МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ НАЦІОНАЛЬНИЙ

УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Звіт

До розрахункової роботи

З дисципліни :

Дискретна Математика

Виконав:

Студент групи КН-113

Стасишин Р. О.

Викладач:

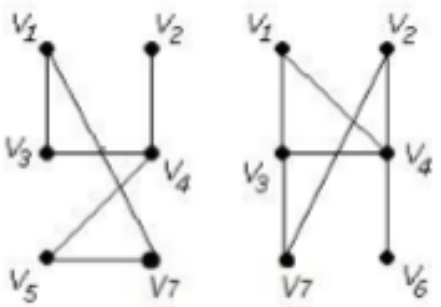
Мельникова.Н.І

Варіант № 17

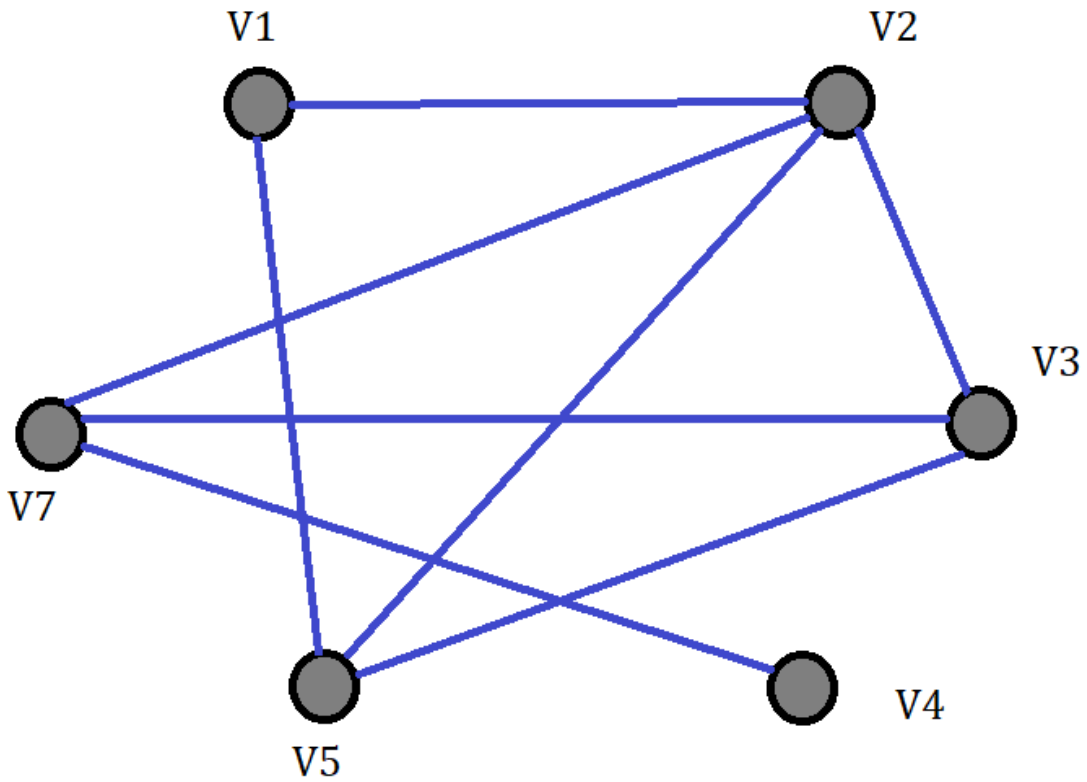
Завдання № 1

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.

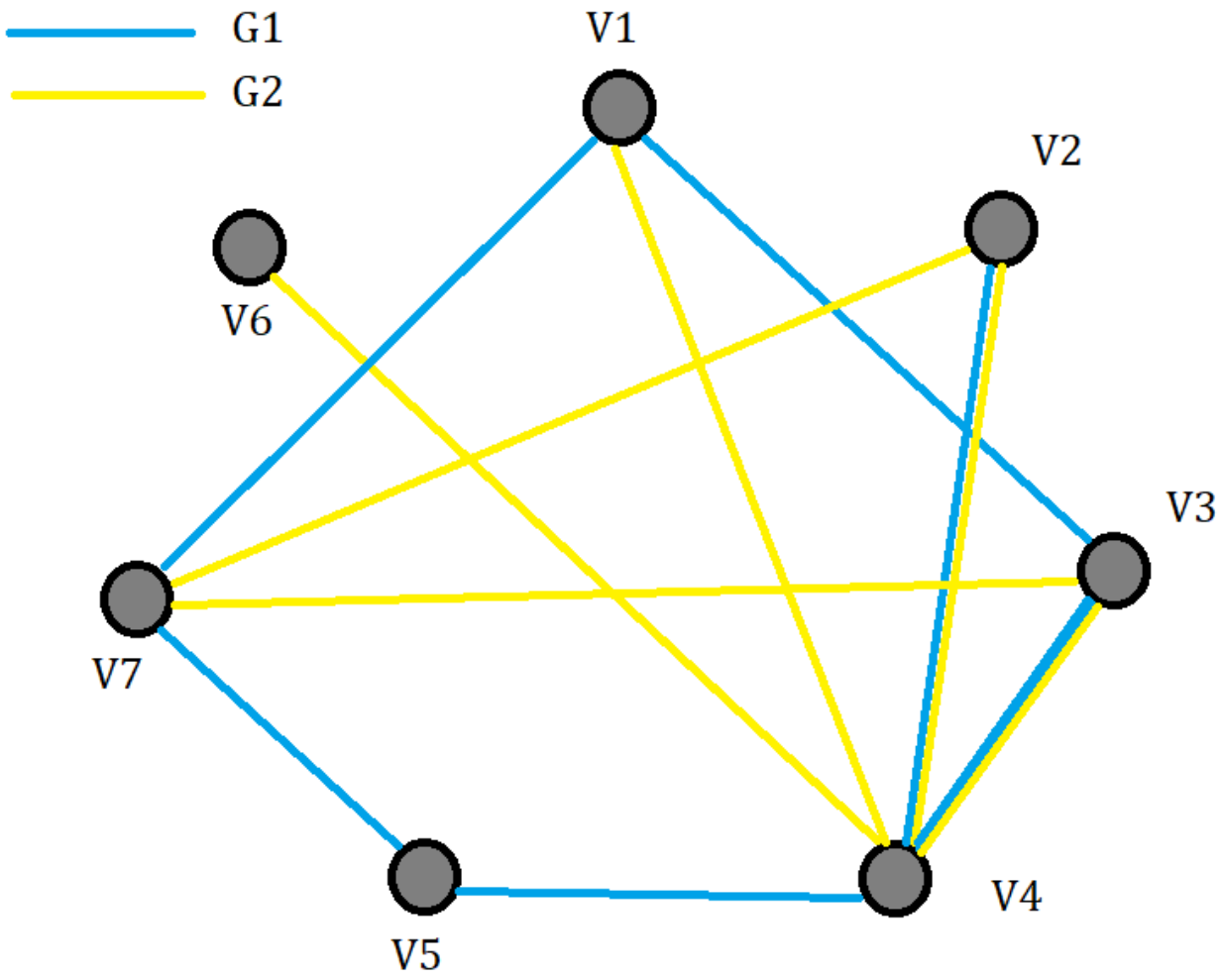
17)



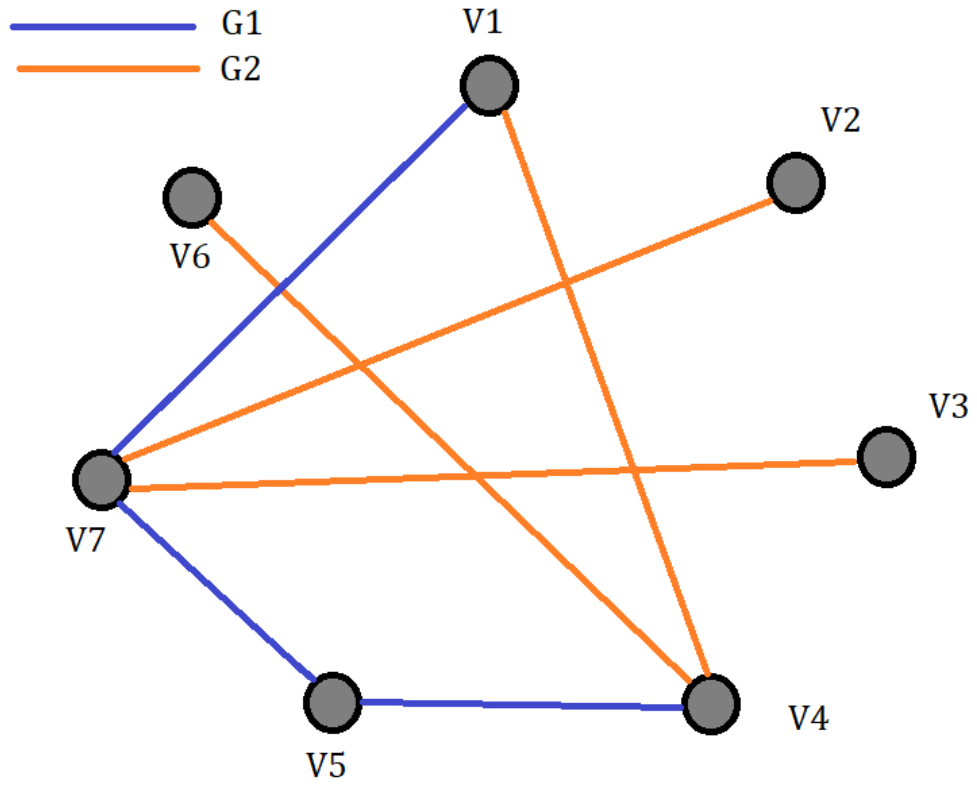
1) Доповнення до графа це створення графа який має ті ж вершини але ребра які відсутні на графі G1



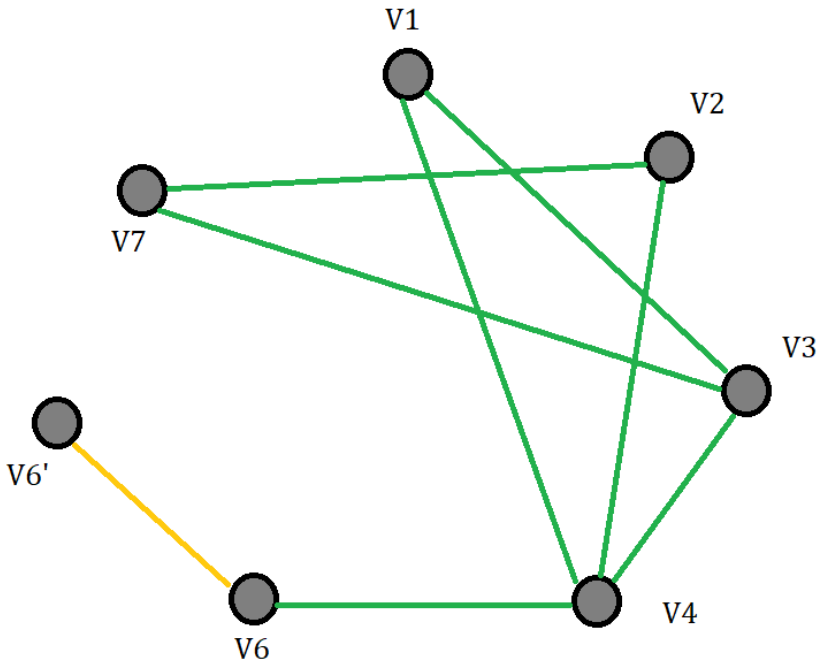
2)Обєднання графів це всі вершини графа G1 і G2 і вершини які були в G1 і G2. І все це обєднуємо в 1 граф



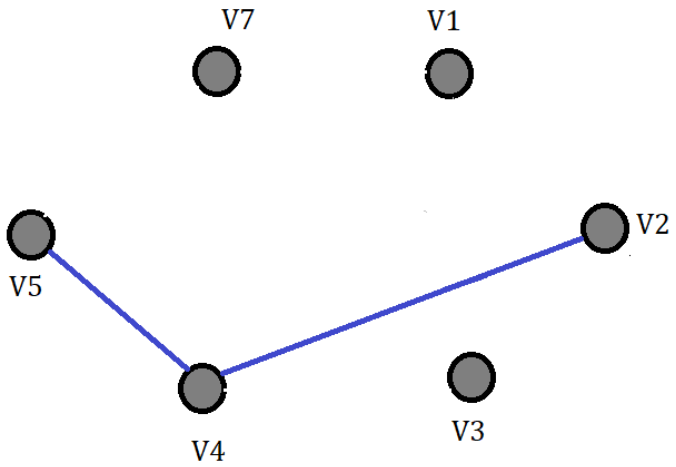
3) Кільцева Сума



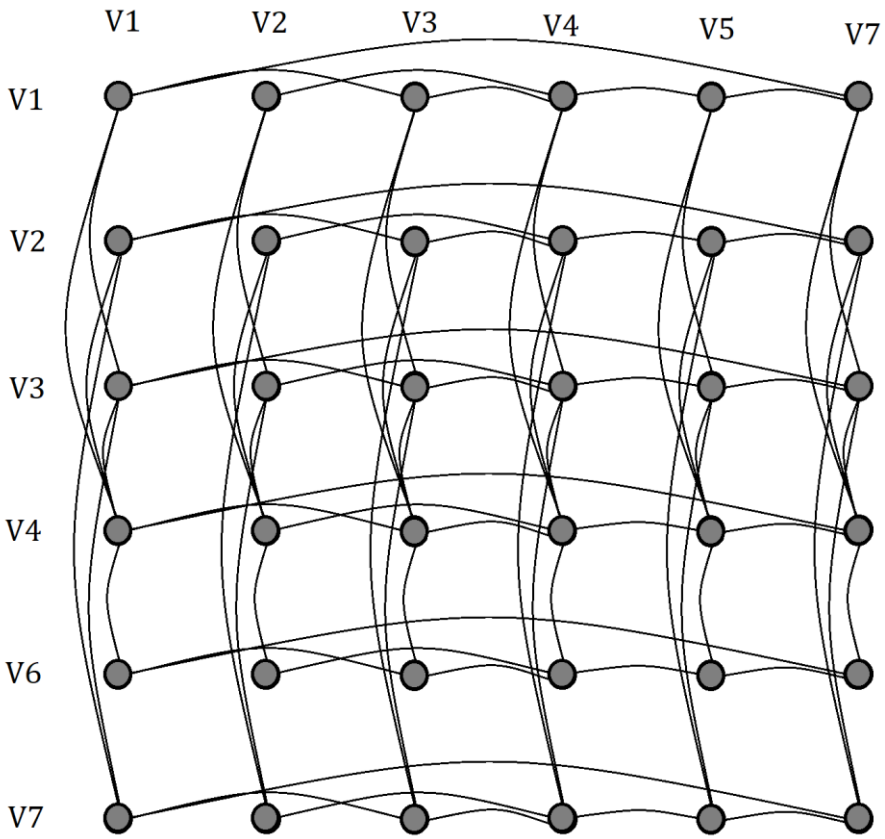
4)Розмноження вершини



5) Виділити підграф A що складається з 3-ох вершин G1



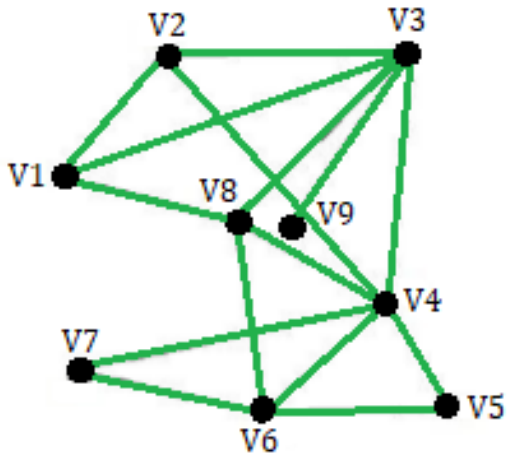
6)Добуток графів



Завдання № 2

Скласти таблицю суміжності для орграфа.

17)

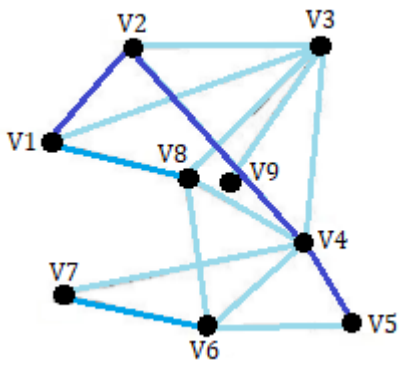


Матриця суміжності

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

Завдання № 3

Для графа з другого завдання знайти діаметр.



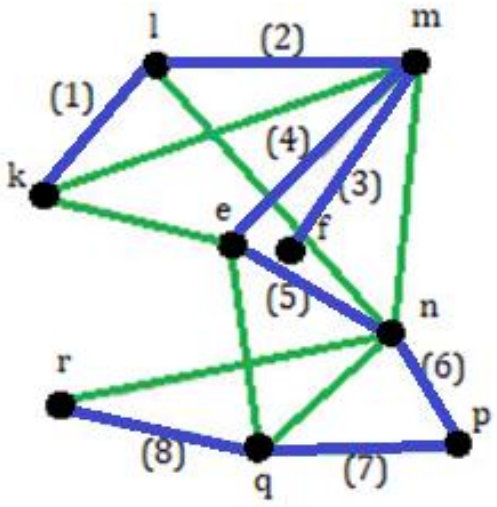
Діаметр графа з завдання №2

Діаметр графа = (**5** -> **4** -> **2** -> **1**)

Діаметр = 3

Завдання № 4

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

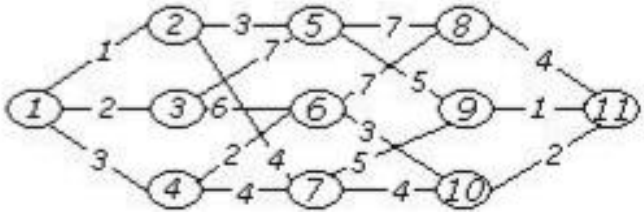


Вершина	DFS номер	Вміст стеку
k	1	k
l	2	kl
m	3	klm
f	4	klmf
_	_	klm
e	5	klme
n	6	klmen
p	7	klmenp
q	8	klmenpq
r	9	klmenpqr
_	_	klmenpq
_	_	klmenp
_	_	klmen
_	_	klme
_	_	klm
_	_	kl
_	_	k
_	_	Ø

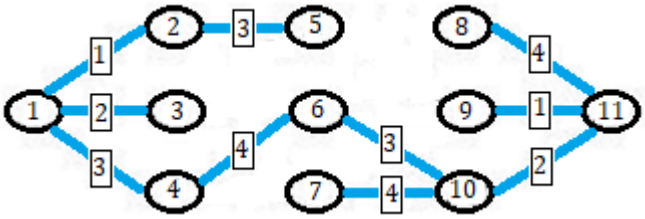
Завдання № 5

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

17)



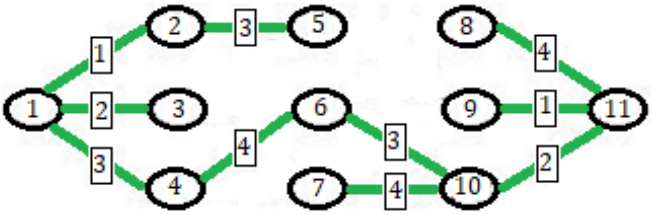
1)Краскала



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

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

2) Прима



	1	2	3	4	5	6	7	8	9	10	11
1)	0	1	∞	∞	∞	∞	∞	∞	∞	∞	∞
2)	0	1	2	∞	∞	∞	∞	∞	∞	∞	∞
3)	0	1	2	3	∞	∞	∞	∞	∞	∞	∞
4)	0	1	2	3	∞	2	∞	∞	∞	∞	∞
5)	0	1	2	3	3	2	∞	∞	∞	∞	∞
6)	0	1	2	3	3	2	∞	∞	∞	3	∞
7)	0	1	2	3	3	2	∞	∞	∞	3	2
8)	0	1	2	3	3	1	4	∞	1	3	2
9)	0	1	2	3	3	1	4	∞	1	3	2
10)	0	1	2	3	3	1	4	4	1	3	2

Завдання № 6

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

17)

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

1. Вихідна вершина 1:

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

Довжина шляху = 1 + 1 + 1 + 1 + 2 + 2 + 7 + 6 = 21

2. Вихідна вершина 2:

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

Довжина шляху = 3 + 1 + 1 + 2 + 3 + 5 + 5 + 7 = 27

3. Вихідна вершина 3:

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

Довжина шляху = 4 + 1 + 1 + 2 + 3 + 5 + 5 + 7 = 28

4. Вихідна вершина 4: 4 -> 5 -> 7 -> 6 -> 8 -> 3 -> 1 -> 2 -> 4

Довжина шляху = 1 +1 + 2 + 3 + 5 + 5 + 5 + 3 = 25

5. Вихідна вершина 5:

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

Довжина шляху = 1 + 1 + 3 + 2 + 4 + 5 + 5 + 3 = 24

6. Вихідна вершина 6:

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

Довжина шляху = 2 + 1 + 1 + 1 + 5 + 5 + 7 + 6 = 28

7. Вихідна вершина 7:

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

Довжина шляху = 1 + 1 + 1 + 3 + 5 + 5 + 5 + 4 = 25

8. Вихідна точка 8

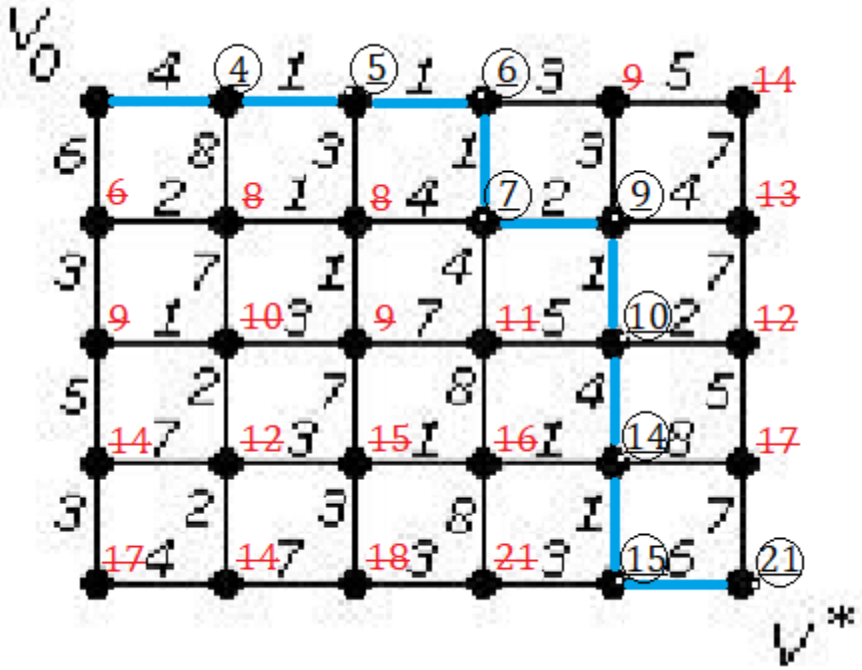
8 -> 4 -> 7 -> 1 -> 5 -> 2 -> 3 -> 6 -> 8

Довжина шляху = 2 + 1 + 1 + 1 + 1 + 5 + 7 + 2 = 20

Найоптимальніший варіант № 8 = 20 …

Завдання № 7

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

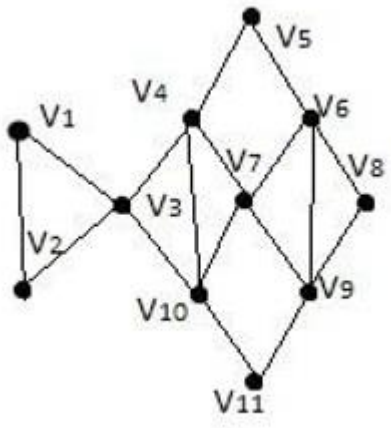


Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері;

б) елементарних циклів.



а) V3 => V1 => V2 => V3 => V4 => V5 => V6 => V7 => V4 => V10 => V7 => V9 => V6 => V8 => V9 => V11 => V10;

б) 3 - 1 - 2 - 3; 3 - 1 - 2 - 3 - 4 - 5; 3 - 1 - 2 - 3 - 4 - 5 - 6 – 7; 3 - 1 - 2 - 3 - 4 - 5 - 6 – 7 - 4 - 10;
3 - 1 - 2 - 3 - 4 - 5 - 6 – 7 - 4 - 10 - 7 - 9; 3 - 1 - 2 - 3 - 4 - 5 - 6 – 7 - 4 - 10 - 7 - 9 - 6 - 8;

3 - 1 - 2 - 3 - 4 - 5 - 6 – 7 - 4 - 10 - 7 - 9 - 6 - 8 - 9 - 11 - 10;

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

17. $x\bar{y} \vee \bar{x}\bar{z} \vee yz$

Напишіть алгоритм

60. Обхід графа вглиб та вшир.

61. Прима знаходження найменшого остову.

62. Краскала знаходження найменшого остову.

63. Дейкстра знаходження найкоротшого ланцюга між парою вершин.

64. «Іди в найближчий» для розв’язання задачі комівояжера.

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

1.Обхід вглиб

```
#include <iostream>
using namespace std;
const int n = 9;
int i, j;
bool* visited = new bool[n];
int graph[n][n] =
{
{0,1,0,0,0,0,1,1,1},
{1,0,1,0,0,0,0,1,0},
{0,1,0,1,0,1,0,1,0},
{0,0,1,0,1,0,1,0,0},
{0,0,0,1,0,1,0,1,0},
{0,0,1,0,1,0,0,1,0},
{1,0,0,1,0,0,0,1,0},
{1,1,1,0,1,1,1,0,0},
{1,0,0,0,0,0,0,0,0}
};
void DFS(int st)
{
  int r;
  cout << st + 1 << " ";
```

```
  visited[st] = true;
  for (r = 0; r <= n; r++)
    if ((graph[st][r] != 0) && (!visited[r]))
      DFS(r);
}
int main()
{
  int start;
  cout << "Matrix: " << endl;
  for (i = 0; i < n; i++)
  {
    visited[i] = false;
    for (j = 0; j < n; j++)
      cout << " " << graph[i][j];
    cout << endl;
  }
  cout << "Start edge >> "; cin >> start;
  bool* vis = new bool[n];
  cout << "Path: ";
  DFS(start - 1);
  delete[]visited;
  system("pause>>void");
}
```

Результат.

```
Matrix:
 0 1 0 0 0 0 1 1 1
 1 0 1 0 0 0 0 1 0
 0 1 0 1 0 1 0 1 0
 0 0 1 0 1 0 1 0 0
 0 0 0 1 0 1 0 1 0
 0 0 1 0 1 0 0 1 0
 1 0 0 1 0 0 0 1 0
 1 1 1 0 1 1 1 0 0
 1 0 0 0 0 0 0 0 0
Start edge >> 3
Path: 3 2 1 7 4 5 6 8 10 9
Process returned 0 (0x0)   execution time : 21.962 s
Press any key to continue.
```

Алгоритм Прима

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

void output(int size, int** adjacencyarr)
 {
 cout << "  ";
for (int i = 0; i < size; i++)
{
 cout << setw(3) << i + 1;
 }  for (int i = 0; i < size; i++)
{   cout << endl << setw(3) << i + 1;
   for (int j = 0; j < i; j++)
 {    cout << setw(3) << adjacencyarr[j][i];
 }   cout << "  -";
for (int j = i + 1; j < size; j++)
{    cout << setw(3) << adjacencyarr[i][j];   }
  }
}

int main()
{  int size;
  cout << "Enter amount of vertices ";
cin >> size;
 int** adjacencyarr = new int* [size];
 bool* vertex = new bool[size];
 cout << "Enter adjacency matrix \n";
 cout << "  ";
 for (int i = 0; i < size; i++)
```

{ cout << setw(3) << i + 1;

adjacencyarr[i] = new int[size];

vertex[i] = 0; }

vertex[0] = 1;

cout << endl; for (int i = 0; i < size; i++) { cout << setw(2) << i + 1; for (int j = 0; j < i; j++) { cout << setw(3) << adjacencyarr[j][i]; } cout << " -"; for (int j = i + 1; j < size; j++) { cin >> adjacencyarr[i][j]; } } cout << endl; int min, minI, minJ; for (int n = 0; n < size - 1; n++) { min = 100, minI = 0, minJ = 0; for (int i = 0; i < size; i++) { for (int j = i + 1; j < size; j++) { if (adjacencyarr[i][j] < min && adjacencyarr[i][j] != 0 && vertex[i] != vertex[j]) {

min = adjacencyarr[i][j]; minI = i, minJ = j; } } } vertex[minI] = 1; vertex[minJ] = 1; cout << n + 1 << ")conected " << minI + 1 << "-" << minJ + 1 << endl; } }

Результат.

```
Enter amount of vertices 11
Enter adjacency matrix
    1  2  3  4  5  6  7  8  9 10 11
 1  -  1  2  3  0  0  0  0  0  0  0
 2  1  -  0  0  3  0  4  0  0  0  0
 3  2  0  -  0  7  6  0  0  0  0  0
 4  3  0  0  -  0  2  4  0  0  0  0
 5  0  3  7  0  -  0  0  7  5  0  0
 6  0  0  6  2  0  -  0  7  0  3  0
 7  0  4  0  4  0  0  -  0  5  4  0
 8  0  0  0  0  7  7  0  -  0  0  4
 9  0  0  0  0  5  0  5  0  -  0  1
10  0  0  0  0  0  3  4  0  0  -  2
11  0  0  0  0  0  0  0  4  1  2  -
1)conected 1-2
2)conected 1-3
3)conected 1-4
4)conected 4-6
```

Алгоритм краскала

#include <stdio.h> #include <iostream> #include <stdlib.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() {  setlocale(LC_ALL, "Ukrainian");  int A[11][11] =  { 0, 1, 2, 3, 0, 0, 0, 0, 0, 0, 0,    1, 0, 0, 0, 3, 0, 4, 0,
0, 0, 0,    2, 0, 0, 0, 7, 6, 0, 0, 0, 0, 0,    3, 0, 0, 0, 0, 2, 4, 0, 0, 0, 0,    0, 3, 7, 0, 0, 0, 0, 7, 5, 0, 0,    0, 0, 6, 2, 0, 0,
0, 7, 0, 3, 0,    0, 4, 0, 4, 0, 0, 0, 0, 5, 4, 0,    0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 4,    0, 0, 0, 0, 5, 0, 5, 0, 0, 0, 1,    0, 0, 0, 0,
0, 3, 4, 0, 0, 0, 2,

  0, 0, 0, 0, 0, 0, 0, 4, 1, 2, 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++){     if
(A[j - 1][k - 1] == i && InDifferTrees(11, B, j, k)){      AddToTheTree(11, B, j, k);      cout << " " << j << "-" << k;
}    }   } } return 0; }


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]; } //Перевірте відсортовані вузли та додайте до дерева


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;  //Лінія  for (int i = 0; i < n; i++){
temp1 = 0;   temp2 = 0;   //перший елемент   for (int j = 0; j < n; j++){    if (A[i][j] == first){     temp1 = 1;    }  }
//другий елемент   for (int k = 0; k < n; k++){    if (A[i][k] == second){     temp2 = 1;    }   if (temp1 &&
temp2){    return 0;   } } return 1; }
```

алгоритм дейкстри

```
#define _CRT_SECURE_NO_WARNINGS #include <iostream> #include <fstream> #include <locale> using namespace std;


int main() {  ifstream fin("MyFile.txt");  setlocale(LC_ALL, "Ukrainian");  int versh, rebro;  fin >> versh >> rebro;  const int SIZE = 30;  int matr[SIZE][SIZE]; // матриця зв'язків  int dist[SIZE]; // мінімальна відстань  int visit[SIZE];  // чи відвідані вершни   int dis, top, min;  int start_index = 0;


for (int i = 0; i < versh; i++){    // Ініціалізація матриці зв'язків   dist[i] = 99999;   visit[i] = 0;   for (int j = 0; j < versh; j++){    matr[i][j] = 0;    matr[j][i] = 0;   }  }  for (int i = 0; i < rebro; i++) {   int v1, v2, dis;   fin >> v1 >> v2 >> dis;   matr[v1 - 1][v2 - 1] = dis;   matr[v2 - 1][v1 - 1] = dis;  }  cout << "Алгоритм Дейкстра\n";


dist[0] = 0;  do {   top = 99999;   min = 99999;   for (int i = 0; i < versh; i++){    if (visit[i] == 0 && dist[i] < min){     min = dist[i];     top = i;

}   }   if (top != 99999){    for (int i = 0; i < versh; i++){     if (matr[top][i] > 0){      dis = min + matr[top][i];      if (dis < dist[i]){       dist[i] = dis;     }    }    }    visit[top] = 1;   }  } while (top < 99999);


int end = versh - 1;  int waga = dist[end];  int way[30];  way[0] = versh - 1;  int k = 1;  while (end != 0){   for (int i = 0; i < versh; i++){    if (matr[end][i] > 0){     if (dist[i] == waga - matr[end][i]){      waga = dist[i];      way[k] = i;      end = i;      k++;     }    }   }  }  cout << "\nНайменший шлях з V0 в V29: ";  for (int i = k; i > 0; i--){   if (i - 1 > 0)    cout << way[i - 1] << " -> ";   else    cout << way[i - 1];  }  cout << "\n\nДовжина відстані: " << dist[versh - 1] << endl;
```

Алгоритм комівояжера

```
#include <iostream>   #include <stdio.h>   #include <string>   #include <fstream>  using namespace std;  ifstream fin; string path = "MyFile1.txt";


struct mass{     int mas[9]; };


int** input() {     int coun = 8;     string str;     str = "";


fin.open(path);     int** arr;     arr = new int* [coun];     for (int i = 0; i < coun; i++)        arr[i] = new int[coun];     for (int i = 0; i < coun; i++){        for (int j = 0; j < coun; j++)           arr[i][j] = 0;     }     for (int i = 0; i < coun; i++){        for (int j = i + 1; j < coun; j++){           getline(fin, str);           arr[i][j] = atoi(str.c_str());  arr[j][i] = atoi(str.c_str());        }     }     fin.close();     return arr; } bool compon(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 povt(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 (compon(mas, count)){  while (mas[var] != count){  mas[var]++;  if (povt(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 (povt(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 (compon(mas, count)){  while (mas[var] != count){  mas[var]++;  if (povt(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 (povt(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; }
```

Алгоритм флейрі

```
#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]; int 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 i;  }  return 0;  } 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]){ //when (u,v) edge is presnt and not forming bridge  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];    cout << "Euler Path Or Circuit: ";    fleuryAlgorithm(findStartVert()); }
```