Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2

з дисципліни «СРМ-1. Дискретна математика»

на тему

«**Характеристики графів**»

Виконав:
студент гр. ІС-12
Фундерат Денис

Викладач:
доц. Рибачук Л.В.

Київ – 2021

Зміст

1 Постановка задачі

Реалізувати програмне застосування (програму), яке виконує наступні функції. Причому на вхід програми подається вхідний файл з описом графу, зі структурою, яка вказана у практичному завданні No1 «Представлення графів».

1. Визначити степінь вершин графу. За запитом користувача програма на екран та/або у файл виводить степінь усіх вершин графу (напівстепені виходу та заходу). Визначити, чи граф є однорідним та якщо так, то вказати степінь однорідності графу.

2. Визначити всі висячі та ізольовані вершини. За запитом користувача програма на екран виводить перелік усіх висячих та ізольованих вершин графу.

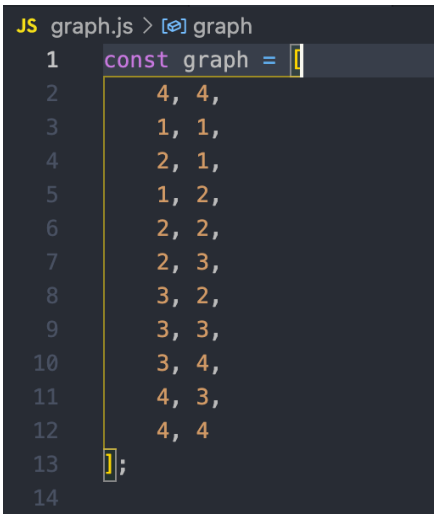
3. Визначення метричних характеристик графу. Програма виводить наступні характеристики:

    - Діаметр графу
    - Радіус графу
    - Центр графу
    - Яруси графу із переліком вершин, які входять до кожного ярусу

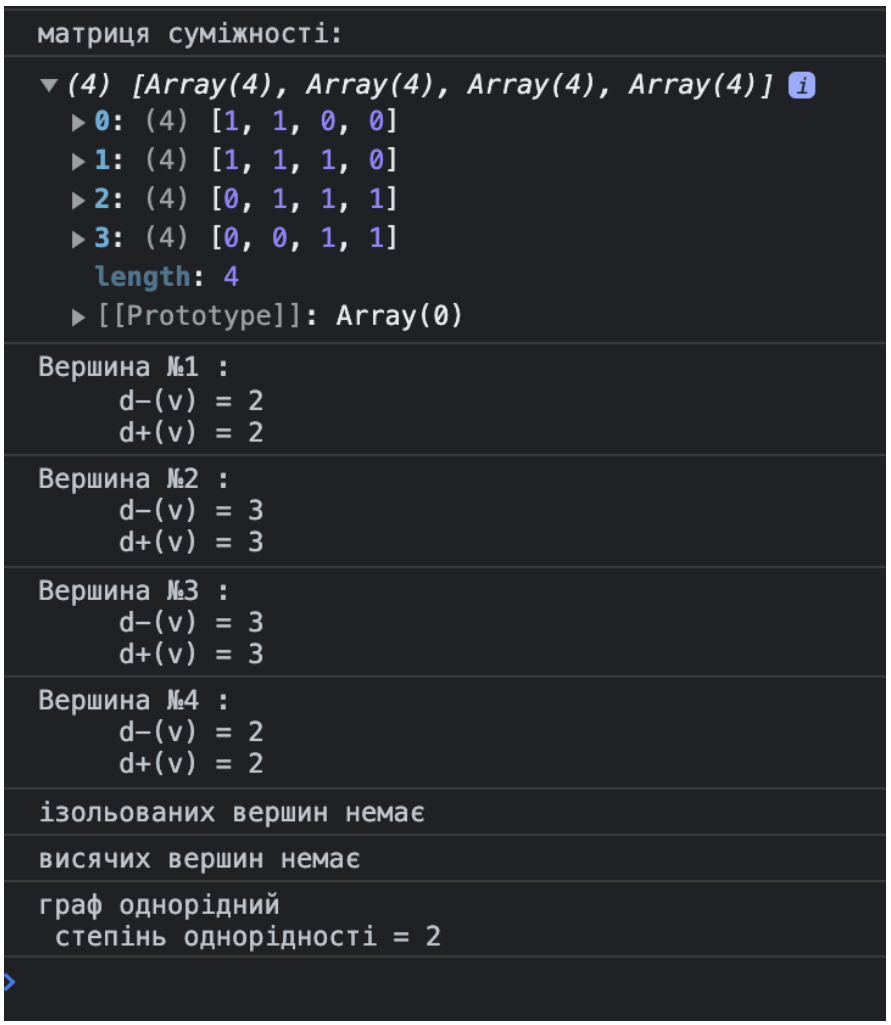
2. Результати виконання програми

2.1. Випадок, коли граф однорідний:

Вхідні дані:



Результат виводу в консоль:



2.2. Випадок, коли граф має ізольовану та висячу вершини:

Вхідні дані:

```
const graph = [
    4, 3,
    2, 2,
    2, 1,
    4, 4
];
```

Результат виводу в консоль:

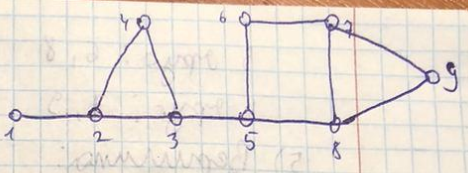
```
матриця суміжності:
▼ (4) [Array(4), Array(4), Array(4), Array(4)]
  ▶ 0: (4) [0, 0, 0, 0]
  ▶ 1: (4) [1, 1, 0, 0]
  ▶ 2: (4) [0, 0, 0, 0]
  ▶ 3: (4) [0, 0, 0, 1]
    length: 4
  ▶ [[Prototype]]: Array(0)
Вершина №1 :
     d-(v) = 0
     d+(v) = 1
Вершина №2 :
     d-(v) = 2
     d+(v) = 1
Вершина №3 :
     d-(v) = 0
     d+(v) = 0
Вершина №4 :
     d-(v) = 1
     d+(v) = 1
ізольовані вершини: №3
висячі вершини: №1
граф неоднорідний
>
```

2.3. Визначення метричних характеристик:

```
const graph = [
    9, 11,
    1, 2,
    2, 3,
    2, 4,
    3, 4,
    3, 5,
    5, 6,
    5, 8,
    6, 7,
    7, 8,
    7, 9,
    8, 9,
];
```

```
Вершина №1 :
     d-(v) = 1
     d+(v) = 0
Вершина №2 :
     d-(v) = 2
     d+(v) = 1
Вершина №3 :
     d-(v) = 2
     d+(v) = 1
Вершина №4 :
     d-(v) = 0
     d+(v) = 2
Вершина №5 :
     d-(v) = 2
     d+(v) = 1
Вершина №6 :
     d-(v) = 1
     d+(v) = 1
Вершина №7 :
     d-(v) = 2
     d+(v) = 1
Вершина №8 :
     d-(v) = 1
     d+(v) = 2
Вершина №9 :
     d-(v) = 0
     d+(v) = 2
ізольованих вершин немає
висячі вершини: №1
граф неоднорідний
---------------- метричні характеристики ----------------
діаметр: 5
радіус: 0
центр: 4 9
```

Ярусы:

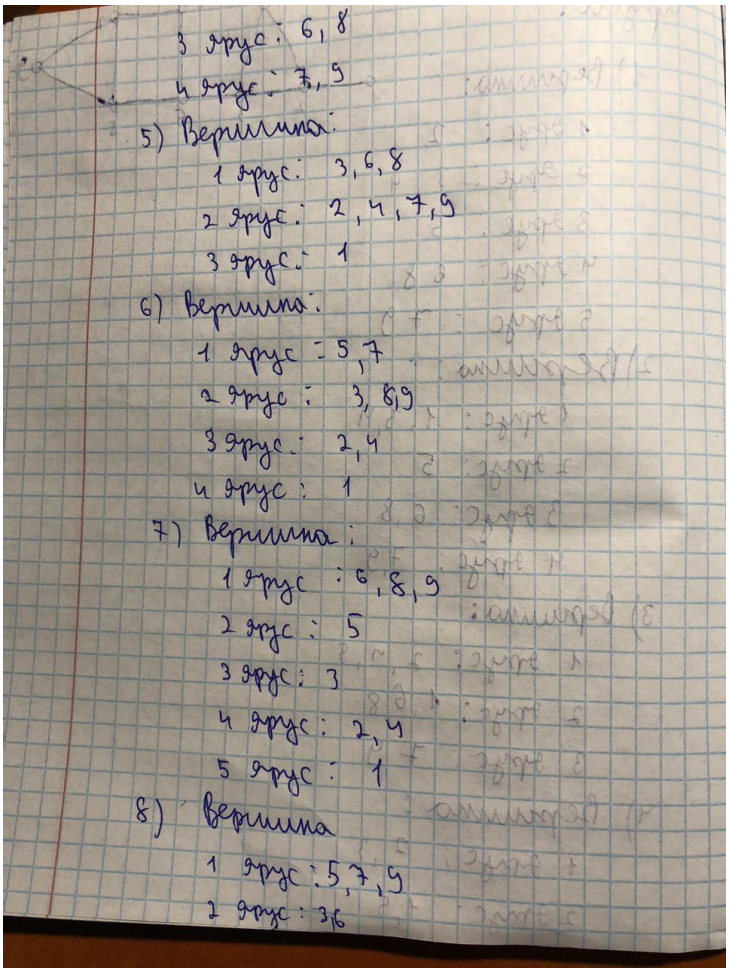


1) Вершина:
   1 ярус: 2
   2 ярус: 3, 4
   3 ярус: 5
   4 ярус: 6 8
   5 ярус - 7 9

2) Вершина:
   1 ярус: 1, 3, 4
   2 ярус: 5
   3 ярус: 6, 8
   4 ярус: 7, 9

3) Вершина:
   1 ярус: 2, 4, 5
   2 ярус: 1, 6, 8
   3 ярус: 7, 9

4) Вершина:
   1 ярус: 2, 3
   2 ярус: 4, 5



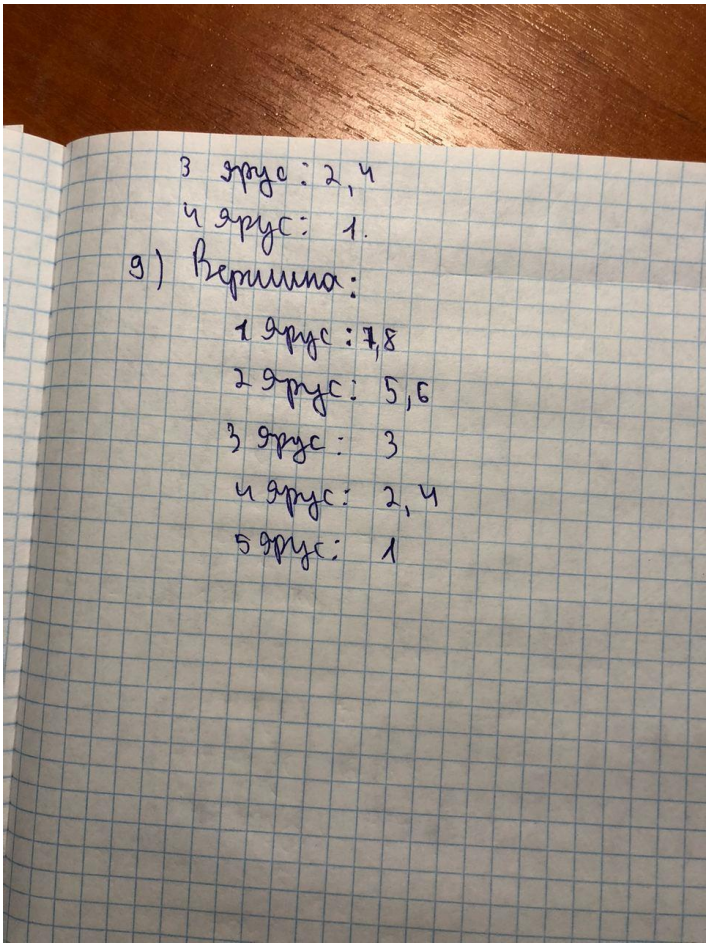
3 ярус: 6, 8
4 ярус: 7, 9

5) Вершина:
1 ярус: 3, 6, 8
2 ярус: 2, 4, 7, 9
3 ярус: 1

6) Вершина:
1 ярус: 5, 7
2 ярус: 3, 6, 9
3 ярус: 2, 4
4 ярус: 1

7) Вершина:
1 ярус: 6, 8, 9
2 ярус: 5
3 ярус: 3
4 ярус: 2, 4
5 ярус: 1

8) Вершина
1 ярус: 5, 7, 9
2 ярус: 3, 6



3 ярус: 2, 4
4 ярус: 1.

9) Вершина:
1 ярус: 7, 8
2 ярус: 5, 6
3 ярус: 3
4 ярус: 2, 4
5 ярус: 1

3 Лістинг програми

```
const n = graph[0];
const m = graph[1];
const n_column = [];
const m_column = [];

for (let i = 2; i < graph.length; i++) {
   if (i % 2 == 0) n_column.push(graph[i]);
   else m_column.push(graph[i]);
}



// ============ матриця суміжності ============
const adjMatrix = []; // adjacency matrix
let adjRow = [];

for (let i = 1; i <= n; i++) { // ініціалізація матриці (only with 0)
   adjRow = [];
   for (let j = 1; j <= n; j++) {
       adjRow.push(0);
   }
   adjMatrix.push(adjRow);
}



for (let i = 0; i < n_column.length; i++) { // підстановка 1
   adjMatrix[n_column[i]-1][m_column[i]-1] = 1;
}

console.log(`матриця суміжності:`);
console.log(adjMatrix);
// ========================
```

```
      // ============ сумма рядків та стовпців (визначення напівстепенів вершин)
============
      let rowSum = 0;
      const rowsSum_arr = [];
      let columnSum = 0;
      const columnsSum_arr = [];

      for (let i = 0; i < n; i++) {
         rowSum = 0;
         columnSum = 0;
         for (let j = 0; j < n; j++) {
             rowSum += adjMatrix[i][j];
             columnSum += adjMatrix[j][i];
         }
         rowsSum_arr.push(rowSum);
         columnsSum_arr.push(columnSum);
      }
      // ========================




      // ============ визначення висячих та ізольованих вершин ============
      let linesSum = 0;
      let isolV = [];
      let hangV = [];

      for (let i = 0; i < n; i++) {
          console.log(`Вершина №${i+1} : \n        d-(v) = ${rowsSum_arr[i]} \n
d+(v) = ${columnsSum_arr[i]}`); // виведення у консоль

         linesSum = rowsSum_arr[i] + columnsSum_arr[i];
         if (linesSum == 0) isolV.push(`№${i+1}`);
         if (linesSum == 1) hangV.push(`№${i+1}`);
      }

      if (isolV.length != 0) console.log(`ізольовані вершини: ${isolV.join(',
')}`);
      else console.log(`ізольованих вершин немає`);
      if (hangV.length != 0) console.log(`висячі вершини: ${hangV.join(', ')}`);
```

```
else console.log(`висячих вершин немає`);
// ========================



// ============ визначення однорідності ============
let k = 0;
let count = 0;
let homogen = false;

for (let i = 0; i < n; i++) {
   if (i == 0) k = rowsSum_arr[i];
   if (rowsSum_arr[i] == k && columnsSum_arr[i] == k) {
       count++;
   }
   if (count == n-1) homogen = true;
}
if (homogen == true) {
   console.log(`граф однорідний \n степінь однорідності = ${k}`);
}
else console.log(`граф неоднорідний`);
// ========================



// ============ матриця відстаней ============
const distMatrix = [];
let distMatrix_row = [];

for (let i = 0; i < n; i++) {
   distMatrix_row = [];
   for (let j = 0; j < n; j++) {
       if (i == j) distMatrix_row.push(0);
       else {
           if (adjMatrix[i][j] == 1) distMatrix_row.push(adjMatrix[i][j]);
           if (adjMatrix[i][j] == 0) distMatrix_row.push(null);
       }
   }
   distMatrix.push(distMatrix_row);
```

```
}


let expMatrix_forDist = [];
let iter = 1;

function distMatrixRender() {
   let ckeckOut_main = false;
   let ckeckOut_matrixExpon = false;
   iter++;

   for (let i = 0; i < n; i++) {
       for (let j = 0; j < n; j++) {
           if (distMatrix[i][j] === null) {
               if (ckeckOut_matrixExpon == false) {
                   matrixExpon(expMatrix_forDist);
                   ckeckOut_matrixExpon = true;
               }

               if (expMatrix_forDist[i][j] != 0) distMatrix[i][j] = iter;
               else if (expMatrix_forDist[i][j] == 0) {
                   if (iter == 40) distMatrix[i][j] = 0;
                   ckeckOut_main = true;
               }
           }
       }
   }

   if (ckeckOut_main == true) distMatrixRender();
   // else {
   //     console.log(`матриця відстані:`);
   //     console.log(distMatrix);
   // }
}
distMatrixRender();



let expMatrix_forReach = [];
```

```
    function matrixExpon(matrix) {
        let expMatrix_row = [];
        let expMatrix_num = 0;
        let ckeckOut = String(matrix);

        for (let i = 0; i < n; i++) {
            expMatrix_row = [];
            for (let k = 0; k < n; k++) {
                expMatrix_num = 0;
                for (let j = 0; j < n; j++) {
                            if (ckeckOut != '') expMatrix_num += matrix[i][j] *
adjMatrix[j][k];
                    else expMatrix_num += adjMatrix[i][j] * adjMatrix[j][k];
                }
                expMatrix_row.push(expMatrix_num);
            }
            matrix[i] = expMatrix_row;
        }
    }
    // =========================



    // ============ метричні характеристики ============
    console.log('----------------- метричні характеристики -----------------');

    let ecc = [];

    for (let i = 0; i < n; i++) {
        let maxSum_row = distMatrix[i][0];
        for (let j = 0; j < n; j++) {
            if (distMatrix[i][j] > maxSum_row) maxSum_row = distMatrix[i][j];
        }
        ecc.push(maxSum_row);
    }
    // console.log(ecc);

    // ------------ діаметр, радіус та центр------------
    let maxElem = ecc[0];
    let minElem = ecc[0];
```

```
let mildElems = 'центр: ';

for (let i = 0; i < n; i++) {
        if (maxElem < ecc[i]) maxElem = ecc[i];
        if (minElem > ecc[i]) minElem = ecc[i];
}
for (let i = 0; i < ecc.length; i++) {
   if (ecc[i] == minElem) mildElems += `${i+1} `;
}
console.log(`діаметр: ${maxElem}`);
console.log(`радіус: ${minElem}`);
console.log(mildElems);
// ------------------------
```