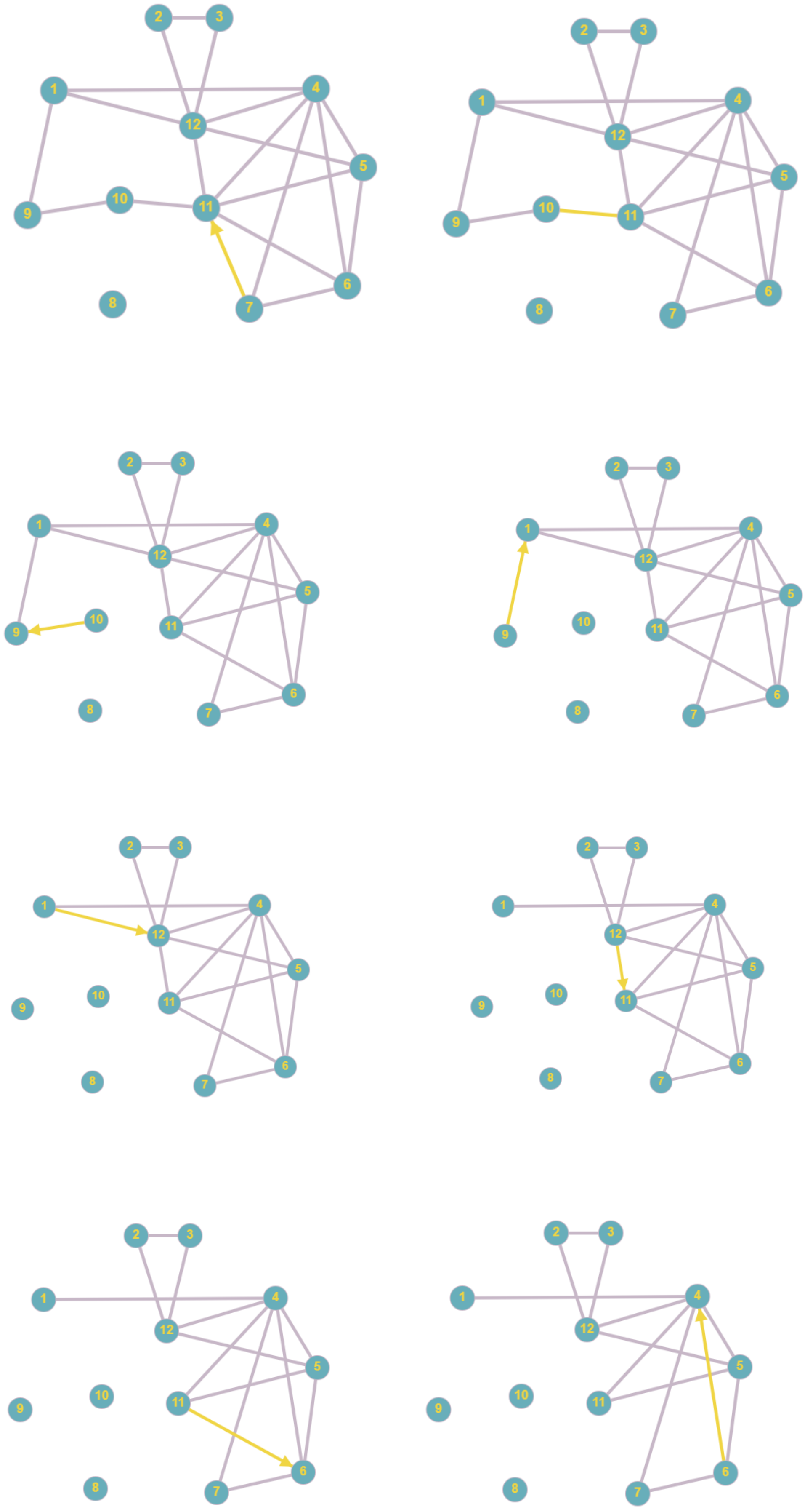


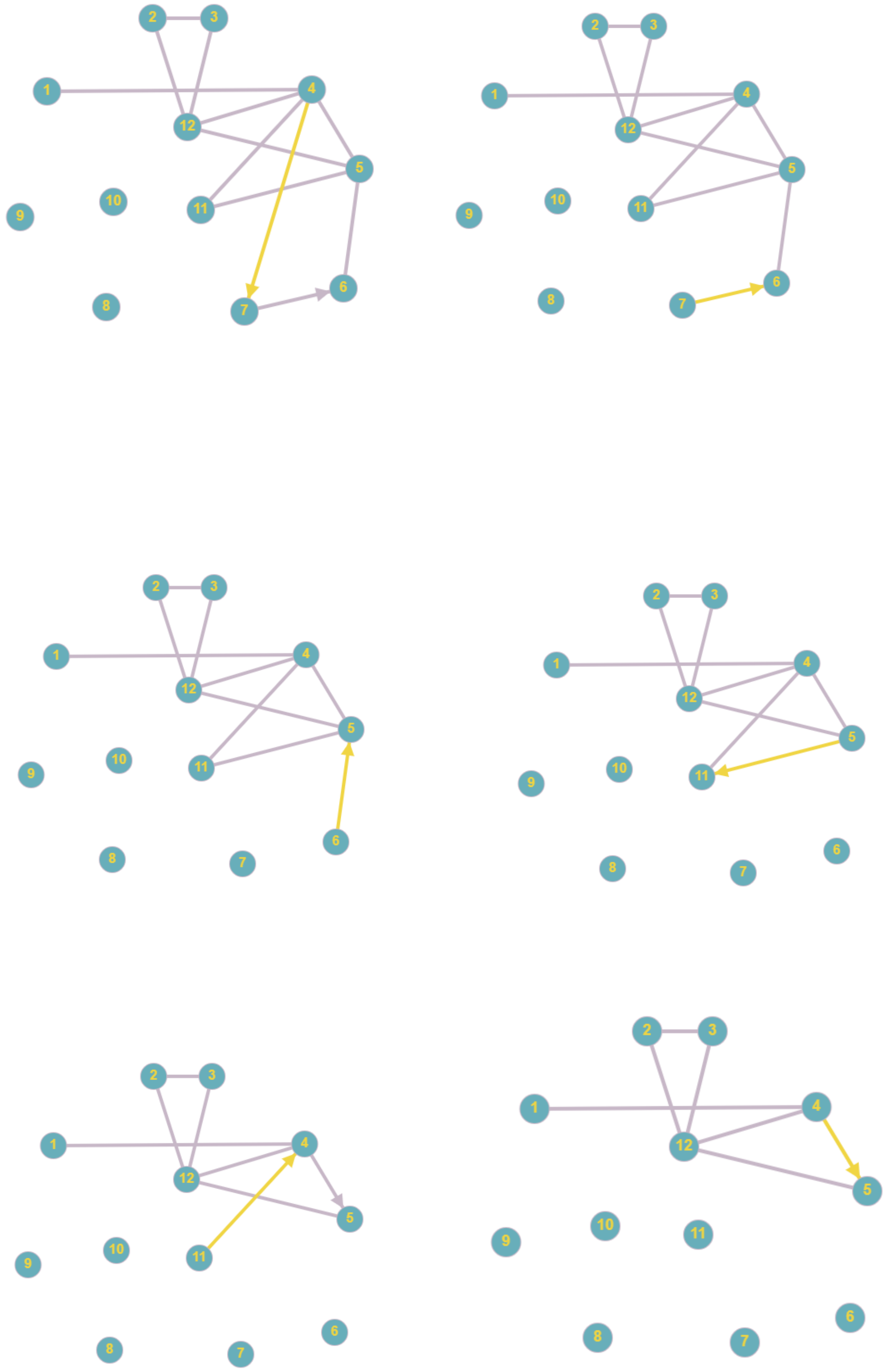




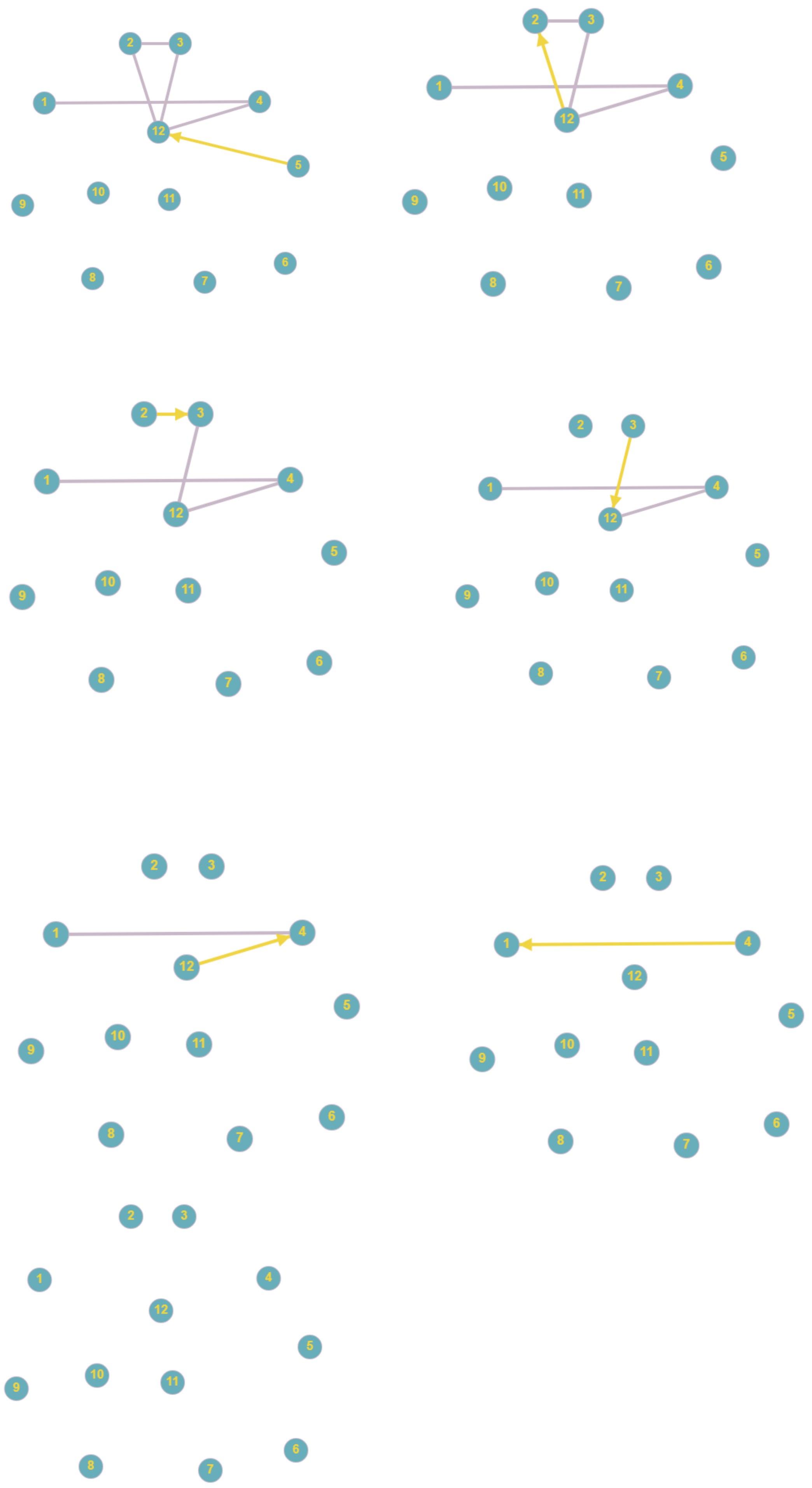








2 3 1 4 12 5 10 11 9 8 7 6
2 3 1 4 12 5 10 11 9 8 7 6
2 3 1 4 12 5 10 11 9 8 7 6
2 3 1 4 12 5 10 11 9 8 7 6
2 3 1 4 12 5 10 11 9 8 7 6
2 3 1 4 12 5 10 11 9 8 7 6



2 3 1 4 12 5 9 10 11 8 7 6
2 3 1 4 12 5 9 10 11 8 7 6
2 3 1 4 12 5 9 10 11 8 7 6
2 3 1 4 12 5 9 10 11 8 7 6
2 3 1 4 12 5 9 10 11 8 7 6
2 3 1 4 12 5 9 10 11 8 7 6
2 3 1 4 12 5 9 10 11 8 7 6

```cpp
#include <iostream>
using namespace std;
#define N 12
#define STACK_SIZE 100
int G[N][N] =
{
    {0,0,0,1,0,0,0,1,1,0,0,1},
    {0,0,1,0,0,0,0,0,0,0,0,1},
    {0,1,0,0,0,0,0,0,0,0,0,1},
    {1,0,0,0,1,1,1,0,0,0,1,1},
    {0,0,0,1,0,1,0,0,0,0,1,1},
    {0,0,0,1,1,0,1,0,0,0,1,0},
    {0,0,0,1,0,1,0,1,0,0,1,0},
    {1,0,0,0,0,0,1,0,1,1,0,0},
    {1,0,0,0,0,0,0,1,0,1,0,1},
    {0,0,0,0,0,0,0,1,1,0,1,1},
    {0,0,0,1,1,1,1,1,0,0,1,0,1},
    {1,1,1,1,1,0,0,0,1,1,1,0}};

int k;
int Stack[STACK_SIZE];

void Search(int V)
{
    int i;
    for(i = 0; i < N; i++)
        if(G[V][i])
        {
            G[V][i] = G[i][V] = 0;
            Search(i);
        }
    Stack[++k] = V;
}

int main()
{
    int T, p, q, s;
```

```cpp
    for(i = 0; i < N; i++)
        if(G[V][i])
        {
            G[V][i] = G[i][V] = 0;
            Search(i);
        }
    Stack[++k] = V;
}

int main()
{
    int T, p, q, s;
    int j, vv;

    T = 1;
    for(p = 0; p < N; p++)
    {
        s = 0;
        for(q = 0; q < N; q++)
        {
            s += G[p][q];
        }
        if(s%2) T = 0;}
    k = -1;
cout<<"start vertex: ";
cin>>vv;
vv--;
if(T)
    {
        Search (vv);
    for(j = 0; j <= k; j++)
            cout<<Stack[j]+1<<" ";}
else
cout<<"not Eulerian graph\n";
return 0;
}
```

Декілька можливих варіантів з різних точок

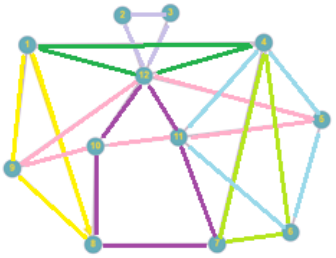
```
D:\code\lbixt|\bin\Debug\lbixt|.exe
start vertex: 1
1 9 10 12 11 10 8 9 12 3 2 12 5 11 7 8 1 12 4 11 6 7 4 6 5 4 1
Process returned 0 (0x0)   execution time : 2.308 s
Press any key to continue.
```

```
D:\code\ЫыхЁ|\bin\Debug\ЫыхЁ|.exe
start vertex: 5
5 12 11 7 6 11 10 12 3 2 12 9 10 8 9 1 12 4 11 5 6 4 7 8 1 4 5
Process returned 0 (0x0)   execution time : 1.259 s
Press any key to continue.
```

```
D:\code\ЇыхЁ|\bin\Debug\ЇыхЁ|.exe
start vertex: 12
12 11 10 12 9 10 8 9 1 8 7 11 5 12 3 2 12 4 11 6 7 4 6 5 4 1 12
Process returned 0 (0x0)   execution time : 1.648 s
Press any key to continue.
```

Елементарні цикли:

Виділила 7 простих циклів



Жовтий з ребрами (1,8),(8,9),(9,1)

Рожевий з ребрами (9,12),(12,5),(5,11),(11,10),(11,9)

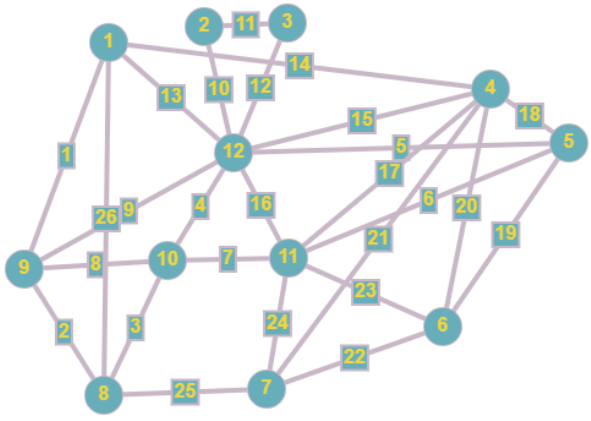
Фіолетовий з ребрами (12,10), (10,8), (8,7),(7,11),(11,12)

Зелений з ребрами (1,4),(4,12),(12,1)

Синій з ребрами (11,4),(4,5),(5,6),(6,11)

Світлозелений з ребрами (7,4),(4,6),(6,7)

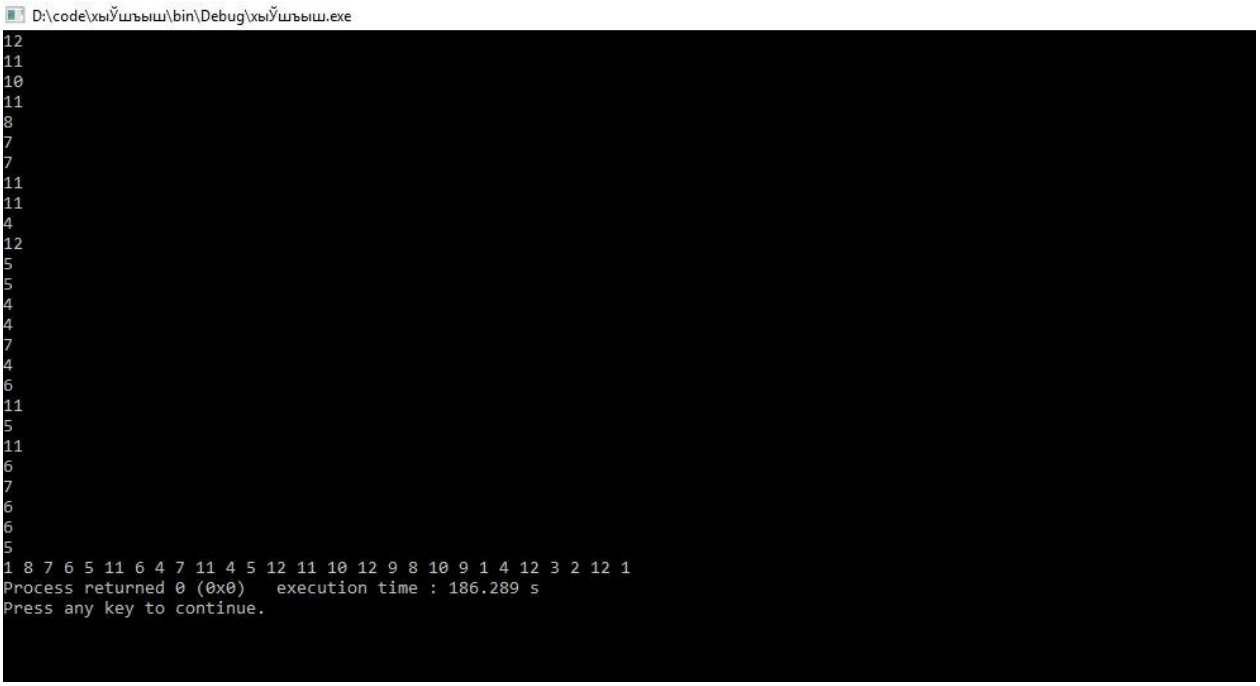
Бузковий з ребрами (2,3),(3,12),(12,2)



in.cpp

```cpp
#include <iostream>
#include <vector>
#include <stack>
#include <algorithm>
#include <list>
using namespace std;

vector < list<int> > graph;
vector <int> deg;
stack<int> head, tail;
int main()
{
//deg is the degree of a given vertex
  int n, a, x, y;
  cout<<"Enter count of tops / edges: ";
  cin >> n >> a; //12 26
  graph.resize(n + 1);
  deg.resize(n + 1);
  for (; a--;)
  {
    cin >> x >> y;
    graph[x].push_back(y);
    graph[y].push_back(x);
    ++deg[x];
    ++deg[y];
  }

if (any_of(deg.begin() + 1, deg.end(), [](int i) {return i & 1; }))
cout << "-1";           /*< no eulex cycle exists (all degrees must be even) */

  else
  {
    head.push(1);
    while (!head.empty())
    {
      while (deg[head.top()])
      {
```

```cpp
  {
    cin >> x >> y;
    graph[x].push_back(y);
    graph[y].push_back(x);
    ++deg[x];
    ++deg[y];
  }

if (any_of(deg.begin() + 1, deg.end(), [](int i) {return i & 1; }))
cout << "-1";           /*< no eulex cycle exists (all degrees must be even) */

  else
  {
    head.push(1);
    while (!head.empty())
    {
      while (deg[head.top()])
      {
        int v = graph[head.top()].back();
        graph[head.top()].pop_back();
        graph[v].remove(head.top());
        --deg[head.top()];
        head.push(v);
        --deg[v]; }

      while (!head.empty() && !deg[head.top()])
      {
        tail.push(head.top());
        head.pop(); }
    }
    /*< tail is the eulerian cycle */
    while (!tail.empty())
    {
      cout << tail.top() << ' ';
      tail.pop();
    } } }
```



Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

8. $(y \cdot x \cdot \bar{y}) \vee x \vee (y \cdot \overline{x \cdot \bar{x}})$

(y·x·$\bar{y}$) V x V (y·x·$\bar{x}$)

(x· F)V x V (y· $F\neg$)

(x· F)V x V (y· T)

F V x V y

F V x V y= x V y.