МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА“

Кафедра систем штучного інтелекту

Розрахункова робота

З дисципліни

«Дискретна математика»

Виконала:

Студентка групи КН-115

Рокицька Анастасія

Викладач:

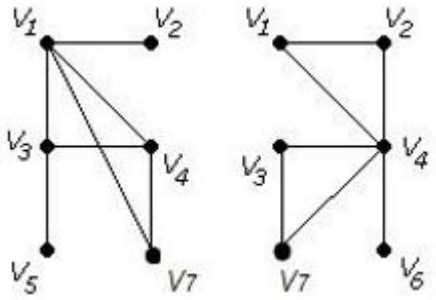
Мельникова Н.І.

Львів – 2019р.

ВАРІАНТ 20

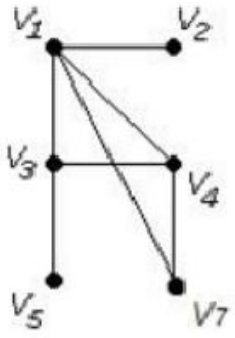
Завдання №1

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.

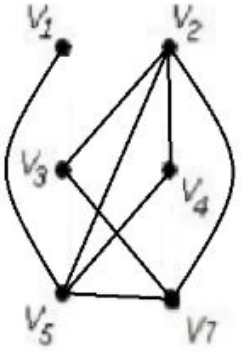


1)

Граф:

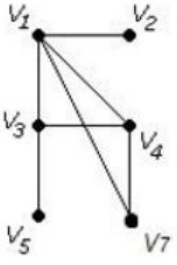


Доповнення до графа:

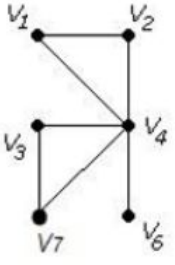


2)

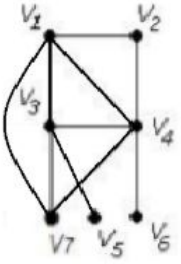
G1



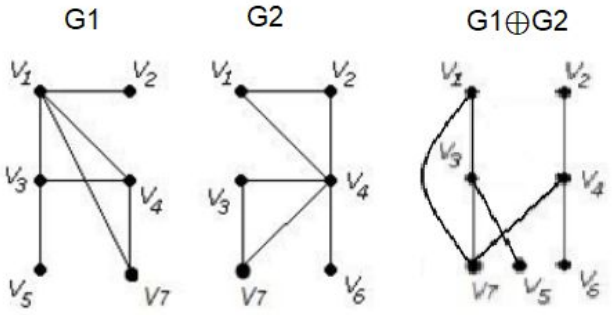
G2



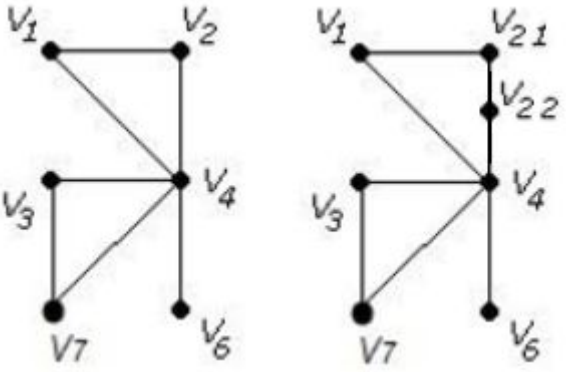
G1UG2



3)

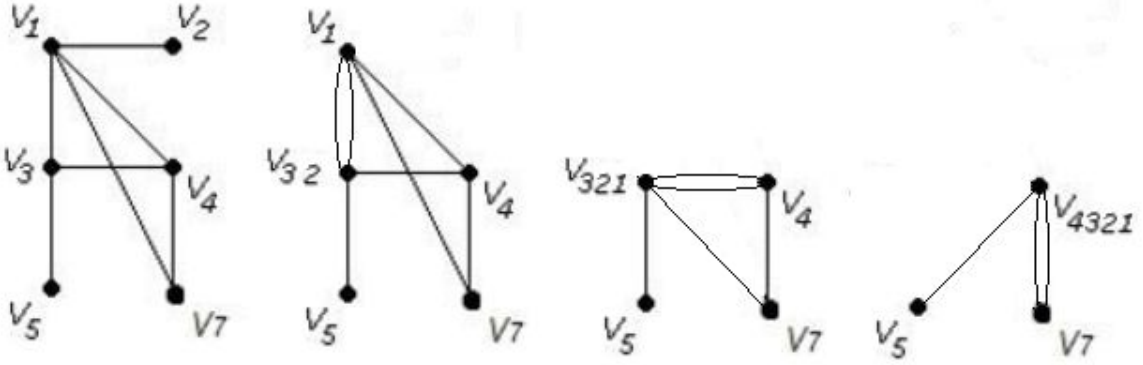


4) Розщеплення вершини V2

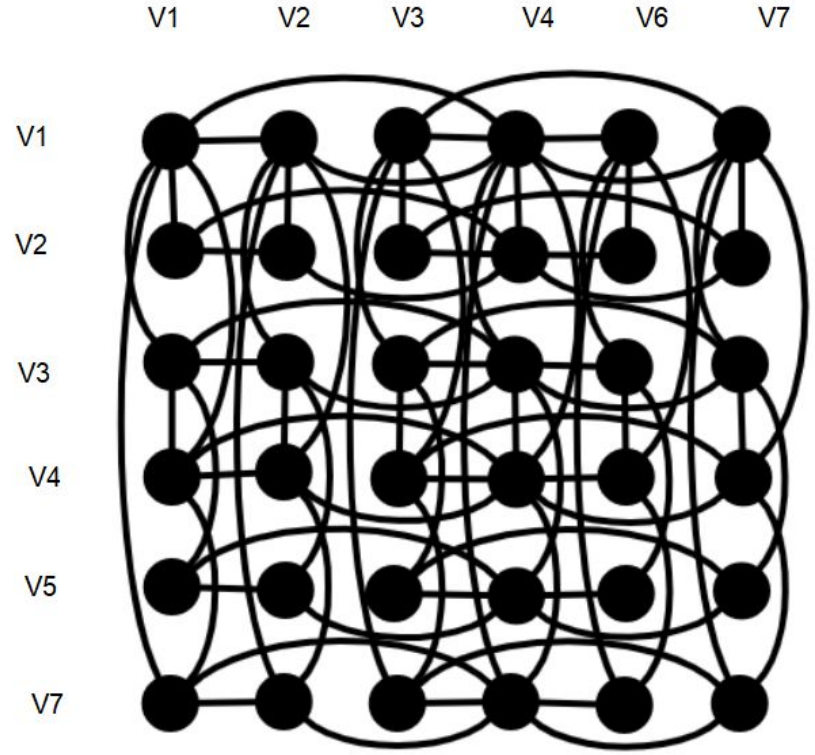


5)

G1: V1={V1,V2,V3,V4,V5,V7}
A : V2={V1,V2,V3}
G1\A={V4321,V5,V7}

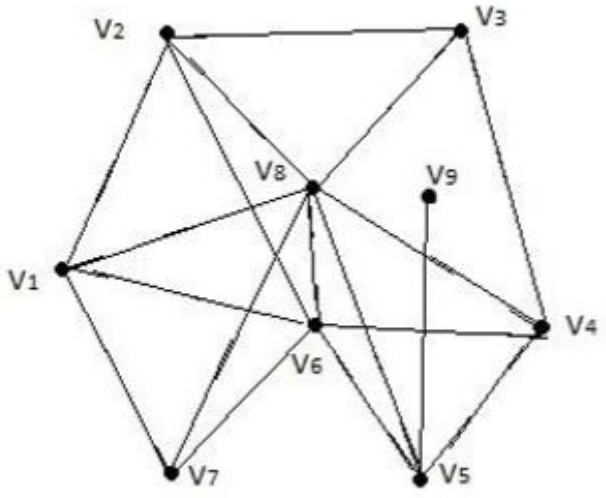


6) $G1 \times G2$



Завдання №2

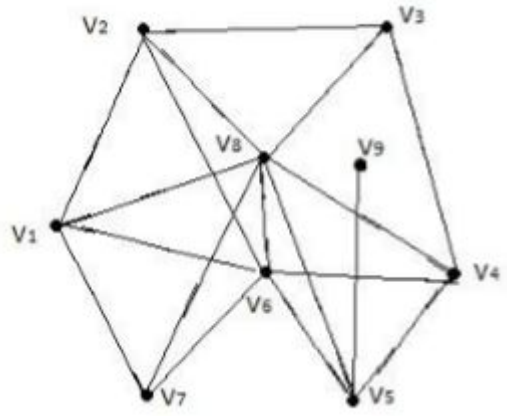
Скласти таблицю суміжності для орграфа.



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

Завдання №3

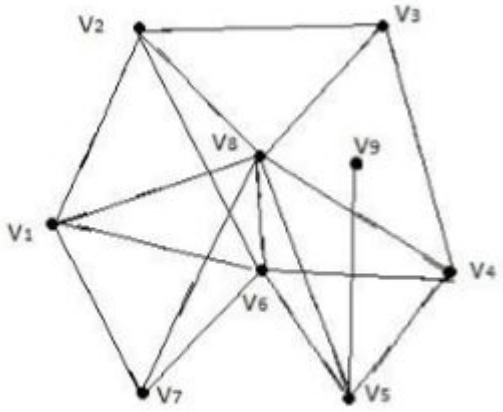
Для графа з другого завдання знайти діаметр.



V1->V6->V5->V9
Діаметр графа дорівнює 3.

Завдання №4

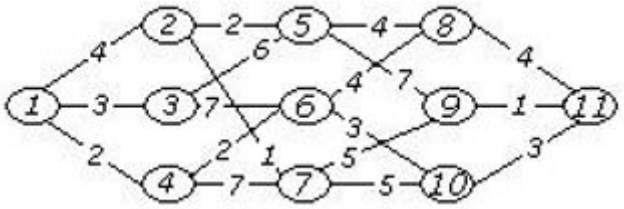
Для графа з другого завдання виконати обхід дерева вшир



Вершина	BFS	Черга
V1	1	1
V2	2	12
V8	3	128
V6	4	1286
V7	5	12867
-	-	2867
V3	6	28673
-	-	8673
V4	7	86734
V5	8	867345
-	-	67345
-	-	7345
-	-	345
-	-	45
-	-	5
V9	9	59
-	-	9
-	-	-

Завдання №5

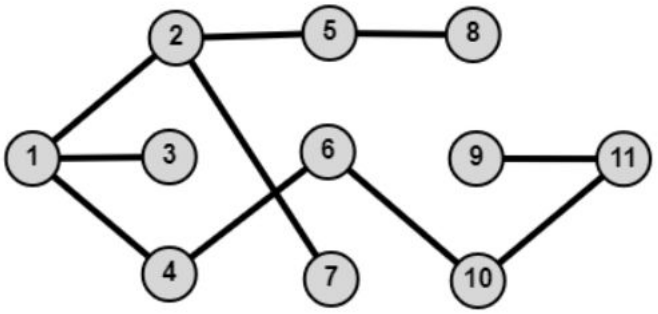
Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



Прима:

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

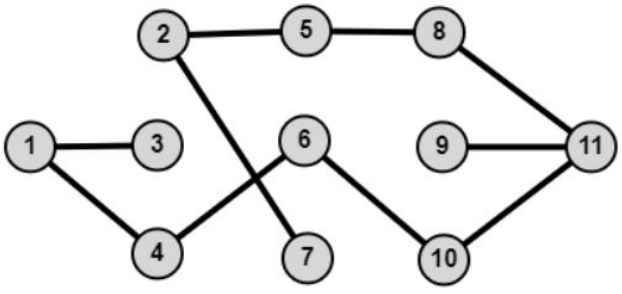
E={(11, 9), (11, 10), (10, 6), (6, 4), (4, 1), (1, 3), (1, 2), (2, 7), (2, 5), (5, 8)}



Краскала:

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

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

	1	2	4	5	63	7	8
1	∞	4	5	1	2	3	5
2	4	∞	1	5	1	5	1
4	5	1	∞	6	4	5	5
5	1	5	6	∞	3	2	2
63	2	1	4	3	∞	2	2
7	3	5	5	2	2	∞	2
8	5	1	5	2	2	2	∞

	163	2	4	5	7	8
163	∞	4	5	1	3	5
2	4	∞	1	5	5	1
4	5	1	∞	6	5	5
5	1	5	6	∞	2	2
7	3	5	5	2	∞	2
8	5	1	5	2	2	∞

	2	4	5163	7	8
2	∞	1	5	5	1
4	1	∞	6	5	5
5361	5	6	∞	2	2
7	5	5	2	∞	2
8	1	5	2	2	∞

	2	4	751163	8
2	∞	1	5	1
4	1	∞	5	5
75361	5	5	∞	2
8	1	5	2	∞

	2	4	8751163
2	∞	1	1
4	1	∞	5
875361	1	5	∞

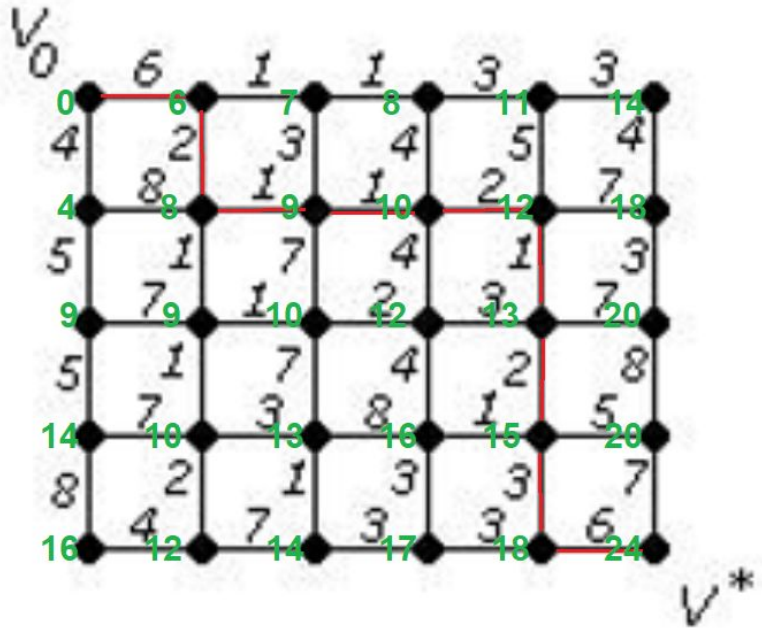
	28751163	4
2875361	∞	1
4	1	∞

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

Найкоротший шлях=1+2+1+2+2+1+1+5=15

Завдання №7

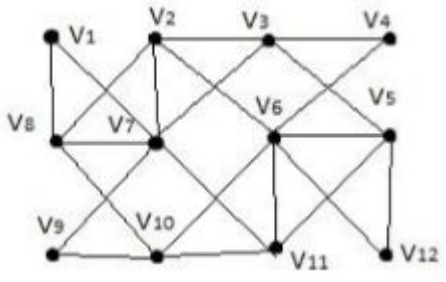
За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V_0 і V^*.



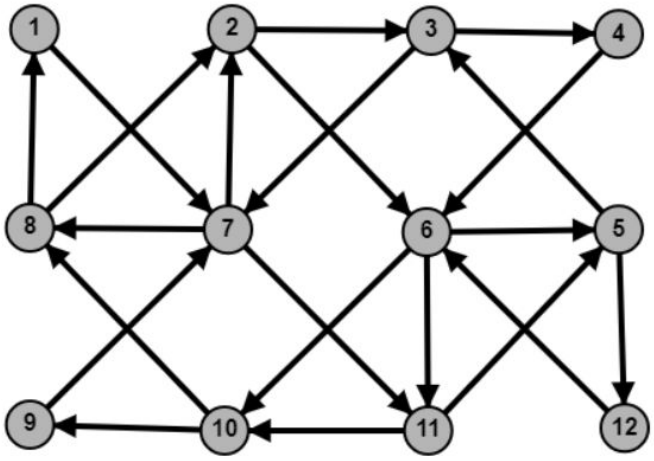
Найкоротший шлях у графі між вершинами дорівнює 24.

Завдання №8

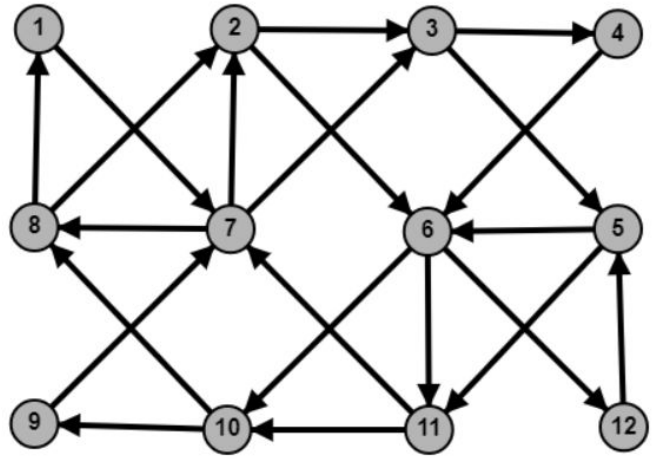
Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.



а) Флері:
8->1->7->2->3->4->6->5->1->2->6->11->5->3->7->11->10->9->7->8->2->6->10->8



б)елементарних циклів:
8->1->7->2->6->10->8->2->3->4->6->12->5->11->10->9->7->3->5->6->11->7->8



Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

$(x \vee \bar{z})(\bar{y} \vee z)$

(x∧¬y)∨(x∧z)∨(¬z∧¬y)∨(¬z∧z)=
=(x∧¬y)∨(x∧z)∨(¬z∧¬y)∨F=
=(x∧¬y)∨(x∧z)∨(¬z∧¬y)=
=((x∧¬y)∧(z∧¬z))∨((x∧z)∧(y∧¬y))∨((¬z∧¬y)∧(x∧¬x))=
=(x∧¬y∧z)∨(x∧¬y∧¬z)∨(x∧y∧z)∨(x∧¬y∧z)∨(x∧¬y∧¬z)∨(¬x∧¬y∧¬z)=
=(x∧¬y∧z)∨(x∧¬y∧¬z)∨(x∧y∧z)∨(¬x∧¬y∧¬z)=
=(x∧z)∨(¬z∧¬y)

Програмна реалізація:

4)Вшир:

```
#include <iostream>
#include <queue>
using namespace std;
int main()
{
  queue<int> Queue;
  int mas[9][9]{ {0, 1, 0, 0, 0, 1, 1, 1, 0},
               { 1,0,1,0,0,1,0,1,0 },
               { 0,1,0,1,0,0,0,1,0 },
               { 0,0,1,0,1,1,0,1,0 },
               { 0,0,0,1,0,1,0,1,1 },
               { 1,1,0,1,1,0,1,1,0 },
               { 1,0,0,0,0,1,0,1,0 },
               { 1,1,1,1,1,1,1,0,0 },
               { 0,0,0,0,1,0,0,0,0 }};
  int nodes[9];
  for (int i = 0; i < 9; i++)
     nodes[i] = 0;
  Queue.push(0);
  while (!Queue.empty())
  {
     int node = Queue.front();
     Queue.pop();
     nodes[node] = 2;
     for (int j = 0; j < 9; j++)
     {
        if (mas[node][j] == 1 && nodes[j] == 0)
        {
           Queue.push(j);
           nodes[j] = 1;
        }
     }
     cout << node + 1 << endl;
  }
  cin.get();
  return 0;
}
```

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

5) Прима:

```
#include <iostream>
using namespace std;
void main()
{
	setlocale(LC_ALL , "ukr");
	int a, b, i, j, min, ne = 1, mincost = 0;
	int visited[15] = { 0 };
	int path[15] = { 0 };
	int num = 0;
	int n = 11;
	int arr[12][12] = {  {0,0,0,0,0,0,0,0,0,0,0,0},
				{0,0,4,3,2,0,0,0,0,0,0,0},
					{0,4,0,0,0,2,0,1,0,0,0,0},
				   {0,3,0,0,0,6,7,0,0,0,0,0},
				   {0,2,0,0,0,0,2,7,0,0,0,0},
				   {0,0,2,6,0,0,0,0,4,7,0,0},
			     {0,0,0,7,2,0,0,0,4,0,3,0},
			     {0,0,1,0,7,0,0,0,0,5,5,0},
			   {0,0,0,0,0,4,4,0,0,0,0,4},
			   {0,0,0,0,0,7,0,5,0,0,0,1},
			     {0,0,0,0,0,0,3,5,0,0,0,3},
			     {0,0,0,0,0,0,0,0,4,1,3,0}
			    };
	int arr2[12][12];
	for (i = 1; i < 12; i++)
		for (j = 0; j < 12; j++) {
			arr2[i][j] = arr[i][j];
			if (arr2[i][j] == 0)
				arr2[i][j] = 999;

		}
	visited[11] = 1;
	cout << "\n\tПошук мінімального остового дерева за методом Прима: ";
	while (ne < n)
	{
		min = 999;
		for (i = 1; i <= n; i++) {
			for (j = 1; j <= n; j++) {
				if (arr2[i][j] < min) {
					if (visited[i] != 0)
					{
						min = arr2[i][j];
						a = i;
						b = j;
					}
				}
			}
```

```
                }
                if (visited[a] == 0 || visited[b] == 0)
                {
                        path[num] = b;

                        num++;
                        ne++;
                        mincost += min;
                        visited[b] = 1;
                        cout << "\n\t( " << a << ",  " << b << ")";
                }
                arr2[a][b] = arr2[b][a] = 999;
        }
        cout << "\n\n";
        cout << "\tV={";
        cout << 11 << ", ";
        for (int i = 0; i < n - 1; i++)
        {
                cout << path[i];
                if (i < n - 2) cout << ", ";
        }
        cout << "}";
        cout << "\n\tВага остового дерева = " << mincost;
        cout << endl << endl;
}
```

```
Пошук мінімального остового дерева за методом Прима:
( 11,  9)
( 11,  10)
( 10,  6)
( 6,  4)
( 4,  1)
( 1,  3)
( 1,  2)
( 2,  7)
( 2,  5)
( 5,  8)

V={11, 9, 10, 6, 4, 1, 3, 2, 7, 5, 8}
Вага остового дерева = 25
```

б) Краскала:

```
#include <iostream>
using namespace std;
struct Rib
{
        int v1, v2, weight;

}Graph[100];
```

```
struct sort_rib {
	int v1;
	int v2;
	int weight;
}sort;
void Fill_Struct(int number_of_ribs) {
	for (int i = 0; i < number_of_ribs; i++) {
		cout << "Firts point: ";
		cin >> Graph[i].v1;
		cout << "Second point: ";
		cin >> Graph[i].v2;

		int sort;
		if (Graph[i].v1 > Graph[i].v2) {
			sort = Graph[i].v1;
			Graph[i].v1 = Graph[i].v2;
			Graph[i].v2 = sort;
		}
		cout << "The rib [" << Graph[i].v1 << ";" << Graph[i].v2 << "] = ";
		cin >> Graph[i].weight;
		cout << endl;
	}
}
void Sort_Sructure(int number_of_ribs) {
	for (int s = 1; s < number_of_ribs; s++) {
		for (int i = 0; i < number_of_ribs - s; i++) {
			if (Graph[i].weight > Graph[i + 1].weight) {
				sort.v1 = Graph[i].v1;
				sort.v2 = Graph[i].v2;
				sort.weight = Graph[i].weight;

				Graph[i].v1 = Graph[i + 1].v1;
				Graph[i].v2 = Graph[i + 1].v2;
				Graph[i].weight = Graph[i + 1].weight;

				Graph[i + 1].v1 = sort.v1;
				Graph[i + 1].v2 = sort.v2;
				Graph[i + 1].weight = sort.weight;
			}
		}
	}
}
void Show_Struct(int number_of_ribs) {
	for (int i = 0; i < number_of_ribs; i++) {
		cout << "The rib [" << Graph[i].v1 << ";" << Graph[i].v2 << "] = " << Graph[i].weight <<
endl;
	}
}
void Algo_Kraskala(int number_of_ribs, int amount_of_points)
{
	int weighttree = 0;
```

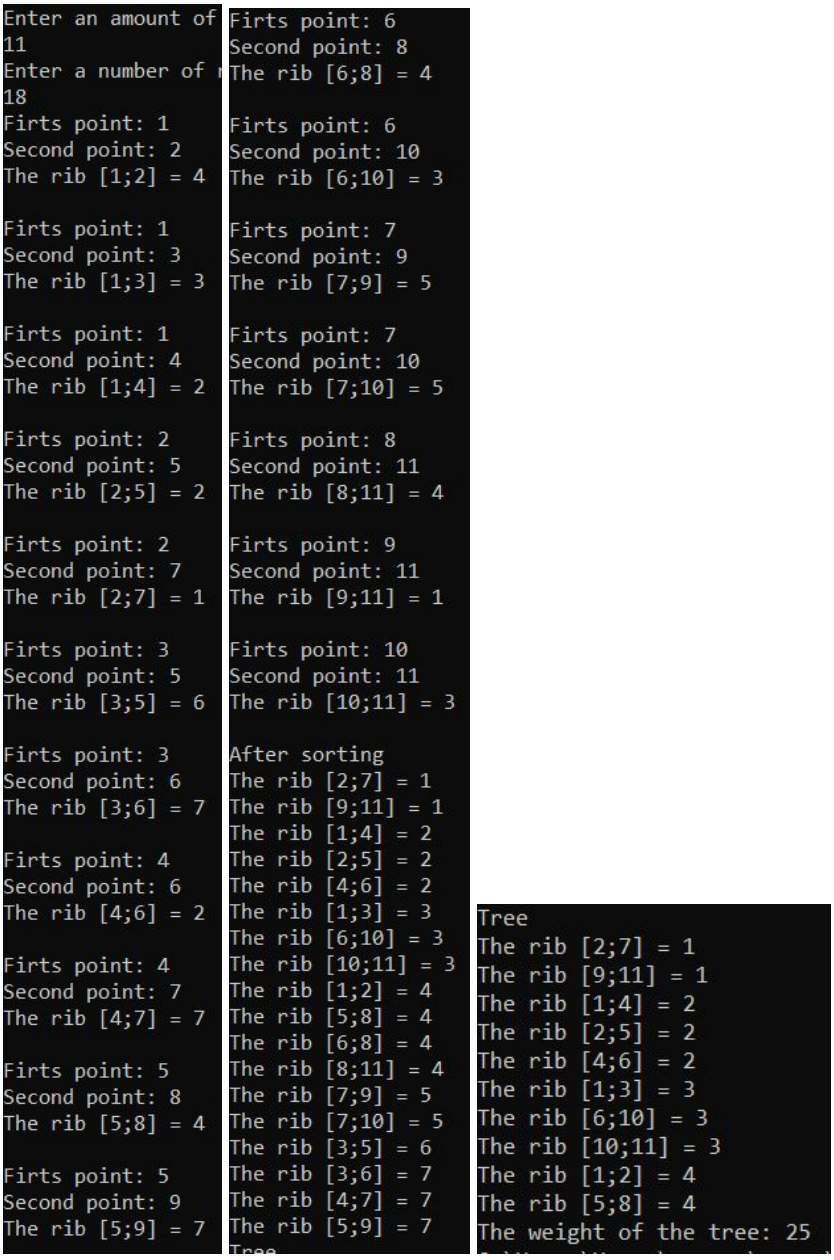
```
	int* parent = new int[amount_of_points];
	int v1, v2, weight;
	int to_change, changed;
	for (int i = 0; i < amount_of_points; i++)
	{
		parent[i] = i;
	}
	for (int i = 0; i < number_of_ribs; i++)
	{
		v1 = Graph[i].v1;
		v2 = Graph[i].v2;
		weight = Graph[i].weight;
		if (parent[v2] != parent[v1])
		{
			cout << "The rib [" << Graph[i].v1 << ";" << Graph[i].v2 << "] = " <<
Graph[i].weight << endl;

			weighttree += weight;

			to_change = parent[v1];

			changed = parent[v2];

			for (int j = 0; j < amount_of_points; j++)
			{
				if (parent[j] == changed)
				{
					parent[j] = to_change;
				}
			}
		}
	}
	delete[] parent;
	cout << "The weight of the tree: " << weighttree;
}
int main() {
	cout << "Enter an amount of points" << endl;
	int q;
	cin >> q;
	int amount_of_points = q + 1;
	cout << "Enter a number of ribs" << endl;
	int number_of_ribs;
	cin >> number_of_ribs;
	Fill_Struct(number_of_ribs);
	Sort_Sructure(number_of_ribs);
	cout << "After sorting" << endl;
	Show_Struct(number_of_ribs);
	cout << "Tree" << endl;
	Algo_Kraskala(number_of_ribs, amount_of_points);
}
```



6)Комівояжера:

```
#include <iostream>
using namespace std;
const int inf=1E9,NMAX=16;
int n,i,j,k,m,temp,ans,d[NMAX][NMAX],t[1<<NMAX][NMAX];
bool get(int nmb,int x)
{ return (x&(1<<nmb))!=0; }

int main()
{
  cin >>n;
  for (i=0;i<n;++i)
```

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

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

15
```

7)Дейкстри:

```
#include<iostream>
using namespace std;
#define INF 999
int V, starting_point;
int temp;
int incidental_matrix[30][30] = {
        {0,6,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {6,0,1,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {0,1,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {0,0,1,0,3,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {0,0,0,3,0,3,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {0,0,0,0,3,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {4,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {0,2,0,0,0,0,8,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
                {0,0,3,0,0,0,0,1,0,1,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0},
```

```
		{0,0,0,4,0,0,0,0,1,0,2,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
		{0,0,0,0,5,0,0,0,0,2,0,7,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
		{0,0,0,0,0,4,0,0,0,0,7,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,7,0,0,0,0,5,0,0,0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0,0,1,0,0,0,0,0,7,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,1,0,2,0,0,0,0,0,7,0,0,0,0,0,0},
		{0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,2,0,3,0,0,0,0,0,4,0,0,0,0,0},
		{0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,3,0,7,0,0,0,0,0,2,0,0,0,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,7,0,0,0,0,0,0,0,8,0,0,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,7,0,0,0,0,8,0,0,0,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,7,0,3,0,0,0,0,0,2,0,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,3,0,8,0,0,0,0,0,1,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,8,0,1,0,0,0,0,3,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,5,0,0,0,0,3},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0,7},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,0,0,4,0,0,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,4,0,7,0,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,7,0,3,0},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3,0,3},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3,0,6},
		{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,6,0}
};

int shortest_dist[30];
bool visited[30] = { 0 };
int parent[30];
int getNearest() {
	int minValue = 999, minNode = 0;
	for (int i = 0; i < V; i++) {
		if (!visited[i] && shortest_dist[i] < minValue) {
			minValue = shortest_dist[i];
			minNode = i;
		}
	}
	return minNode;
}
int main(void) {
	V = 30;
	for (int i = 0; i < V; i++) {
		for (int j = 0; j < V; j++) {
			if (incidental_matrix[i][j] == 0 && i != j) {
				incidental_matrix[i][j] = 999;
			}
		}
	}
	starting_point = 0;
	for (int i = 0; i < V; i++) {
		parent[i] = i;
		shortest_dist[i] = INF;
	}
	shortest_dist[starting_point] = 0;
```

```
	for (int i = 0; i < V; i++) {
		int nearest = getNearest();
		visited[nearest] = true;
		for (int adj = 0; adj < V; adj++) {
			if (incidental_matrix[nearest][adj] != INF && shortest_dist[adj] >
shortest_dist[nearest] + incidental_matrix[nearest][adj]) {
				shortest_dist[adj] = shortest_dist[nearest] +
incidental_matrix[nearest][adj];
				parent[adj] = nearest;
			}
		}
	}
	cout << "Cost : \t\t\tPath" << endl;
	cout << shortest_dist[V - 1] << "\t\t\t" << " ";
	cout << V << " ";
	temp = parent[V - 1];
	while (temp != starting_point) {
		if (temp == 1) {
			cout << " <- " << temp + 1 << " <- 1";
		}
		else {
			cout << " <- " << temp + 1 << " ";
		}
		temp = parent[temp];
	}
	cout << endl;
	return 0;
}
```

```
Cost :                  Path
24                       30  <- 29  <- 23  <- 17  <- 11  <- 10  <- 9  <- 8  <- 2 <- 1
```

8)a) Флері:

```
#include<iostream>
#include<vector>
#define NODE 132
using namespace std;
int graph[NODE][NODE] = {
		{0,0,0,0,0,0,1,1,0,0,0,0},
		{0,0,1,0,0,1,1,1,0,0,0,0},
		{0,1,0,1,1,0,1,0,0,0,0,0},
		{0,0,1,0,0,1,0,0,0,0,0,0},
		{0,0,1,0,0,1,0,0,0,0,1,1},
		{0,1,0,1,1,0,0,0,0,1,1,1},
		{1,1,1,0,0,0,0,1,1,0,1,0},
		{1,1,0,0,0,0,1,0,0,1,0,0},
		{0,0,0,0,0,0,1,0,0,1,0,0},
		{0,0,0,0,0,1,0,1,1,0,1,0},
		{0,0,0,0,1,1,1,0,0,1,0,0},
```

```
                {0,0,0,0,1,1,0,0,0,0,0,0}
};
int tempGraph[NODE][NODE];
bool findStartVert() {
        for (int i = 1; i < NODE; i++) {
                int deg = 0;
                for (int j = 0; j < NODE; j++) {
                        if (tempGraph[i][j])
                                deg++;
                }
                if (deg % 2 != 0)
                        return false;
        }
        return true;
}
bool isBridge(int u, int v) {
        int deg = 0;
        for (int i = 0; i < NODE; i++)
                if (tempGraph[v][i])
                        deg++;
        if (deg > 1) {
                return false;
        }
        return true;
}
int edgeCount() {
        int count = 0;
        for (int i = 0; i < NODE; i++)
                for (int j = i; j < NODE; j++)
                        if (tempGraph[i][j])
                                count++;
        return count;
}
void fleuryAlgorithm(int start) {
        static int edge = edgeCount();
        for (int v = 0; v < NODE; v++) {
                if (tempGraph[start][v]) {
                        if (edge <= 1 || !isBridge(start, v)) {
                                cout << start + 1 << "--" << v + 1 << " ";
                                tempGraph[start][v] = tempGraph[v][start] = 0;
                                edge--;
                                fleuryAlgorithm(v);
                        }
                }
        }
}
int main() {
        for (int i = 0; i < NODE; i++)
                for (int j = 0; j < NODE; j++)

                        tempGraph[i][j] = graph[i][j];
```

```
        if (findStartVert) {
                cout << "Euler Path Or Circuit: ";
                fleuryAlgorithm(7);
        }
        else
                cout<<"Euler Path Or Circuit is imposible"
}
```

```
Euler Path Or Circuit: 8--1 1--7 7--2 2--3 3--4 4--6 6--2 2--8 8--7
7--3 3--5 5--6 6--10 10--9 9--7 7--11 11--5 5--12 12--6 6--11 11--10 10--8
```