МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Варіант 19

Виконав: студент групи
КН-113

ПІ студента: Сидорук Михайло
Викладач: Мельникова Н.І.

Львів 2019

Завдання № 1

Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

2) об’єднання графів,

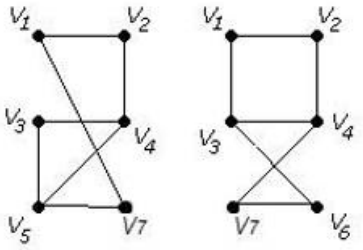
3) кільцеву сумму G1 та G2 (G1+G2),

4) розмножити вершину у другому графі,

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

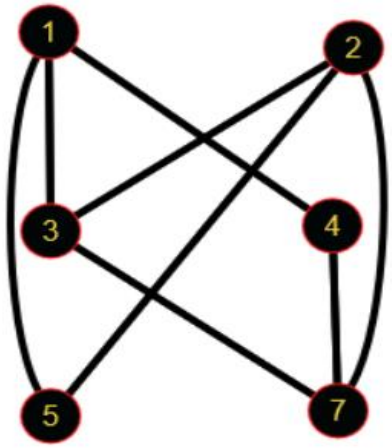
6) добуток графів.

19)

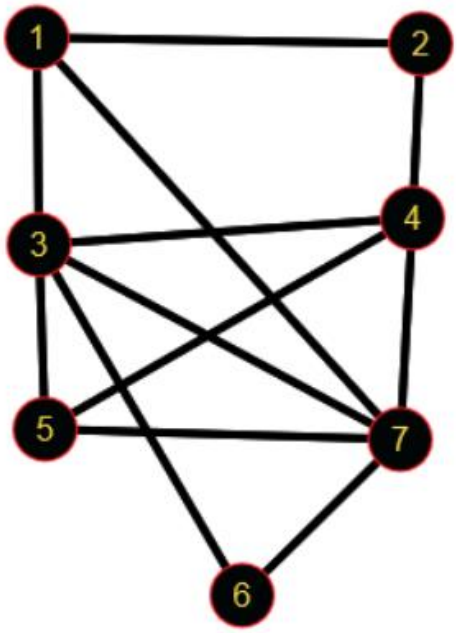


Розв’язання:

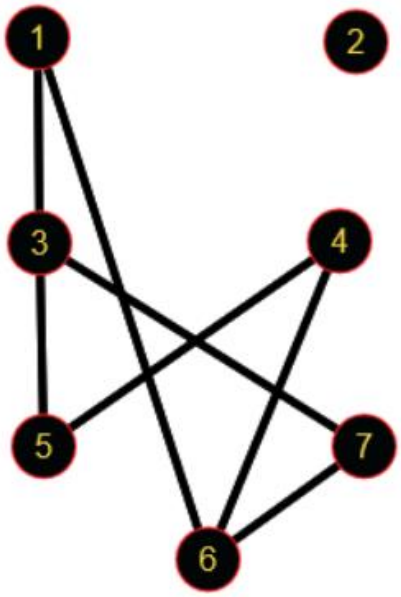
1.



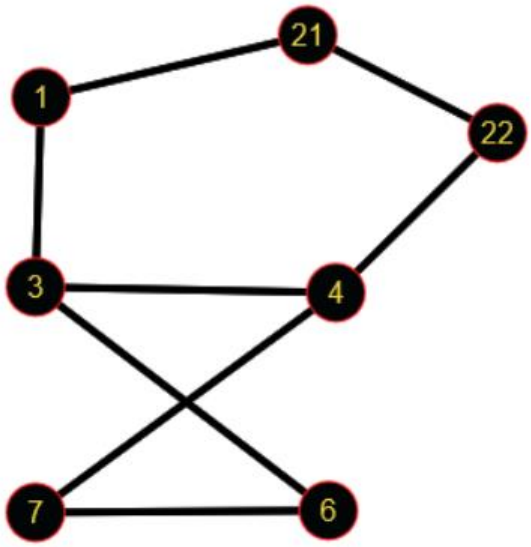
2.



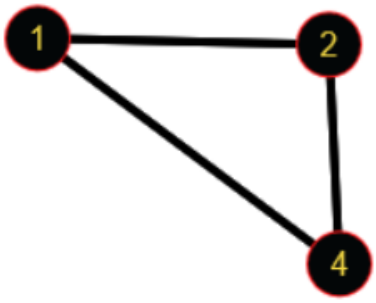
3.



4.



5.

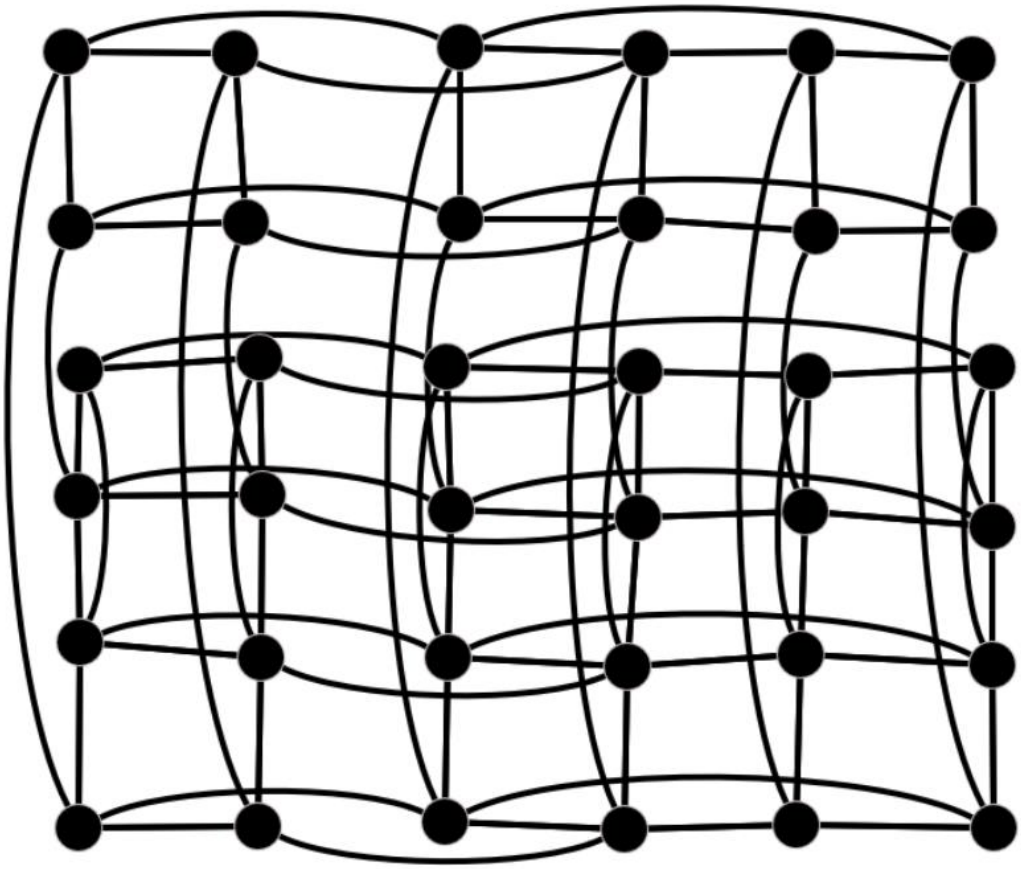


6.

G1:

G2:

Відповідно, добуток графів буде виглядати наступним чином:

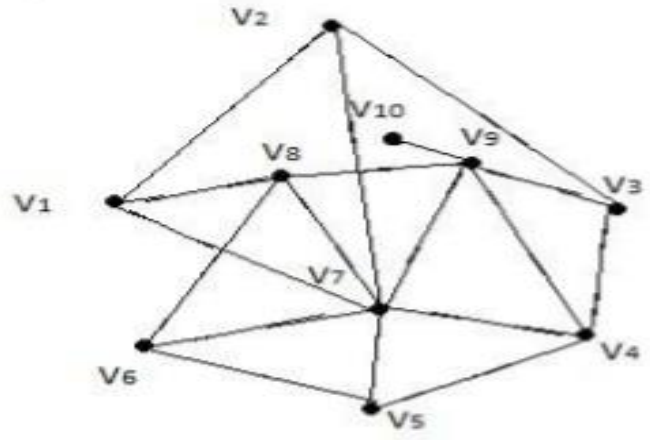


По горизонталі G2, по вертикалі G1.

Завдання № 2

Скласти таблицю суміжності для орграфа.

19)



Розв’язання:

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

Результат виконання:

Завдання № 3

Для графа з другого завдання знайти діаметр.

Діаметр даного графа = 3, а саме відстань між вершинами 10 і 6.

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
1	1	1
2	2	1 2
3	3	1 2 3
9	4	1 2 3 9
10	5	1 2 3 9 10
-	-	1 2 3 9
8	6	1 2 3 9 8
7	7	1 2 3 9 8 7
6	8	1 2 3 9 8 7 6
5	9	1 2 3 9 8 7 6 5
4	10	1 2 3 9 8 7 6 5 4
-	-	1 2 3 9 8 7 6 5
-	-	1 2 3 9 8 7 6
-	-	1 2 3 9 8 7
-	-	1 2 3 9 8
-	-	1 2 3 9
-	-	1 2 3
-	-	1 2
-	-	1
-	-	∅

Програмна реалізація:

```
#include "pch.h"
#include <iostream>
#include <string>
#include <sstream>

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()
{
		setlocale(LC_ALL, "Ukrainian");
		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;
}
```

Результат роботи програми:

Введіть кількість ребер у графі: 18

Введіть кількість вершин у графі: 10

Введіть першу вершину, інцидентну ребру №1: 1

Введіть другу вершину, інцидентну ребру №1: 2

Введіть першу вершину, інцидентну ребру №2: 1

Введіть другу вершину, інцидентну ребру №2: 8

Введіть першу вершину, інцидентну ребру №3: 1
Введіть другу вершину, інцидентну ребру №3: 7

Введіть першу вершину, інцидентну ребру №4: 2
Введіть другу вершину, інцидентну ребру №4: 7

Введіть першу вершину, інцидентну ребру №5: 2
Введіть другу вершину, інцидентну ребру №5: 3

Введіть першу вершину, інцидентну ребру №6: 3
Введіть другу вершину, інцидентну ребру №6: 4

Введіть першу вершину, інцидентну ребру №7: 3
Введіть другу вершину, інцидентну ребру №7: 9

Введіть першу вершину, інцидентну ребру №8: 4
Введіть другу вершину, інцидентну ребру №8: 9

Введіть першу вершину, інцидентну ребру №9: 4
Введіть другу вершину, інцидентну ребру №9: 7

Введіть першу вершину, інцидентну ребру №10: 4
Введіть другу вершину, інцидентну ребру №10: 5

Введіть першу вершину, інцидентну ребру №11: 5
Введіть другу вершину, інцидентну ребру №11: 7

Введіть першу вершину, інцидентну ребру №12: 5
Введіть другу вершину, інцидентну ребру №12: 6

Введіть першу вершину, інцидентну ребру №13: 6
Введіть другу вершину, інцидентну ребру №13: 7

Введіть першу вершину, інцидентну ребру №14: 6
Введіть другу вершину, інцидентну ребру №14: 8

Введіть першу вершину, інцидентну ребру №15: 8
Введіть другу вершину, інцидентну ребру №15: 9

Введіть першу вершину, інцидентну ребру №16: 9
Введіть другу вершину, інцидентну ребру №16: 10

Введіть першу вершину, інцидентну ребру №17: 7
Введіть другу вершину, інцидентну ребру №17: 8

Введіть першу вершину, інцидентну ребру №18: 7
Введіть другу вершину, інцидентну ребру №18: 9

З якої вершини почати обхід? 1
Якщо ви хочете зробити:
обхід вглиб натисніть 1,
обхід вшир - натисніть 2:
1

1

1 2

1 2 7

1 2 7 4

1 2 7 4 3

1 2 7 4 3 9

1 2 7 4 3 9 8

1 2 7 4 3 9 8 6

1 2 7 4 3 9 8 6 5

1 2 7 4 3 9 8 6

1 2 7 4 3 9 8

1 2 7 4 3 9

1 2 7 4 3 9 10

1 2 7 4 3 9

1 2 7 4 3

1 2 7 4

1 2 7

1 2

1

Стек пустий

C:\Users\Misha_Sydoruk\source\repos\Diskretka_rozraha\Debug\Diskretka_rozraha.exe (процесс 19756) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

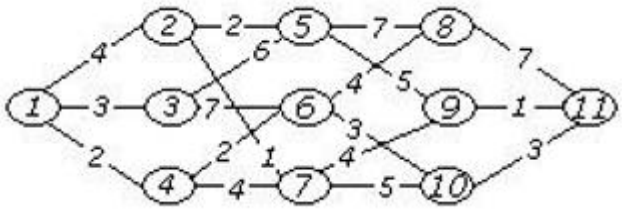
Чтобы закрыть это окно, нажмите любую клавишу…

Як бачимо, програма виконала обхід іншим способом, але обидва способи є правильними.

Завдання № 5

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

19)



Метод Краскала:

Спочатку сортуємо ребра по зростанню. Отримуємо такий результат:

(2, 7) - 1;

(9, 11) - 1;

(1, 4) - 2;

(2, 5) - 2;

(4, 6) - 2;

(1, 3) - 3;

(6, 10) - 3;

(10, 11) - 3;

(1, 2) - 4;

(4, 7) - 4;

(6, 8) - 4;

(7, 9) - 4;

(5, 9) - 5;

(7, 10) - 5;

(3, 5) - 6;

(3, 6) -7;

(5, 8) - 7;

(8, 11) - 7;

Тоді включаємо ребра в такій послідовності і слідкуємо щоб не утворився цикл:

9 11

2 7

1 4

2 5

4 6

1 3

10 11

6 10

1 2

6 8

Вага такого дерева = 25.

Програмна реалізація:

```
#include "pch.h"
#include <iostream>
#include <stdio.h>
using namespace std;
struct rebro {

	int leng;
	int v1;
	int v2;
	bool in = false;
};
struct mas {

	int arr[100];
	int c = 0;
};

int in(rebro *reb, int n) {
	setlocale(LC_ALL, "Ukrainian");
	for (int i = 0; i < n; i++)
	{
		cout << "Введіть довжину " << i + 1 << " ребра: ";
```

```
		cin >> reb[i].leng;
		cout << "Введіть першу суміжну вершину з " << i + 1 << " ребром: ";
		cin >> reb[i].v1;
		cout << "Введіть другу суміжну вершину з " << i + 1 << " ребром: ";
		cin >> reb[i].v2;
		cout << endl;
	}
	return 0;
}
int main() {

	setlocale(LC_ALL, "Ukrainian");
	int n = 0, x = 100, y = 100;

	cout << "Введіть кількість ребер у графі: ";
	cin >> n;
	int z;
	cout << "Введіть кількість вершин у графі: ";
	cin >> z;
	cout << endl;

	rebro *reb = new rebro[n];
	mas inn[15];

	for (int i = 0; i < 15; i++)
	{
		for (int j = 0; j < z; j++)
		{
			inn[i].arr[j] = 0;
		}
	}
	in(reb, n);

	for (int i = 0; i < n - 1; i++)
	{
		for (int j = 0; j < n - 1; j++)
		{
			if (reb[j].leng > reb[j + 1].leng) { swap(reb[j].leng, reb[j + 1].leng);
swap(reb[j].v1, reb[j + 1].v1); swap(reb[j].v2, reb[j + 1].v2); }
		}
	}

	for (int i = 0; i < n; i++)
		cout << reb[i].leng << " " << reb[i].v1 << " " << reb[i].v2 << endl;


	int c = -1;
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < 15; j++)
		{
			for (int k = 0; k < z; k++)
			{
				if (reb[i].v1 == inn[j].arr[k]) { x = j; goto point0;; }
			}
		}
	point0:;

		for (int j = 0; j < 15; j++)
		{
			for (int k = 0; k <z; k++)
```

```
                        {
                                if (reb[i].v2 == inn[j].arr[k]) { y = j; goto point1; }
                        }
                }
        point1:;
                if (x != y && x == 100) { inn[y].arr[inn[y].c] = reb[i].v1; inn[y].c++; }

                if (x != y && y == 100) { inn[x].arr[inn[x].c] = reb[i].v2; inn[x].c++; }

                if (x != y && x != 100 && y != 100) {
                        if (x < y) {
                                for (int l = 0; l < inn[y].c; l++)
                                {
                                        inn[x].arr[inn[x].c+l] = inn[y].arr[l];
                                        inn[y].arr[l] = 0;
                                }
                                inn[x].c += inn[y].c;
                                inn[y].c = 0;
                        }

                        if (y < x) {
                                for (int l = 0; l < inn[x].c; l++)
                                {
                                        inn[y].arr[inn[y].c+l] = inn[x].arr[l];
                                        inn[x].arr[l] = 0;
                                }
                                inn[y].c += inn[x].c;
                                inn[x].c = 0;
                        }
                }
                if (x == 100 && y == 100) { c++;  inn[c].arr[inn[c].c] = reb[i].v1;
inn[c].arr[inn[c].c + 1] = reb[i].v2; inn[c].c += 2; }

                reb[i].in = true;

                if (x == y && x != 100) { reb[i].in = false; }

                x = 100; y = 100;

                for (int j = 0; j < 15; j++)
                {
                        for (int k = 0; k < 11; k++)
                                cout << inn[j].arr[k] << " ";
                        cout << endl;
                }
                cout << endl;
        }

        cout << "Щоб побудувати остове дерево мінімальної ваги, ми повинні включити в нього
такі ребра: " << endl;

        int s = 0;;

        for (int i = 0; i < n; i++)
        {
                if (reb[i].in == true) { cout << "Ребро" << ", що сполучає вершини " <<
reb[i].v1 << " " << reb[i].v2 << endl; s += reb[i].leng; }
        }
        cout << "Остове дерево мінімальної ваги для даного графа: " << s;
        return 0;
        }
```

Результат роботи програми:

Введіть кількість ребер у графі: 18

Введіть кількість вершин у графі: 11

Введіть довжину 1 ребра: 4

Введіть першу суміжну вершину з 1 ребром: 1

Введіть другу суміжну вершину з 1 ребром: 2

Введіть довжину 2 ребра: 3

Введіть першу суміжну вершину з 2 ребром: 1

Введіть другу суміжну вершину з 2 ребром: 3

Введіть довжину 3 ребра: 2

Введіть першу суміжну вершину з 3 ребром: 1

Введіть другу суміжну вершину з 3 ребром: 4

Введіть довжину 4 ребра: 2

Введіть першу суміжну вершину з 4 ребром: 2

Введіть другу суміжну вершину з 4 ребром: 5

Введіть довжину 5 ребра: 7

Введіть першу суміжну вершину з 5 ребром: 3

Введіть другу суміжну вершину з 5 ребром: 6

Введіть довжину 6 ребра: 4

Введіть першу суміжну вершину з 6 ребром: 4

Введіть другу суміжну вершину з 6 ребром: 7

Введіть довжину 7 ребра: 7

Введіть першу суміжну вершину з 7 ребром: 5

Введіть другу суміжну вершину з 7 ребром: 8

Введіть довжину 8 ребра: 5

Введіть першу суміжну вершину з 8 ребром: 7

Введіть другу суміжну вершину з 8 ребром: 10

Введіть довжину 9 ребра: 7

Введіть першу суміжну вершину з 9 ребром: 8

Введіть другу суміжну вершину з 9 ребром: 11

Введіть довжину 10 ребра: 1

Введіть першу суміжну вершину з 10 ребром: 9

Введіть другу суміжну вершину з 10 ребром: 11

Введіть довжину 11 ребра: 3

Введіть першу суміжну вершину з 11 ребром: 10

Введіть другу суміжну вершину з 11 ребром: 11

Введіть довжину 12 ребра: 1

Введіть першу суміжну вершину з 12 ребром: 2

Введіть другу суміжну вершину з 12 ребром: 7

Введіть довжину 13 ребра: 5

Введіть першу суміжну вершину з 13 ребром: 5

Введіть другу суміжну вершину з 13 ребром: 9

Введіть довжину 14 ребра: 3
Введіть першу суміжну вершину з 14 ребром: 6
Введіть другу суміжну вершину з 14 ребром: 10

Введіть довжину 15 ребра: 6
Введіть першу суміжну вершину з 15 ребром: 3
Введіть другу суміжну вершину з 15 ребром: 5

Введіть довжину 16 ребра: 4
Введіть першу суміжну вершину з 16 ребром: 6
Введіть другу суміжну вершину з 16 ребром: 8

Введіть довжину 17 ребра: 2
Введіть першу суміжну вершину з 17 ребром: 4
Введіть другу суміжну вершину з 17 ребром: 6

Введіть довжину 18 ребра: 4
Введіть першу суміжну вершину з 18 ребром: 7
Введіть другу суміжну вершину з 18 ребром: 9

1 9 11
1 2 7
2 1 4
2 2 5
2 4 6

3 1 3

3 10 11

3 6 10

4 1 2

4 4 7

4 6 8

4 7 9

5 7 10

5 5 9

6 3 5

7 3 6

7 5 8

7 8 11

9 11 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 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 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 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

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 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

9 11 0 0 0 0 0 0 0 0 0
2 7 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 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 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 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
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 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

9 11 0 0 0 0 0 0 0 0 0
2 7 0 0 0 0 0 0 0 0 0
1 4 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 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 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 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 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 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

9 11 0 0 0 0 0 0 0 0 0

2 7 5 0 0 0 0 0 0 0 0

1 4 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 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 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 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 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

9 11 0 0 0 0 0 0 0 0 0

2 7 5 0 0 0 0 0 0 0 0

1 4 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 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 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 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 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 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

9 11 0 0 0 0 0 0 0 0 0

2 7 5 0 0 0 0 0 0 0 0

1 4 6 3 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 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 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

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 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 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

9 11 10 0 0 0 0 0 0 0 0

2 7 5 0 0 0 0 0 0 0 0

1 4 6 3 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 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 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 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

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 0 0 0 0 0 0

9 11 10 1 4 6 3 0 0 0 0

2 7 5 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 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 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 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 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 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

9 11 10 1 4 6 3 2 7 5 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 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 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 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 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 0 0 0 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 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 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 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 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 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 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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 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 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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 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 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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 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 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 0 0 0

0 0 0 0 0 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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 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 0 0 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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 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 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 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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

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 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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

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 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 0 0 0 0 0 0

9 11 10 1 4 6 3 2 7 5 8

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 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 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 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

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 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 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

Щоб побудувати остове дерево мінімальної ваги, ми повинні включити в нього такі ребра:

Ребро, що сполучає вершини 9 11

Ребро, що сполучає вершини 2 7

Ребро, що сполучає вершини 1 4

Ребро, що сполучає вершини 2 5

Ребро, що сполучає вершини 4 6

Ребро, що сполучає вершини 1 3

Ребро, що сполучає вершини 10 11

Ребро, що сполучає вершини 6 10

Ребро, що сполучає вершини 1 2

Ребро, що сполучає вершини 6 8

Остове дерево мінімальної ваги для даного графа: 25

C:\Users\Misha_Sydoruk\source\repos\diskretkalaba4\Debug\diskretkalaba4.exe (процесс 19220) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Чтобы закрыть это окно, нажмите любую клавишу…

Метод Прима

Вибираємо ребро з найменшою вагою і включаємо його в дерево. В даному випадку це ребро (2, 7).

Шукаємо вершину, яка сполучена з будь якою з тих, що вже є в дереві ребром мінімальної ваги і повторюємо доки всі вершини не будуть

включеними в дерево. Слідкуємо, щоб не утворився цикл. Включаємо ребра в такому порядку:

2 5

1 2

1 4

4 6

1 3

6 10

10 11

9 11

6 8

Вага дерева не змінюється і дорівнює 25.

Програмна реалізація:

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

ifstream fin;
string path = "MyFile.txt";
int main() {
	string str = "";

	setlocale(LC_ALL, "Ukrainian");
	int number = 11;

	int**arr = new int*[number];
	for (int i = 0; i < number; i++)
	{
		arr[i] = new int[number];
	}



	fin.open(path);
	for (int i = 0; i < number; i++)
	{
		for (int j = 0; j < number; j++)
		{
			getline(fin, str);
			arr[i][j] = atoi(str.c_str());
			str = "";
		}
	}
```

```
fin.close();

int s = 0;
int min=100, MIN, var, count=1;
int vershina[11];
vershina[0] = 0;

cout << "Включаємо в граф такі ребра:  " << endl;
do {

	for (int i = 0; i < count; i++) {
		for (int j = 0; j < 11; j++) {
			if (arr[vershina[i]][j] != 0 && arr[vershina[i]][j] < min) {
				min = arr[vershina[i]][j];
				MIN = j;
				var = vershina[i];
			}
		}
	}

	vershina[count] = MIN;

	count++;
	min = 10;

	s += arr[var][MIN];
	for (int i = 0; i < 11; i++) {
		arr[i][MIN] = 0;
	}

	cout <<"Ребро, що сполучає вершини "<<var  + 1 << " і " << MIN + 1 << endl;

} while (count < 11);
cout << "Вага дерева = " << s;
}
```

Вміст текстового файлу:

0

4

3

2

0

0

0

0

0

0

0
0
0
0
0
2
0
1
0
0
0
0
0
0
0
0
6
7
0
0
0
0
0
0
0
0
0
0

0

2

4

0

0

0

0

0

2

6

0

0

0

0

0

7

5

0

0

0

0

0

7

2

0

0

0

4

0

3
0
0
1
0
4
0
0
0
0
4
5
0
0
0
0
0
0
7
4
0
0
0
0
7
0
0
0

0

5

0

4

0

0

0

1

0

0

0

0

0

0

3

5

0

0

0

3

0

0

0

0

0

0

0

0

0

7

1

3

0

Результат роботи програми:

Включаємо в граф такі ребра:

Ребро, що сполучає вершини 1 і 4

Ребро, що сполучає вершини 4 і 6

Ребро, що сполучає вершини 1 і 3

Ребро, що сполучає вершини 6 і 10

Ребро, що сполучає вершини 10 і 11

Ребро, що сполучає вершини 11 і 9

Ребро, що сполучає вершини 1 і 2

Ребро, що сполучає вершини 2 і 7

Ребро, що сполучає вершини 2 і 5

Ребро, що сполучає вершини 6 і 8

Вага дерева = 25

C:\Users\Misha_Sydoruk\source\repos\Prim\Debug\Prim.exe (процесс 15052) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Чтобы закрыть это окно, нажмите любую клавишу…

Завдання № 6

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

19)

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

Розв’язання:

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

Починаємо з першої вершини і йдемо до найближчої, потім продовжуємо.

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

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

	3	5	6	7	8
3	-	6	5	**1**	5
5	6	-	5	1	5
6	5	5	-	2	1
7	1	1	2	-	5
8	5	5	1	5	-

	5	6	7	8
5	-	5	**1**	5
6	5	-	2	1
7	1	2	-	5
8	5	1	5	-

	5	6	8
5	-	**5**	5
6	5	-	1
8	5	1	-

	6	8
6	-	**1**
8	1	-

Маршрут:

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

Довжина такого маршруту буде складати

2+1+6+1+1+5+1+2=19

Але це не найкоротший маршрут, ми мусимо пройти стільки разів, скільки є вершин, кожного разу вибираючи іншу вершину для початку шляху.

Другий пошук:

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

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

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

	1	2	4	6	8
1	∞	2	2	3	2
2	2	∞	1	3	4
4	2	1	∞	6	6
6	3	3	6	∞	1
8	2	4	6	1	∞

	1	4	6	8
1	∞	2	3	2
4	2	∞	6	6
6	3	6	∞	1
8	2	6	1	∞

	1	6	8
1	∞	3	2
6	3	∞	1
8	2	1	∞

	6	8
6	∞	1
8	1	∞

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

Це і є найкоротший маршрут.

Його довжина складає 15.

Програмна реалізація:

```
#include "pch.h"
#include <iostream>
#include <stdio.h>
#include <string>
#include <fstream>

using namespace std;
ifstream fin;
string path = "MyFile.txt";

struct mass
{
	int mas[9];
};

int** input() {
	int count = 8;

	string str;
	str = "";

	fin.open(path);
	int **arr;
	arr = new int*[count];
	for (int i = 0; i < count; i++)
		arr[i] = new int[count];

	for (int i = 0; i < count; i++)
	{
		for (int j = 0; j < count; j++)
			arr[i][j] = 0;
	}
	for (int i = 0; i < count; i++)
	{
		for (int j = i + 1; j < count; j++)
		{
			getline(fin, str);
			arr[i][j] = atoi(str.c_str());
			arr[j][i] = atoi(str.c_str());
		}
	}
```

```
	fin.close();

	return arr;
}
bool comp(int* arr, int count)
{
	int* mas = new int[count];



	for (int i =0; i < count; i++)
	{
		mas[i] = count -i;
	}

	for (int i = 0; i < count; i++)
	{
		if (mas[i] != arr[i])
		{
			return true;
		}
		else
		{
			continue;
		}
	}
	return false;
}
bool povtor(int* mas, int size)
{
	bool k = true;

	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			if (mas[i] == mas[j] && i != j)
			{
				return false;
			}
		}
	}

	return true;
}
int way(int** mat, int* arr)
{
	int count = 0;

	for (int i = 0; i < 7; i++)
	{
		count += mat[arr[i] - 1][arr[i + 1] - 1];
	}

	count += mat[arr[7] - 1][arr[0] - 1];
	return count;
}

int main() {
	int const count = 8;
	int **arr;
```

```
arr = input();
int var = count - 1;
bool k = true;
int *mas = new int[count];

int* minmas = new int[9];
int min = 1000;
int leng = 0;

int m = 0;


for (int i = 0; i < count; i++)
{
	mas[i] = 1;
	minmas[i] = 1;
}

while (comp(mas, count))
{
	while (mas[var] != count)
	{
		mas[var]++;

		if (povtor(mas, count))
		{
			leng = way(arr, mas);

			for (int i = 0; i < count; i++)
			{
				cout << mas[i] << "-> ";
			}
			cout << mas[0] << " (" << leng << ") ";
			cout << endl;

			if (leng < min)
			{
				min = leng;
				m = 1;
			}
			if (leng == min)
			{
				m++;
			}
		}
	}

	while (mas[var] == count)
	{
		mas[var] = 1;
		var--;
	}
	mas[var]++;

	if (povtor(mas, count))
	{
		for (int i = 0; i < count; i++)
		{
			cout << mas[i] << "-> ";
		}
		cout << mas[0] << " (" << leng << ") ";
```

```
                cout << endl;

                leng = way(arr, mas);

                if (leng < min)
                {
                        min = leng;
                        m = 1;
                }
                if (leng == min)
                {
                        m++;
                }

        }
        var = count - 1;
}

for (int i = 0; i < count; i++)
{
        mas[i] = 1;
        minmas[i] = 1;
}
mass *rez = new mass[m];
int iter = 0;

while (comp(mas, count))
{
        while (mas[var] != count)
        {
                mas[var]++;

                if (povtor(mas, count))
                {
                        leng = way(arr, mas);

                        if (leng == min)
                        {
                                for (int i = 0; i < count; i++)
                                {
                                        rez[iter].mas[i] = mas[i];
                                }
                                rez[iter].mas[count] = mas[0];
                                iter++;
                        }
                }
        }
        while (mas[var] == count)
        {
                mas[var] = 1;
                var--;
        }
        mas[var]++;

        if (povtor(mas, count))
        {
                leng = way(arr, mas);

                if (leng == min)
                {
                        for (int i = 0; i < count; i++)
```

```
                    {
                        rez[iter].mas[i] = mas[i];
                    }
                    rez[iter].mas[count] = mas[0];
                    iter++;
                }

        }
        var = count - 1;
}

cout << "Ways: " << endl;

for (int i = 0; i < iter - 1; i++)
{
        for (int j = 0; j <= count; j++)
        {
                if (j != 0)
                {
                        cout << "-> ";
                }
                cout << rez[i].mas[j] << " ";
        }
        cout << endl;
}
cout << "Minimal leng = " << min;

return 0;
}
```

Вміст файлу: 2

2

2

2

3

2

2

5

1

2

3

2

4

6

6

5

1

5

6

6

6

6

5

1

5

2

1

5

Результат роботи програми:

Я не пишу сюди всі можливі шляхи, оскільки їх існує 8! і це займе дуже багато місця, лише найменші.

Ways:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

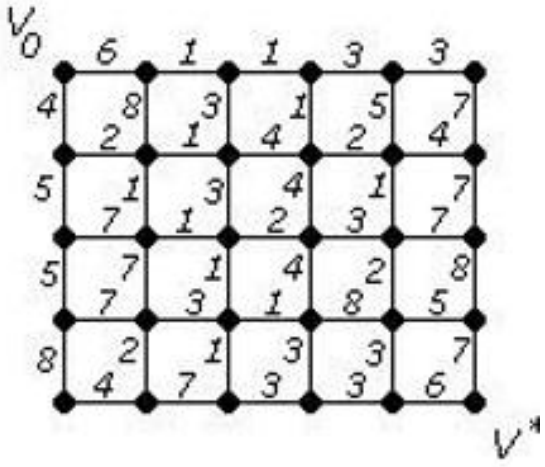
8 -> 1 -> 4 -> 2 -> 5 -> 7 -> 3 -> 6 -> 8

Minimal leng = 15

Завдання № 7

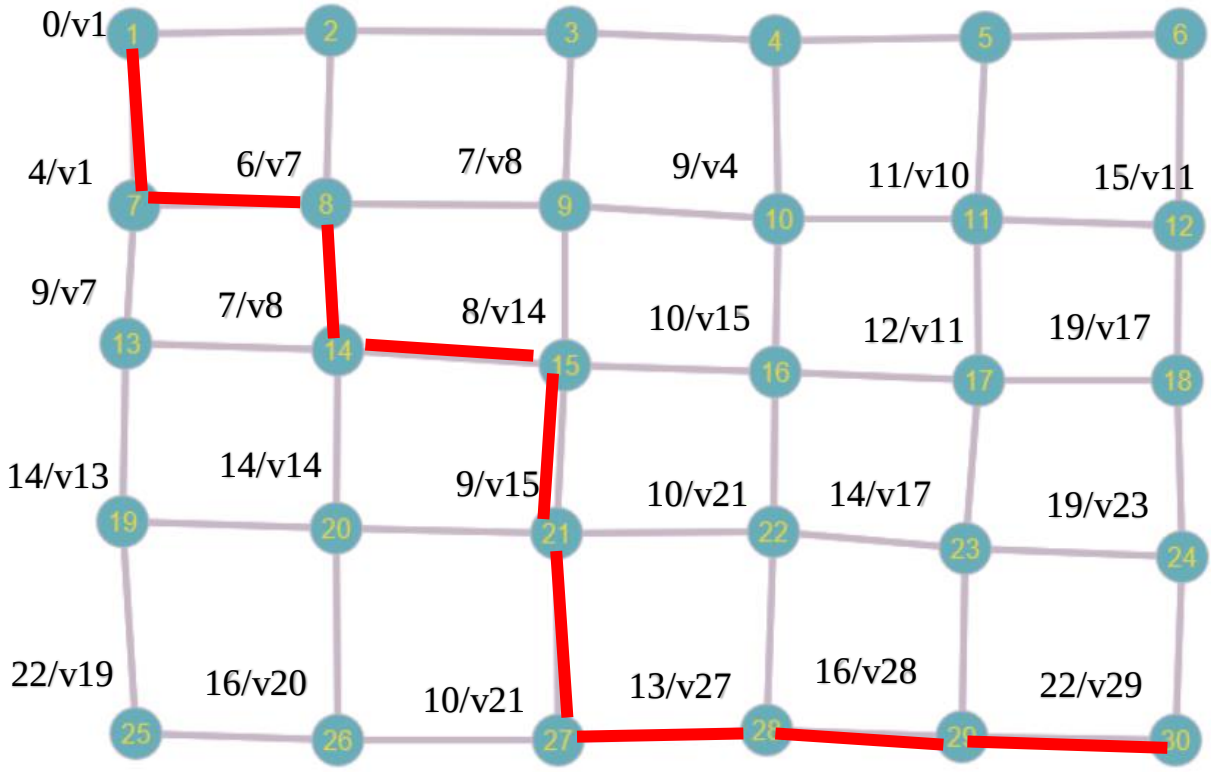
За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V*

19)



Розв’язання:

6/v1 7/v2 8/v3 11/v4 14/v4



Довжина шляху=22.

Програмна реалізація:

```
#include "pch.h"
#include <iostream>
#include <sstream>
#include <fstream>
#include <stdlib.h>
using namespace std;
string path = "MyFile1.txt";
ifstream fin;
ofstream fout;

struct vershina {
	int number;
	bool proid = false;
	bool wiev = false;
	int minleng = 1000;
	string way="";
};
struct rebro {
	int v1;
	int v2;
	int leng;
};
void Add(rebro *reb, int i) {
	setlocale(LC_ALL, "Ukrainian");
	string str;
```

```
        /*cout <<"Введіть вагу " << i + 1 << " ребра: " ;
        cin >> reb[i].leng;
        cout <<"Перша суміжна вершина: ";
        cin >> reb[i].v1;
        cout << "Друга суміжна вершина: ";
        cin >> reb[i].v2;
        cout << endl;*/
        str = "";
        getline(fin, str);
        reb[i].leng = atoi(str.c_str());

        str = "";
        getline(fin, str);
         reb[i].v1=atoi(str.c_str());

        str = "";
        getline(fin, str);
         reb[i].v2=atoi(str.c_str());
}

int main() {
        setlocale(LC_ALL, "Ukrainian");

        int n = 49, m = 0;
        int k = 0;
        int x = 0;
        int min;
        int minleng = 1000;
        stringstream ss[200];
        string str;
        int t = 0;
        int begin, end;
        /*cout << "Введіть кількість ребер у графі:  ";
        cin >> n;
        cout << endl;*/
        rebro *reb = new rebro[n];
        vershina v[30];
        cout << "З якої вершини почати? (не більше 30) ";
        cin >> begin;
        cout << "До якої вершини знайти шлях?(не більше 30) ";
        cin >> end;

        for (int i = 0; i < 30; i++)
        {
                v[i].number = i+1;
        }

        fin.open(path);
        for (int i = 0; i < n; i++)
        {
                Add(reb, i);
        }
        fin.close();
        v[begin-1].wiev = true;
        v[begin-1].minleng = 0;
        ss[t] << begin;
        ss[t] >> str;
        v[begin-1].way += str;
        t++;
        str = "";
```

```
while (v[end-1].proid == false )
{
	for(int i=0; i<30; i++)
	{
		if(v[i].wiev==true && v[i].proid==false)
		{
			//cout << v[i].number<<" "<<" вершина з шляхом не 1000"<<" ";
			m++;
		}
	}
	//cout << endl;
	//кількість вершин до яких вже визначений шлях.
	//cout << m<<" кількість з шляхом не 1000"<<endl;

	int *mas_v = new int [m]; //виділення пам'яті під них.

	for (int i = 0; i < 30; i++)
	{
		if (v[i].wiev==true && v[i].proid==false)
		{
			mas_v[k] = i;
			//cout << mas_v[k] << " - index of wievs ";
			k++;
		}
	}
	//cout << endl;
	//заповнення масиву вершин, що можуть розглядатися.
	k = 0;

	for (int i = 0; i < 30; i++)
	{
		if (v[i].wiev == true && v[i].minleng<minleng && v[i].proid==false)
		{
			min = i;
			minleng = v[i].minleng;
			//cout << min << " " << minleng << endl;
		}
	}
		//cout << v[min].number <<" індекс вершини з найменшим шляхом"<< endl;

	//визначення вершини відстань до якої найменша.

	for (int i = 0; i < n; i++)
	{
		if (v[min].number == reb[i].v1 || v[min].number == reb[i].v2)
		{
			x++;
		}
	}
	//cout <<"суміжних непройдених вершин: " << x<<endl;
	//визначення кількості суміжих вершин.

	int *mas_sum = new int[x];
	//виділення пам'яті під суміжні вершини.

	m = 0;
	for (int i = 0; i < n; i++)
	{
		if (v[min].number == reb[i].v1)
		{
			for (int j = 0; j < 30; j++)
```

```
                        {
                                if (v[j].number == reb[i].v2 && v[j].proid==false)
                                {
                                        mas_sum[m] = j;

                                        v[j].wiev = true;

                                        //cout << v[j].number << " -індекс суміжної ";
                                        m++;

                                        break;
                                }
                        }
                }
                if (v[min].number == reb[i].v2)
                {
                        for (int j = 0; j < 30; j++)
                        {
                                if (v[j].number == reb[i].v1&& v[j].proid == false)
                                {
                                        mas_sum[m] = j;

                                        v[j].wiev = true;

                                        //cout << v[j].number << "-індекс суміжної ";
                                        m++;
                                }
                        }
                }
        }
        //заповнення масиву суміжних вершин.

        for (int i = 0; i < m; i++)
        {
                for (int j = 0; j < n; j++)
                {

                        if ((reb[j].v1 == v[min].number || reb[j].v2 ==v[min].number) &&
(reb[j].v1 == v[mas_sum[i]].number || reb[j].v2 == v[mas_sum[i]].number))
                        {
                                //cout << i <<" "<<j<< endl;
                                if (v[mas_sum[i]].minleng > (v[min].minleng +
reb[j].leng))
                                {
                                        v[mas_sum[i]].minleng = v[min].minleng +
reb[j].leng;
                                        //cout << v[mas_sum[i]].number << "
"<<v[mas_sum[i]].minleng << " індекс/довжина шляху" << endl;
                                        v[mas_sum[i]].way = v[min].way;
                                        ss[t] << v[mas_sum[i]].number;
                                        ss[t] >> str;
                                        v[mas_sum[i]].way += ( " -> " + str);
                                        //cout << v[mas_sum[i]].way <<" шлях"<< endl;
                                        str = "";
                                        t++;
                                }

                        }
                }

        }
```

```
        //визначення яка з відстаней до суміжних вершин (нова чи стара) більша.

                v[min].proid = true;

        min = end-1;
        m = 0;
        k = 0;
        x = 0;
        minleng = 1000;
        delete[] mas_v;
        delete[] mas_sum;
}


        cout << "Відстань : " << v[end-1].minleng << endl;
        cout << "Шлях  " << v[end-1].way << endl;
        cout << endl;


delete[] reb;
return 0;
}
```

Вміст файлу:

6

1

2

1

2

3

1

3

4

3

4

5

3

5

6

2

7

8

1

8

9

4

9

10

2

10

11

4

11

12

7

13

14

1

14

15

2

15

16

3

16

17

7

17

18

7

19

20

3

20

21

1

21

22

8

22

23

5

23

24

4

25

26

7

26

27

3

27

28

3

28

29

6

29

30

4

1

7

5

7

13

5

13

19

8

19

25

8

2

8

1

8

14

7

14

20

2

20

26

3

3

9

3

9

15

1

15

21

1

21

27

1

4

10

4

10

16

4

16

22

3

22

28

5

5

11

1

11

17

2

17

23

3

23

29

7

6

12

7

12

18

8

18

24

7

24

30

Результат роботи програми:

З якої вершини почати? (не більше 30) 1

До якої вершини знайти шлях?(не більше 30) 30

Відстань : 22

Шлях 1 -> 7 -> 8 -> 14 -> 15 -> 21 -> 22 -> 28 -> 29 -> 30

C:\Users\Misha_Sydoruk\source\repos\diskretka5\Debug\diskretka5.exe (процесс 3392) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

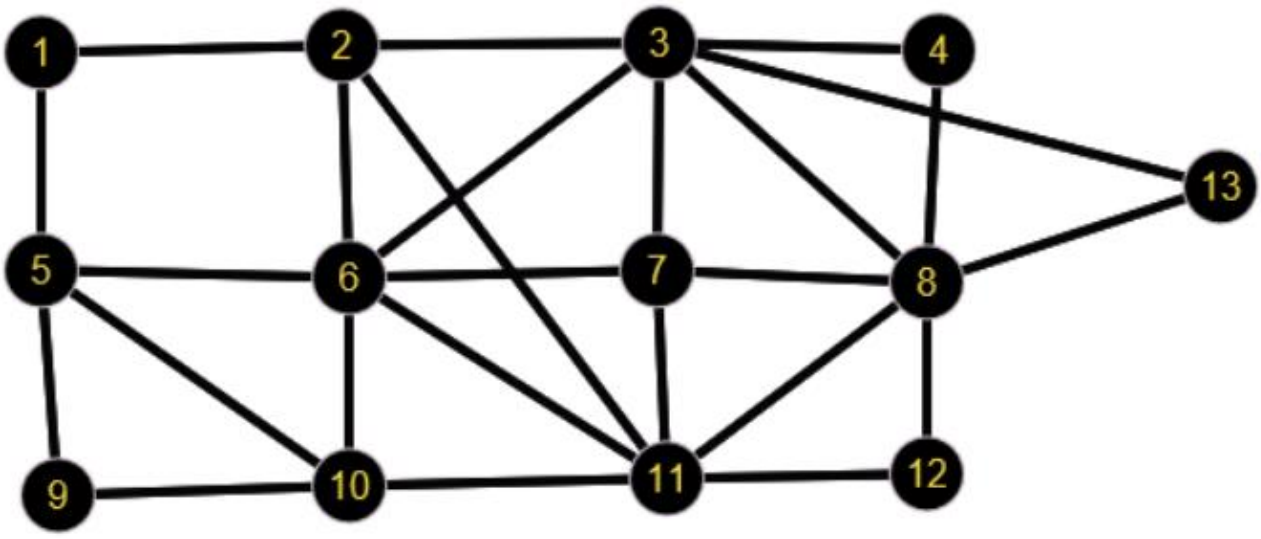
Чтобы закрыть это окно, нажмите любую клавишу…

Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері;

б) елементарних циклів



Примітка:

Нумерація вершин графа не збігається з тією, що у завданні, але граф той самий.

Розв’язання:

Виділимо **прості цикли** в графі:

1 2 6 5

2 6 7 3

3 7 8 4

5 6 10 9

6 10 11 7

7 11 12 8

5 10 6

2 6 11

3 8 11 6

3 13 8

1 2 3 4 8 12 11 10 9 5

3 8 11 6

Починаємо грамотно об’єднувати цикли:

Розпочинаємо з вершини 5

5-1-2-6-5

Рухаємося по крайніх ребрах:

5-9-10-11-12-8

Включимо цикл 3-13-8

8-13-3-8-4-3-2-11-7-3-6-11-8-7-6-10-5

Шуканий Ейлеровий цикл:

5-1-2-6-5-9-10-11-12-8-13-3-8-4-3-2-11-7-3-6-11-8-7-6-10-5

Ще одним варіантом буде наступний цикл:

3-6-11-2-6-7-3-2-1-5-6-10-5-9-10-11-8-3-4-8-12-11-7-8-13-3

Ще один Ейлеровий цикл можна знайти **методом Флері:**

Він полягає в проходженні по ребру за умови якщо це проходження не розіб'є граф на 2 підграфи.

11-2-3-4-8-13-3-8-12-11-1-7-8-11-10-9-5-10-6-2-1-5-6-7-3-6-11

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

19. $\overline{\overline{xy}(x\bar{y}z \vee \bar{x}y)}$

¬(¬(xy)^(x(¬y)z v (¬x)y))=

xy v ¬(x(¬y)z v (¬x)y)=

=xy v (¬(x(¬y)z ^ ¬(¬xy)=

=xy v (¬x v y v ¬ z) ^ (x v ¬y)=

=xy v (x v ¬y) ^ (¬x v y) v (x v ¬y) ^ ¬z=

=xy v (¬x v y) ^x v (¬x v y) ^ ¬y v x ^¬z v ¬y ^¬z=

=xy v x¬x v xy v ¬x¬y v y¬y v x¬z v ¬y¬z=

=xy v ¬x¬y v xz v ¬y¬z.