МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

З дисципліни

«Дискретна математика»

Виконав:

студент групи КН-113

Рябчук Андрій

Викладач:

Мельникова Н.І.

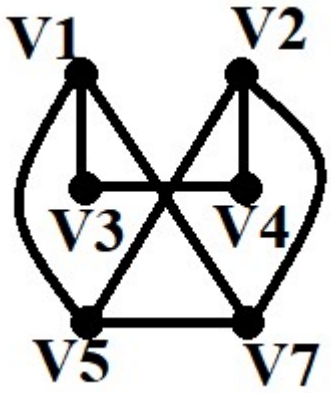
Львів-2019

Варіант №5

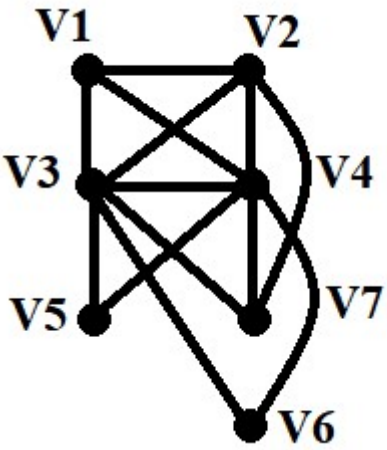
Завдання № 1

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 скадається з 3-х вершин в G1 6) добуток графів.

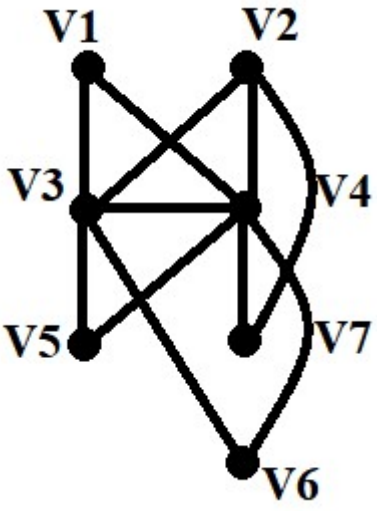
1)



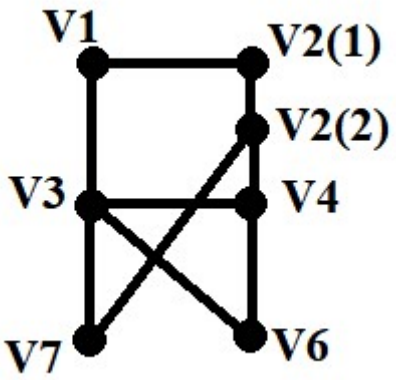
2)



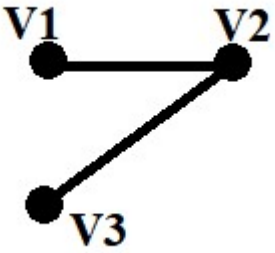
3)



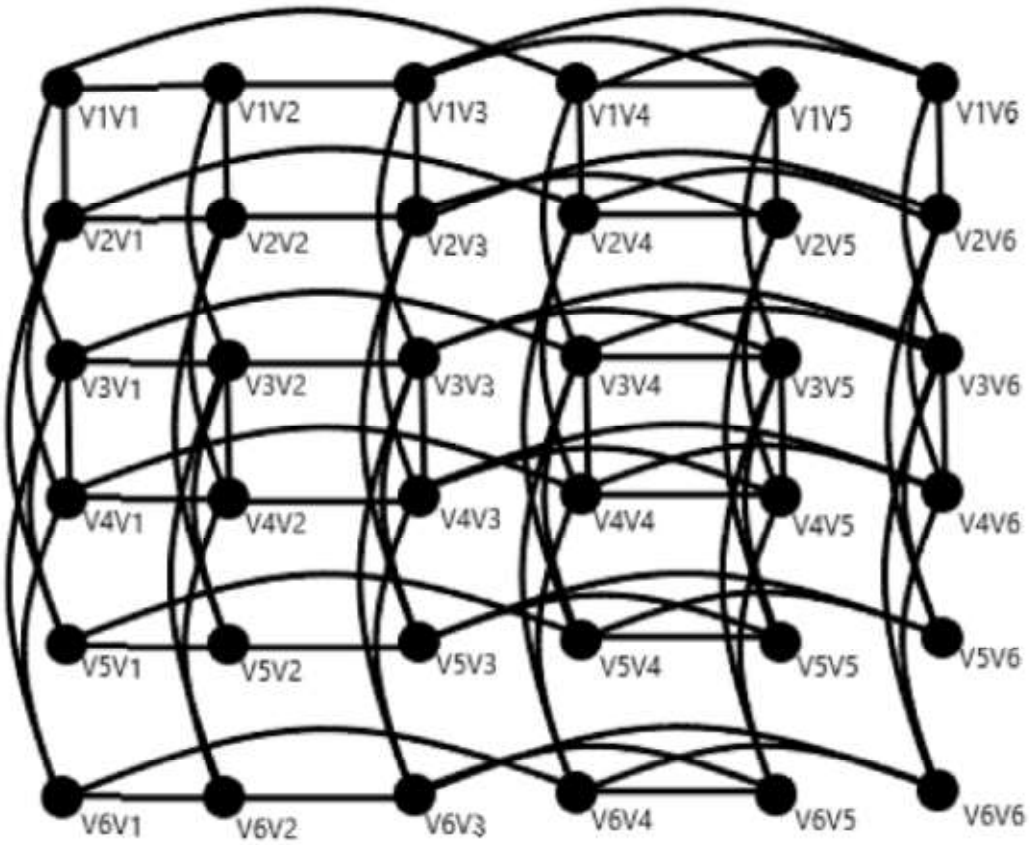
4)


V1
V2(1)
V2(2)
V3
V4
V7
V6

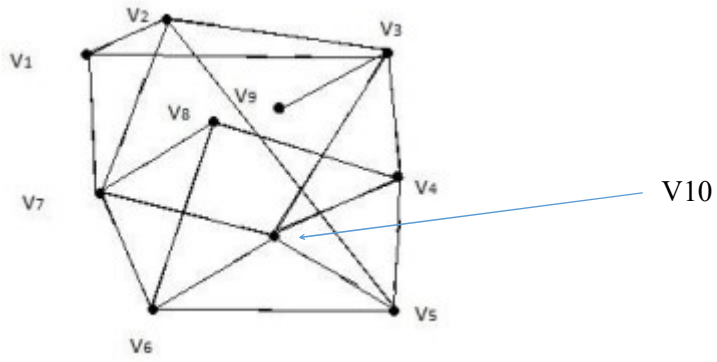
5)


V1
V2
V3

6)


V1V1
V1V2
V1V3
V1V4
V1V5
V1V6
V2V1
V2V2
V2V3
V2V4
V2V5
V2V6
V3V1
V3V2
V3V3
V3V4
V3V5
V3V6
V4V1
V4V2
V4V3
V4V4
V4V5
V4V6
V5V1
V5V2
V5V3
V5V4
V5V5
V5V6
V6V1
V6V2
V6V3
V6V4
V6V5
V6V6

Завдання № 2
Скласти таблицю суміжності для орграфа.



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	1	1	0	0	0	1	0	0	0
V2	1	0	1	0	1	0	1	0	0	0
V3	1	1	0	1	0	0	0	0	1	1
V4	0	0	1	0	1	0	0	1	0	1
V5	0	1	0	1	0	1	0	0	0	1
V6	0	0	0	0	1	0	1	1	0	1
V7	1	1	0	0	0	1	0	1	0	1
V8	0	0	0	1	0	1	1	0	0	0
V9	0	0	1	0	0	0	0	0	0	0
V10	0	0	1	1	1	1	1	0	0	0

Завдання № 3
Для графа з другого завдання знайти діаметр.

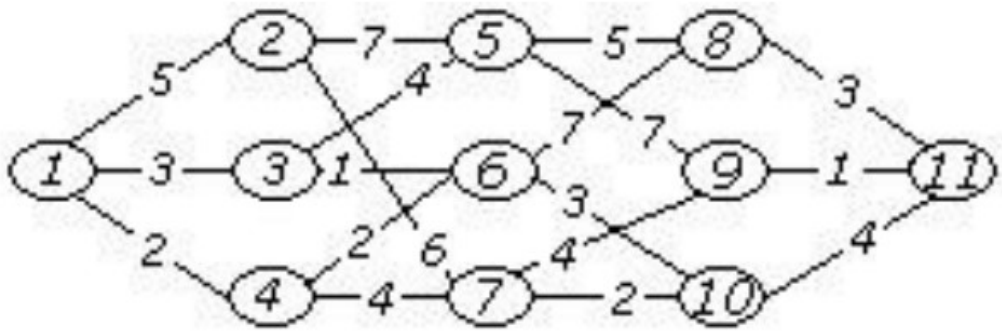
Діаметр - 3

Завдання №4

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

Вершина	DFS - номер	Вміст стеку
V1	1	V1
V2	2	V1V2
V3	3	V1V2V3
V10	4	V1V2V3V10
V4	5	V1V2V3V10V4
V8	6	V1V2V3V10V4V8
V6	7	V1V2V3V10V4V8V6
V7	8	V1V2V3V10V4V8V6 V7
-	-	V1V2V3V10V4V8V6
V5	9	V1V2V3V10V4V8V6 V5
-	-	V1V2V3V10V4V8V6
-	-	V1V2V3V10V4V8
-	-	V1V2V3V10V4
-	-	V1V2V3V10
-	-	V1V2V3
V9	10	V1V2V3V9
-	-	V1V2V3
-	-	V1V2
-	-	V1
-	-	0

Завдання №5
Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

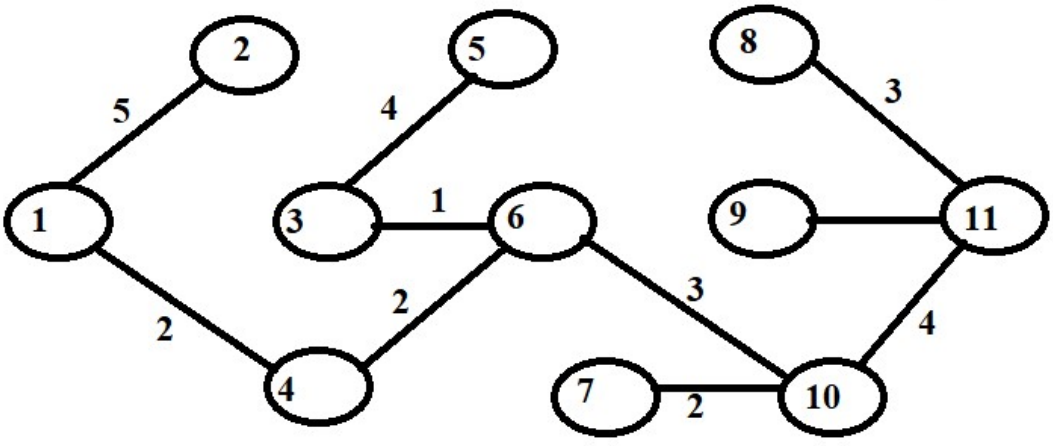


Алгоритм Краскала

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

Алгоритм Прима

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



Мінімальна вага - 27

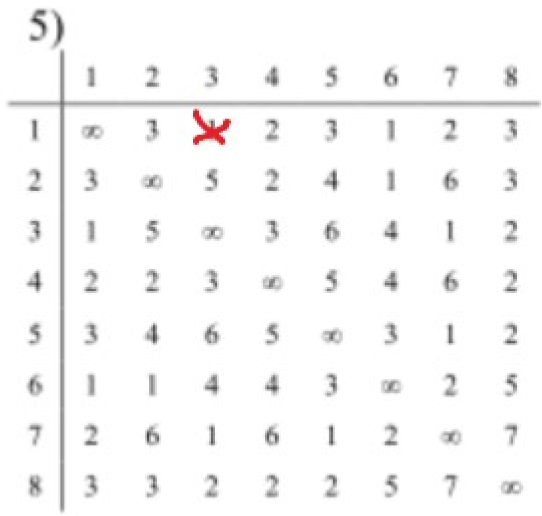
Завдання №6

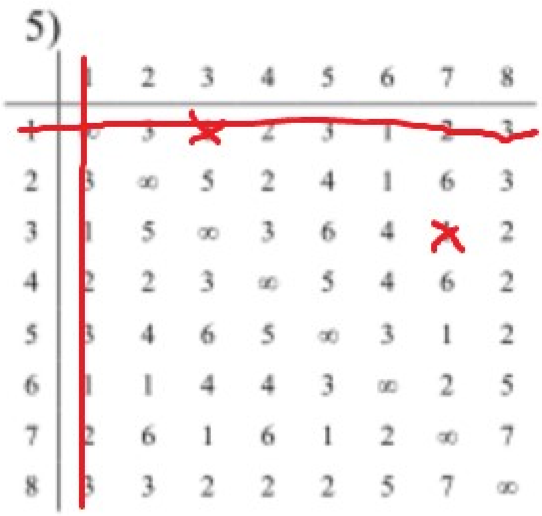
Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

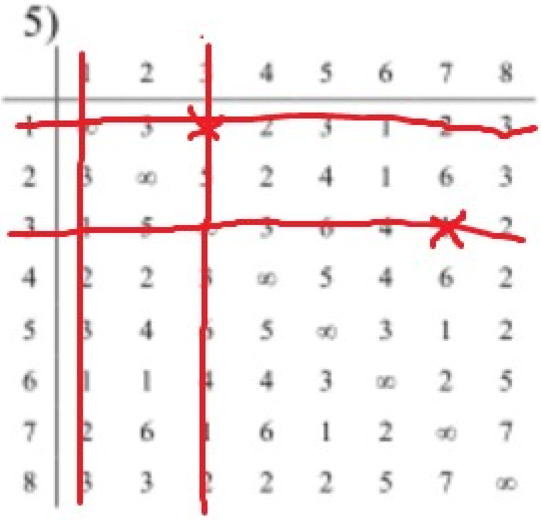
5)

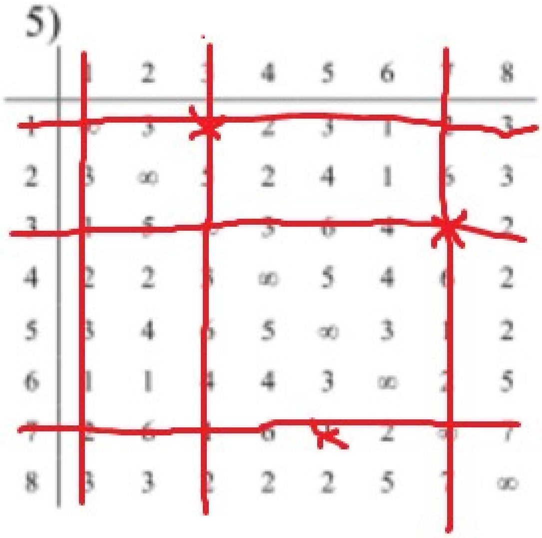
	1	2	3	4	5	6	7	8
1	∞	3	1	2	3	1	2	3
2	3	∞	5	2	4	1	6	3
3	1	5	∞	3	6	4	1	2
4	2	2	3	∞	5	4	6	2
5	3	4	6	5	∞	3	1	2
6	1	1	4	4	3	∞	2	5
7	2	6	1	6	1	2	∞	7
8	3	3	2	2	2	5	7	∞

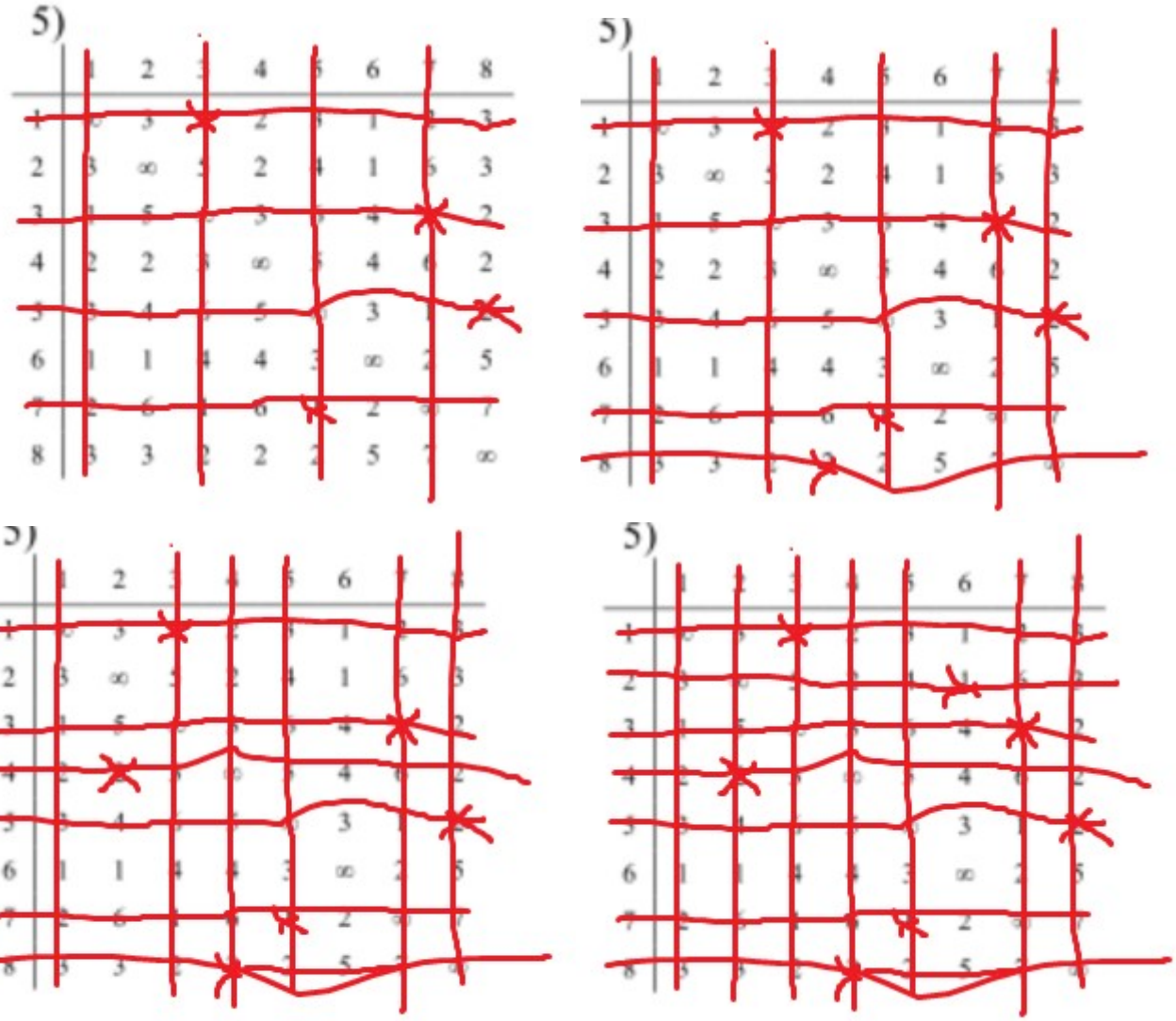
З першого рядка вибираємо ребро 3 та викреслюємо перший рядок перший стовпець











Шлях:

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

Вага : 1 + 1 +1+ 2+ 2 +2 +1+ 1 = 11

Вибравши на першому етапі вершину 6 , отримаємо цикли:

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

Вага : 1+1+2+2+2+1+1+3 = 13

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

Вага:1+1+2+2+2+1+1+1 = 11

Ми перевірили всі можливі цикли отже мінімальна вага - 11

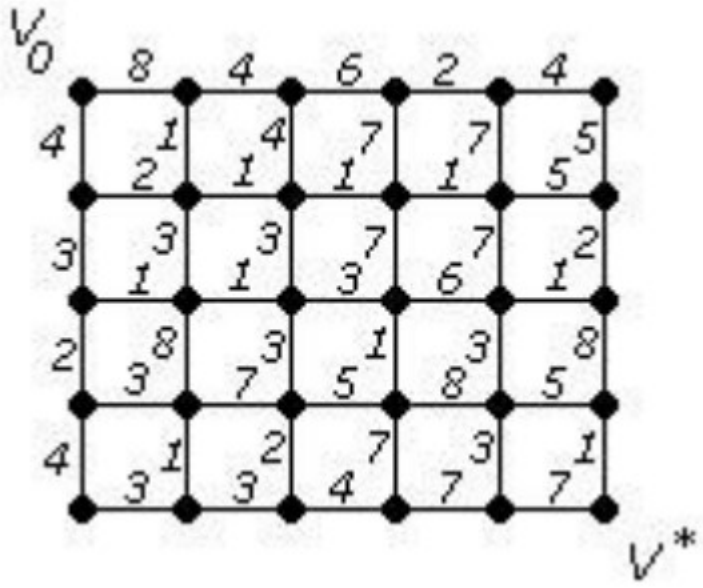
Деякі з оптимальних шляхів :

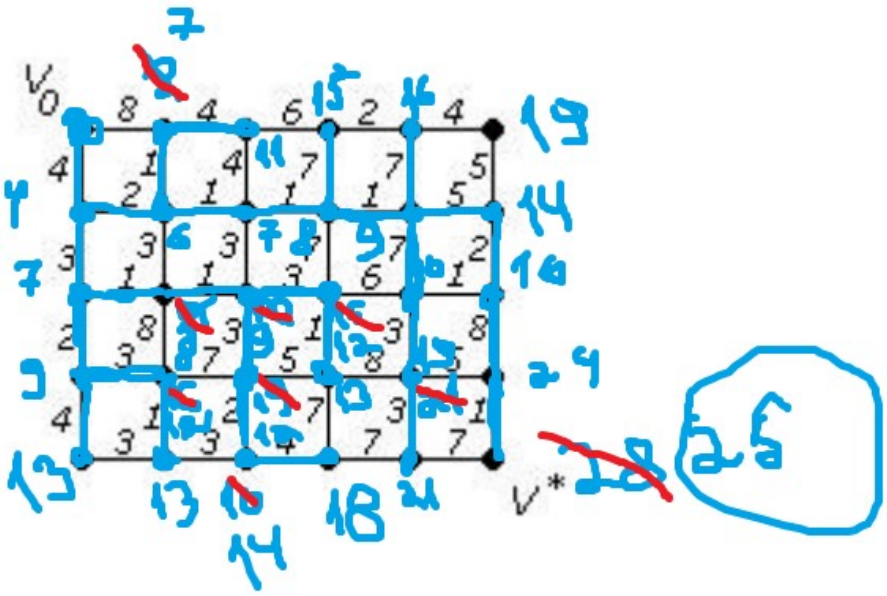
1->6->2->4->8->5->7->3->1

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

Завдання №7

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

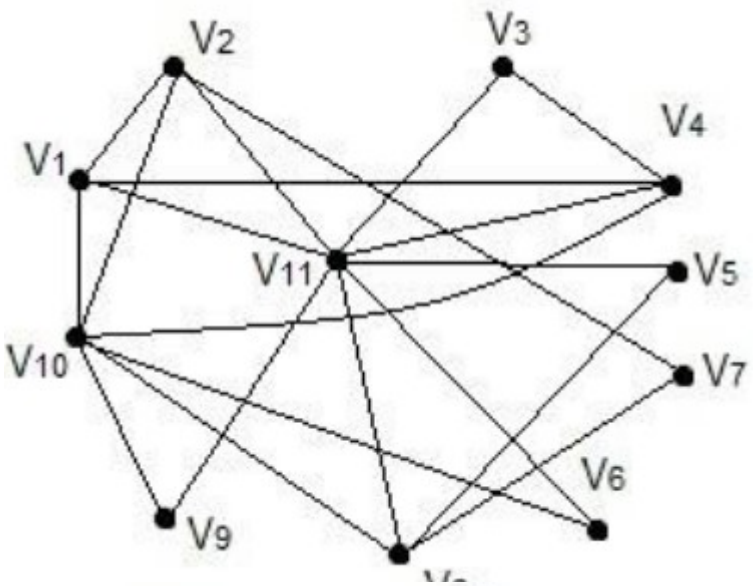




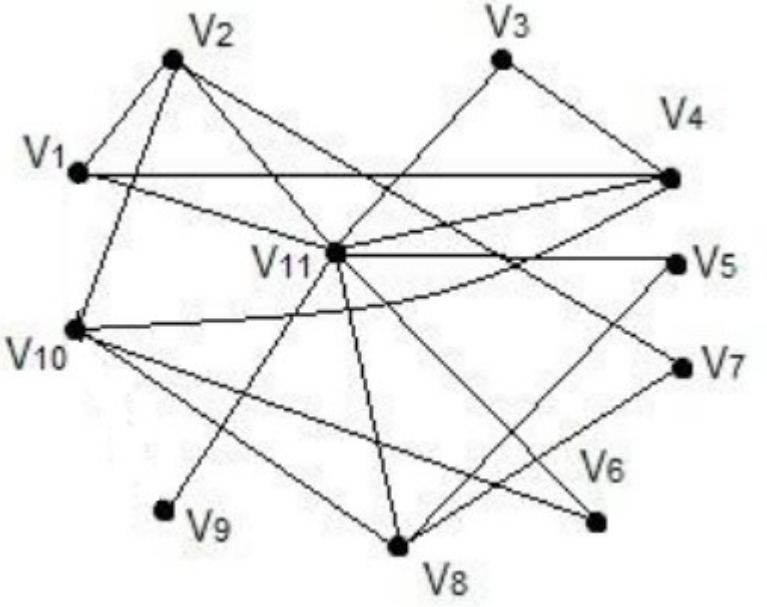
Найкоротший шлях - [v0,v6,v7,v8,v9,v10,v11,v17,v23,v29] - 25

Завдання №8

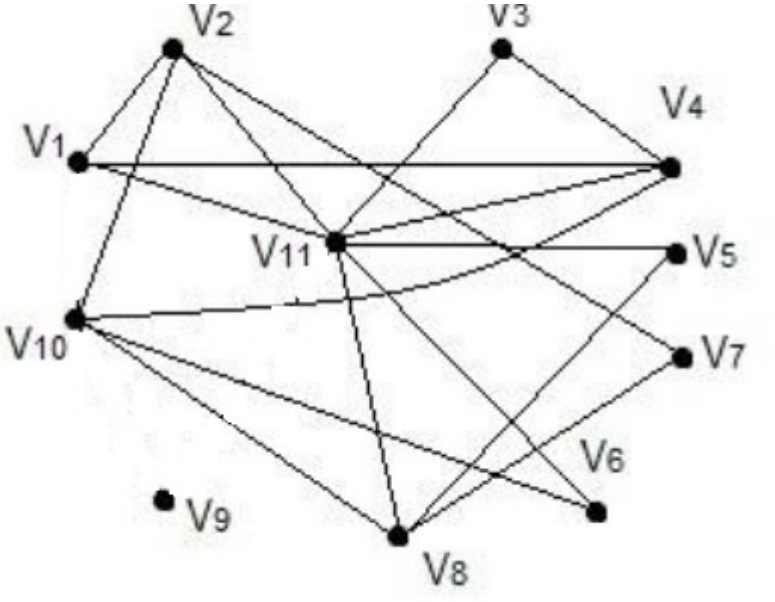
Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.



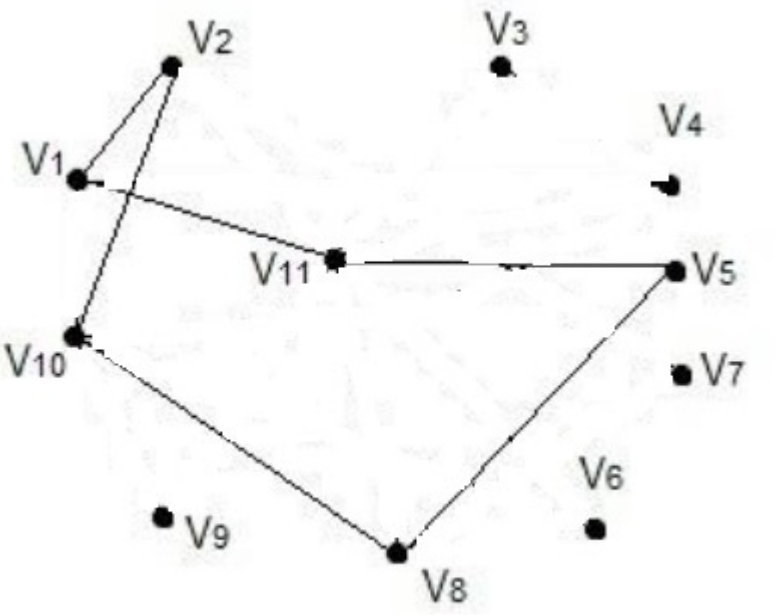
А)Флері



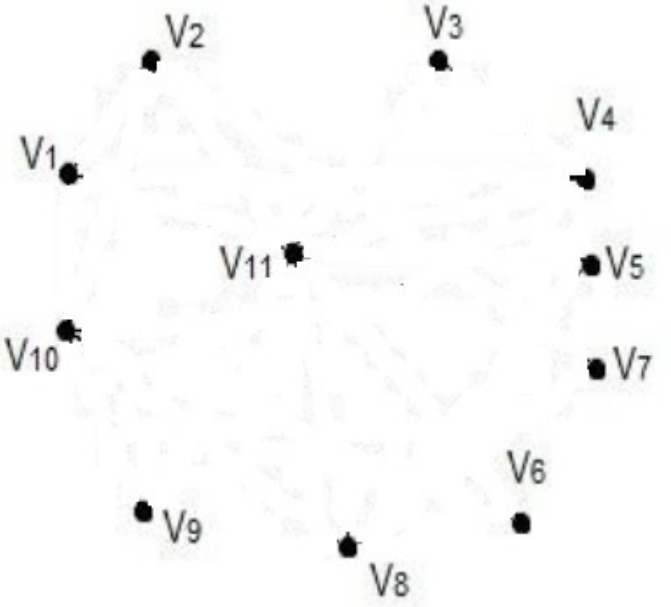
L = [V1,V10,V9]



L = [V1,V10,V9,V11]



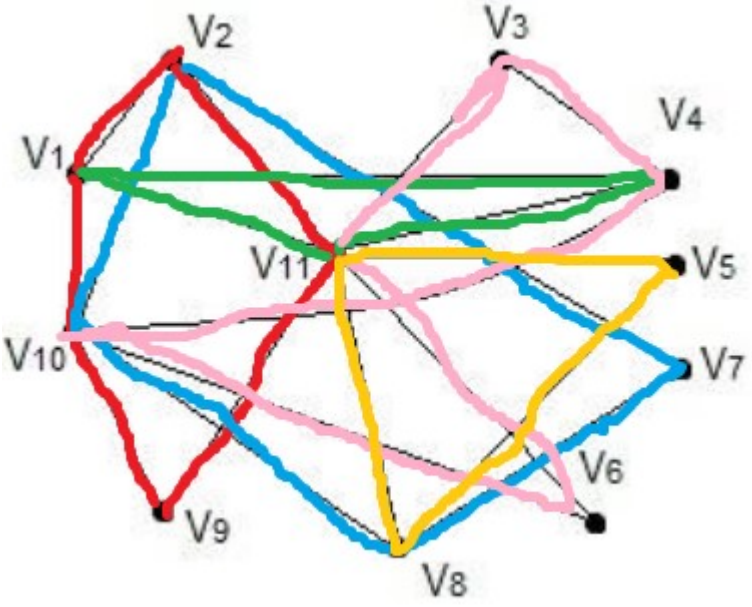
L=[V1,V10,V9,V11,V8,V7,V2,V11,V4,V3,
V11,V6,V10,V4,V1]



L=[V1,V10,V9,V11,V8,V7,V2,V11,V4,V3,
V11,V6,V10,V4,V1,V11,V5,V8,V10,V2,V1]

Ми пройшли всі ребра і повернулись в початкову точку отже цикл який ми отримали є Ейлеровим

Б)Елементарними циклами



Знайшли 5 циклів:

[V1,V2,V11,V9,V10,V1]
[V2,V10,V8,V7,V2]
[V1,V11,V4,V1]
[V11,V8,V5,V11]
[V11,V3,V4,V10,V6,V11]

Вставимо другий цикл в перший:
[V1,V2,V10,V8,V7,V2,V11,V9,V10,V1]
Тепер третій в перший:
[V1,V11,V4,V1,V2,V10,V8,V7,V2,V11,V9,V10,V1]
Пятий в четвертий і четвертий в перший:
[V1,V11,V8,V5,V11,V3,V4,V10,V6,V11,V4,V1,V2,V10,V8,V7,V2,V11,
V9,V10,V1] - цикл Ейлера

Завдання №9
Спростити формули (привести їх до скороченої ДНФ).

$$(x \to y)\cdot(y \to z) \to (z \to x)$$

Запишемо таблицю істинності для цієї формули:

X	Y	Z	X→Y	Y→Z	(X→Y)&(Y→Z)	Z→X	((X→Y)&(Y→Z))→(Z→X)
0	0	0	1	1	1	1	1
0	0	1	1	1	1	0	0
0	1	0	1	0	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1
1	0	1	0	1	0	1	1
1	1	0	1	0	0	1	1
1	1	1	1	1	1	1	1

Запишемо ДДНФ:

$$(xyz) \vee (xy\overline{z}) \vee (x\overline{y}z) \vee (x\overline{y}\,\overline{z}) \vee (\overline{x}y\overline{z}) \vee (\overline{x}\,\overline{y}z) \vee (\overline{x}\,\overline{y}\,\overline{z})$$

Використавши основні формули склеювання отримаємо:

$$(xy) \vee (\overline{x}\,\overline{z}) \vee (x\overline{y}) = x \vee (\overline{x} \wedge \overline{z}) = x \vee \overline{z}$$

Алгоритм «Іди в найближчий»

```
#include <iostream>
#include <windows.h>
#include <bits/stdc++.h>
using namespace std;
int consisting(string x,set <string> s){
int c = 0;
for(auto i : s){
	if (x == i)
		c = 1;
}
if (c ==1)
	return 1;
else
	return 0;
}
class edge {
	public :
		string vertices_1;
		string vertices_2;
		int weight;
};
int main()
{
	SetConsoleOutputCP(CP_UTF8);
	set < string > vertices;

	char *p;
	char * s = new char[255];

	map <string , int > vertice_weight;
	set <string> ban;
	gets(s);
	p = strtok(s," ");
	vertices.insert(p);
	string current = p;
	while(p){
		vertices.insert(p);
		vertice_weight.insert(pair <string,int>(p,-1));
		p = strtok(NULL," ");

	}
	vector <edge> edges;
	while(1){
		edge x;
		cin>>x.vertices_1;
		if (x.vertices_1=="end")
			break;
		cin>>x.vertices_2>>x.weight;
		edges.push_back(x);
	}
```

```
	int min;
	string next,current_city,start;
	current_city = start;
	int sum,sum_min = 99999999;
	vector <string> way;
	for(auto start:vertices){
		vector <string> buf;
		buf.push_back(start);
		current_city = start;
		sum = 0;
		int n =0;
		ban.clear();
		while(1){
			min = 99999999;
			for (auto eedge : edges){
				if (eedge.vertices_1 == current_city){
					if   ((eedge.weight<   min)&&(!consisting(eedge.vertices_2,ban)||((n
==4)&&(eedge.vertices_2==start)))){
						min = eedge.weight;
						next = eedge.vertices_2;
					}
				}
			}
			buf.push_back(next);
			if (min == 99999999){
				cout<<"шлях неможливий\n";
				break;
			}
			sum += min;
			n++;
			cout<<sum<<current_city<<next<<endl;
			ban.insert(current_city);
			current_city = next;
			if (current_city ==start){
				if (sum < sum_min){
					sum_min = sum;
					way.clear();
					for (auto vert: buf ){
						way.push_back(vert);

					}
				}
				break;
			}

		}
	}
	cout << sum_min<<endl<<"[";
	for(auto i : way){
		cout<<i<<",";
	}
	cout<<"] - Мінімальний шлях та його вага";

	return 0;
}
```

Вхідні дані:

v1 v2 v3 v4 v5 v6 v7 v8
v1 v2 3
v1 v3 1
v1 v4 2
v1 v5 3
v1 v6 1
v1 v7 2
v1 v8 3
v2 v1 3
v2 v3 5
v2 v4 2
v2 v5 4
v2 v6 1
v2 v7 6
v2 v8 3
v3 v1 1
v3 v2 5
v3 v4 3
v4 v5 6
v4 v6 4
v4 v7 1
v4 v8 2
v5 v1 3
v5 v2 4
v5 v3 6
v5 v4 5
v5 v6 3
v5 v7 1
v5 v8 2
v6 v1 1
v6 v2 1
v6 v3 4
v6 v4 4
v6 v5 3
v6 v7 2
v6 v8 5
v7 v1 2
v7 v2 6
v7 v3 1
v7 v4 6
v7 v5 1
v7 v6 2
v7 v8 7
v8 v1 3
v8 v2 3
v8 v3 2
v8 v4 2
v8 v5 2
v8 v6 5
v8 v7 7

Результат роботи:

```
3v1v6
4v6v2
6v2v4
8v4v8
10v8v5
11v5v7

2v8v3
3v3v1
4v1v6
5v6v2
7v2v4
8v4v7
9v7v5
11v5v8

11
[v3,v1,v6,v2,v4,v7,v5,v8,v3,] - Мінімальний шлях та його вага
Process returned 0 (0x0)   execution time : 4.252 s
Press any key to continue.
```

Алгоритм Дейкстри

```
#include <iostream>
#include <windows.h>
#include <bits/stdc++.h>
using namespace std;
int consisting(string x,set <string> s){
int c = 0;
for(auto i : s){
	if (x == i)
		c = 1;
}
if (c ==1)
	return 1;
else
	return 0;
}
class edge {
	public :
		string vertices_1;
		string vertices_2;
		int weight;
};
int main()
{
	SetConsoleOutputCP(CP_UTF8);
	set < string > vertices;

	char *p;
	char * s = new char[255];

	map < string , vector<string> > derevo;
	map <string , int > vertice_weight;
	set <string> ban;
	gets(s);
	string v0,ve;
	cin>>v0>>ve;
```

```
        p = strtok(s," ");
        vertices.insert(p);
        string current = p;
        vector <string> rebro;
        rebro.push_back("0");
        rebro.push_back("0");

        while(p){
                vertices.insert(p);
                derevo.insert(pair<string,vector<string>> (p,rebro));
                vertice_weight.insert(pair <string,int>(p,-1));
                p = strtok(NULL," ");

        }
        vector <edge> edges;
        while(1){
                edge x;
                cin>>x.vertices_1;
                if (x.vertices_1=="end")
                        break;
                cin>>x.vertices_2>>x.weight;
                edges.push_back(x);
        }


//      string current = "v0";
        vertice_weight[v0] = 0;
        int x = 0;

        while(x < vertices.size()){
                for (auto edge : edges){
                        if ((edge.vertices_1 == current)&&(!consisting(edge.vertices_2,ban))){
                                for (auto p = vertice_weight.begin();p != vertice_weight.end(); p++){
                                        if (vertice_weight[edge.vertices_2] == -1){
                                                        vertice_weight[edge.vertices_2] = edge.weight + vertice_weight[current];
                                                        derevo[edge.vertices_2][0] = edge.vertices_1;
                                                        derevo[edge.vertices_2][1] = edge.vertices_2;

                                        }
                                        else if ( edge.weight+vertice_weight[current] <    vertice_weight[edge.vertices_2] ){
                                                vertice_weight[edge.vertices_2] = edge.weight + vertice_weight[current];
                                                derevo[edge.vertices_2][0] = edge.vertices_1;
                                                derevo[edge.vertices_2][1] = edge.vertices_2;
                                        }
                                }
                        }
                        if ((edge.vertices_2 == current)&&(!consisting(edge.vertices_1,ban))){
                                for (auto p = vertice_weight.begin();p != vertice_weight.end(); p++){
                                        if (vertice_weight[edge.vertices_1] == -1){
                                                vertice_weight[edge.vertices_1] = edge.weight + vertice_weight[current];
                                                derevo[edge.vertices_1][0] = edge.vertices_2;
                                                derevo[edge.vertices_1][1] = edge.vertices_1;
                                        }
                                        else if ( edge.weight +vertice_weight[current]<    vertice_weight[edge.vertices_1] ){
                                                vertice_weight[edge.vertices_1] = edge.weight + vertice_weight[current];
                                                derevo[edge.vertices_1][0] = edge.vertices_2;
                                                derevo[edge.vertices_1][1] = edge.vertices_1;
                                        }
                                }
                        }


                }

                ban.insert(current);
                int mini = 999999999;
                for(auto v : vertice_weight){
```

```
                if ((v.second < mini)&&(!consisting(v.first ,ban))&&(v.second != -1)){
                    mini = v.second;
                    current = v.first;
                }
            }
            x++;
        }
        cout<<endl;
        cout<<endl;
        cout<<"Мінімальна відстань до "<<ve<<" - "<<vertice_weight[ve]<<endl;
        vector <string> min_chain;
        min_chain.push_back(ve);
        string cur = ve;

        while(cur!=v0){
            for(auto i : derevo){
                if (i.second[1] == cur){
                    cur = i.second[0];
                    min_chain.push_back(cur);
                }
            }
        }
        cout<<"Шуканий найкоротший ланцюг: [";
        for (int i = min_chain.size()-1 ; i > 0; i--)
            cout<<min_chain[i]<<",";
        cout<<min_chain[0]<<"]";
        return 0;
}
```

Вхідні дані: ребра і вершини графа з 7 завдання

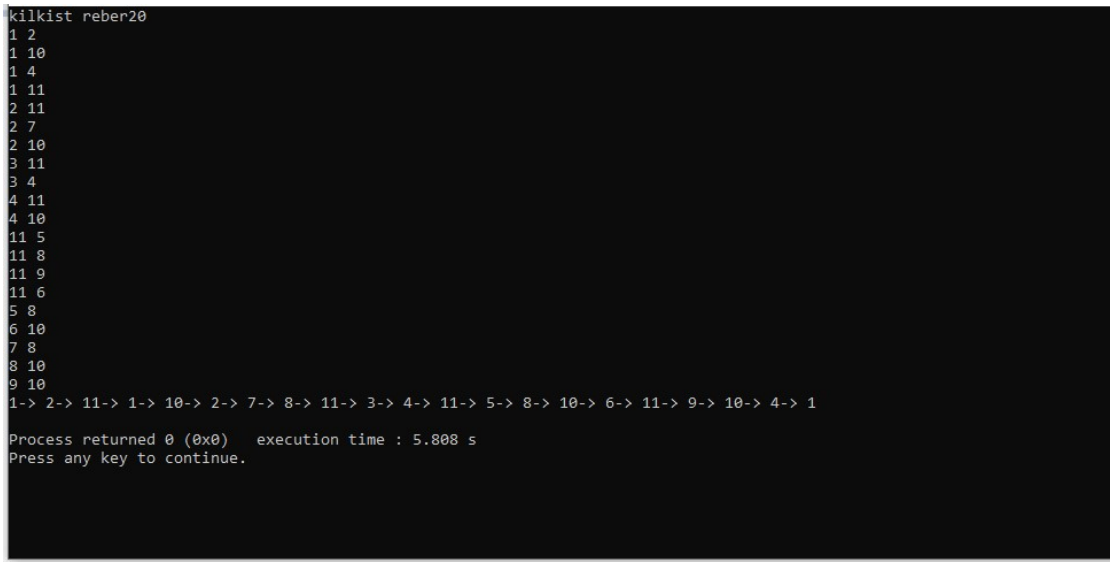
Результат роботи програми:

```
v12 v18 2
v18 v24 4
v5 v11 5
v11 v17 2
v17 v23 8
v23 v29 1
v1 v7 1
v7 v13 3
v13 v19 8
v19 v25 1
v2 v8 4
v8 v14 3
v14 v20 3
v20 v26 2
v3 v9 7
v9 v15 7
v15 v21 1
v21 v27 7
v4 v10 7
v10 v16 7
v16 v22 3
v22 v28 3
end


Мінімальна відстань до v29 - 25
Шуканий найкоротший ланцюг: [v0,v6,v7,v8,v9,v10,v11,v17,v23,v29]
Process returned 0 (0x0)   execution time : 5.942 s
Press any key to continue.
```

Алгоритм Флері

```
#include <iostream>
#include <string.h>
#include <algorithm>
#include<windows.h>
#include <set>
#include <list>

using namespace std;

class Graph

{

int V;

list<int>* adj;

public:

Graph(int V) { this->V = V; adj = new list<int>[V+1]; }
};

void Graph::printcycle()

{

int u = 1;

for (int i = 1; i <= V; i++)

if (adj[i].size() & 1)

{

u = i; break;

}

printEulerUtil(u);

printEulerUtil(u);

cout << endl;

}

void Graph::printEulerUtil(int u)

{

list<int>::iterator i;

for (i = adj[u].begin(); i != adj[u].end(); ++i)

{

int v = *i;

if (v != -1 && isValidNextEdge(u, v))

{
return count;

}

int main()

{

int number_of_edgel,u,v;
cout<<"kilkist reber";
cin>>number_of_edgel;
Graph gl(number_of_edgel);
for(int i = 0; i<number_of_edgel; i++){
    cin>>u>>v;
    gl.addEdge(u,v);
}
cout<<"1";
gl.printcycle();
return 0;

}
```

Результат програми:



Алгоритм Прима

```cpp
#include <iostream>

using namespace std;

int main()
{
    int x, y;
    int u, v;
    int number;
    int i, j;
    int ne = 1;
    int scan [12] = {0};
    int minimal, minimall = 0;
    int cost [12][12];
    int path [100] = {0};
    int path_index = 0;

    cout <<"Введіть кількість вершин:  ";
    cin >> number ;
    cout <<"Введіть матрицю суміжності \n";


    for(i = 1; i < number ; i++)

    for(i = 1; i <= number ; i++)
        for(j = 1; j <= number ; j++)
        {
            cin >> cost[i][j];
            if(cost[i][j] == 0)
                cost[i][j] = 999;
        }
    scan [1] = 1;
    cout << endl;

    while(ne < number )
    {
        for (i = 1, minimal = 999; i <= number; i++)
            for (j = 1;j <= number; j++)
                if (cost [i][j] < minimal)
                    if (scan [i] != 0)
                    {
                        minimal = cost [i][j];
                        x = u = i;
                        y = v = j;
                    }
        if (scan [u] ==0 || scan [v] == 0)
        }
        cost[x][y] = cost[y][x] = 999;
    }


    cout << endl;
    cout << 1 << " -> ";

    for (int i = 0; i < number-1; i++)
    {
        cout << path[i];
        if (i < number-2) {
            cout << " -> ";
        }
    }
    cout << endl;
    cout << "Minimal cost: " << endl;
    cout << minimall;

    cin.get();
    cin.get();
```

```
╨Æ╨▓╨╡╨┤╤û╤é╤î ╨║╤û╨╗╤î╨║╤û╤ü╤é╤î ╨▓╨╡╤Ç╤ê╨╕╨╜:  11
╨Æ╨▓╨╡╨┤╤û╤é╤î ╨╝╨░╤é╤Ç╨╕╤å╤Ä ╤ü╤â╨╝╤û╨╢╨╜╨╛╤ü╤é╤û
0 5 3 2 0 0 0 0 0 0 0
5 0 0 0 7 0 6 0 0 0 0
3 0 0 0 4 1 0 0 0 0 0
2 0 0 0 0 2 4 0 0 0 0
0 7 4 0 0 0 0 5 7 0 0
0 0 1 2 0 0 0 7 0 3 0
0 6 0 4 0 0 0 0 4 2 0
0 0 0 0 5 7 0 0 0 0 3
0 0 0 0 7 0 4 0 0 0 1
0 0 0 0 0 3 2 0 0 0 4
0 0 0 0 0 0 0 3 1 4 0


1  1  42
2  4  62
3  6  31
4  6  103
5  10  72
6  3  54
7  7  94
8  9  111
9  11  83
10  1  25
1 -> 4 -> 6 -> 3 -> 10 -> 7 -> 5 -> 9 -> 11 -> 8 -> 2
Minimal cost:
27
```

Алгоритм Краскала

```
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include<windows.h>
using namespace std;

const int q = 11;
int BuildTrees(int n, int A[q][q]);
void DeleteDuplicates(int n, int A[q][q]);
int InDifferTrees(int n, int A[q][q], int first, int second);
void AddToTheTree(int n, int A[q][q], int first, int second);

int main()
{
    SetConsoleOutputCP(CP_UTF8);
  int A[11][11] =
  { 0 ,5, 3, 2, 0, 0, 0, 0, 0 ,0,
    5 ,0 ,0, 0, 7, 0, 6, 0, 0, 0,
    3, 0, 0, 0, 4, 1, 0, 0 ,0, 0 ,0,
    2, 0, 0, 0, 0, 2 ,4, 0, 0, 0, 0,
    0, 7 ,4, 0, 0, 0, 0, 5, 7, 0, 0,
    0, 0, 1, 2, 0, 0, 0,7, 0, 3, 0,
    0  6  0  4  0  0  0  0  4  2  0
   0, 0, 0 ,0, 0, 0, 0, 3, 1, 4, 0 };

  DeleteDuplicates(11, A);
  for (int i = 1; i <= 7; i++){
    cout << "\nВузли з вагою: " << i << ": ";
    for (int j = 1; j <= 11; j++){
      for (int k = 1; k <= 11; k++){
        if (A[j - 1][k - 1] == i){
          cout << " " << j << "-" << k;;
        }
      }
    }
  }
  cout << "\n";

  int B[11][11];
  BuildTrees(11, B);
  cout << "\n\nНове дерево: ";//вага 7 - максимальна вага
  for (int i = 1; i <= 7; i++){
    //перший вузол
    for (int j = 1; j <= 11; j++){
      //другий вузол
      for (int k = 1; k <= 11; k++)

void DeleteDuplicates(int n, int A[q][q]) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      if (j < i) {
        A[i][j] = 0;
      }
    }
  }
}

int BuildTrees(int n, int A[q][q]) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      A[i][j] = 0;
    }
  }
  for (int i = 0; i < n; i++) {
    A[i][i] = i + 1;
  }
  return A[n][n];
}

void AddToTheTree(int n, int A[q][q], int first, int second) {
  int scndLine;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      if (A[i][j] == second) {
        scndLine = i;
      }
    }
  }
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      if (A[i][j] == first) {
        for (int k = 0; k < n; k++) {
          if (A[scndLine][k]) {
            A[i][k] = A[scndLine][k];
            A[scndLine][k] = 0;
          }
        }
      }
    }
  }
}
```

```
int InDifferTrees(int n, int A[q][q], int first, int second){
  int temp1, temp2;
  for (int i = 0; i < n; i++){
    temp1 = 0;
    temp2 = 0;
    for (int j = 0; j < n; j++){
      if (A[i][j] == first){
        temp1 = 1;
      }
    }
    for (int k = 0; k < n; k++){
      if (A[i][k] == second){
        temp2 = 1;
      }
    }
    if (temp1 && temp2){
      return 0;
    }
  }
  return 1;
}
```

```
Вага: 1:  3-6 9-11
Вага: 2:  1-4 4-6 7-10
Вага: 3:  1-3 6-10 8-11
Вага: 4:  3-5 4-7 7-9 10-11
Вага: 5:  1-2 5-8
Вага: 6:  2-7
Вага: 7:  2-5 5-9 6-8


остове дерево:  3-6 9-11 1-4 4-6 7-10 6-10 8-11 3-5 7-9 1-2
Process returned 0 (0x0)   execution time : 0.064 s
Press any key to continue.
```

Алгоритм пошуку вглиб

```
#include <iostream>
#include <string>
#include <sstream>
#include <windows.h>
using namespace std;

struct vershina
{
 bool dfs = false;
};
struct rebro
{
 int v1;
 int v2;
};

int leng(string str)
{
 int i = 0;

 while (str[i] != '\0')
 {
  i++;
 }
 return i;
}
int correct(int m, int n)
{
 int c = 0;
 bool count = false;
 string str;
 stringstream ss;
 while (count == false)
 {
  cin >> str;
  for (int i = 0; i < leng(str); i++)
  {
   if (!isdigit(str[i]))
   {
    if (i == 0 && str[i] == '-')
    {
     count = true;
    }
    else
    {
     count = false;
     break;
    }
   }
   else
   {
    count = true;
   }
  }

  if (count == true)
  {
   ss << str;
   ss >> c;
   ss.clear();

   if (c < m || c > n)
   {
    count = false;
   }
   else
   {
    count = true;
   }
  }
  if (count == false)
  {
   cout << "Error! Try again!" << endl;
  }
  str = "";
 }

 return c;
}
```

```
void input(rebro *reb, int n, int m)
{
 for (int i = 0; i < n; i++)
 {
  cout << "Введіть першу вершину, інцидентну ребру №" << i + 1 << ": ";
  reb[i].v1 = correct(1, m);
  cout << "Введіть другу вершину, інцидентну ребру №" << i + 1 << ": ";;
  reb[i].v2 = correct(1, m);
  cout << endl;
 }
}

int main()
{
            SetConsoleOutputCP(CP_UTF8);
 int n, m, p;
 int begin;
 int count = 0;
 int t = 0;
 int head = 0;

 cout << "Введіть кількість ребер у графі: ";
 n = correct(1, 1000);
 cout << "Введіть кількість вершин у графі: ";
 m = correct(1, 1000);
 cout << endl;

 int *vec = new int[m];
 rebro *reb = new rebro[n];
 vershina *v = new vershina[m];

 input(reb, n, m);

 cout << "З якої вершини почати обхід? ";
 begin = correct(1, m);

 vec[0] = begin;
 v[begin - 1].dfs = true;
 count++;

 cout << "Якщо ви хочете зробити обхід вглиб натисніть 1, обхід вшир - натисніть 2: ";
 p = correct(1, 2);

 switch (p)
 {
  case 1:
  {
   while (count != 0)
   {
    for (int i = 0; i < n; i++)
    {

     if ((vec[count - 1] == reb[i].v1 && v[reb[i].v2 - 1].dfs == false) || (vec[count - 1] == reb[i].v2 && v[reb[i].v1 - 1].dfs == false))
     {
      t++;
     }
    }

    if (t == 0)
    {
     count--;
    }
    else
    {
     for (int i = 0; i < n; i++)
     {
      if (vec[count - 1] == reb[i].v2 && v[reb[i].v1 - 1].dfs == false)
      {
       vec[count] = reb[i].v1;
       v[reb[i].v1 - 1].dfs = true;

       count++;
       goto point;
      }

      if (vec[count - 1] == reb[i].v1 && v[reb[i].v2 - 1].dfs == false)
      {
       vec[count] = reb[i].v2;
       v[reb[i].v2 - 1].dfs = true;
```

```
            count++;
            goto point;
           }
          }
        }
      point:;
        for (int i = 0; i < count; i++)
        {
          cout << vec[i] << " ";
        }
        if (count != 0)
        {
          cout << endl;
        }
        t = 0;
      }

      cout << "Стек пустий" << endl;
      break;
    }
    case 2:
    {
      while (head != m)
      {
        for (int i = head; i < n; i++)
        {

          if ((vec[head] == reb[i].v1 && v[reb[i].v2 - 1].dfs == false) || (vec[head] == reb[i].v2 && v[reb[i].v1 - 1].dfs == false))
          {
            t++;
          }
        }

        if (t == 0)
        {
          head++;
        }
        else
        {
          for (int i = head; i < n; i++)
          {
            if (vec[head] == reb[i].v2 && v[reb[i].v1 - 1].dfs == false)
            {
              vec[count] = reb[i].v1;
              v[reb[i].v1 - 1].dfs = true;

              count++;
              goto point1;
            }

            if (vec[head] == reb[i].v1 && v[reb[i].v2 - 1].dfs == false)
            {
              vec[count] = reb[i].v2;
              v[reb[i].v2 - 1].dfs = true;

              count++;
              goto point1;
            }
          }
        }
      point1:;
        for (int i = head; i < count; i++)
        {
          cout << vec[i] << " ";
        }
        if (head != m)
        {
          cout << endl;
        }
        t = 0;
      }

      cout << "Черга порожня" << endl;
      break;
    }
  }
  return 0;
}
```