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

Лабораторна робота №1

з дисципліни «СРМ-1. Дискретна математика»

на тему

«**Представлення графів**»

Виконав:
студент гр. ІС-12
Фундерат Денис

Викладач:
доц. Рибачук Л.В.

Київ – 2021

Зміст

Зміст 3
1 Постановка задачі 4
2 Результати виконання програми 5
3 Лістинг програми 6

1 Постановка задачі

Реалізувати програмне застосування (програму), яке виконує наступні функції.

1. Зчитування графу з вхідного файлу. На вхід подається текстовий файл наступного вигляду:

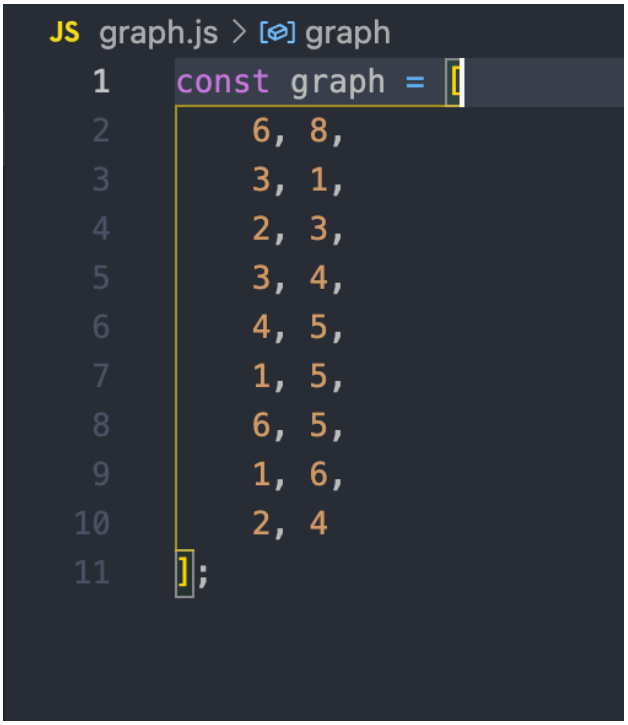
n m
v1 u1
v2 u2
....
vm um

Тут n – кількість вершин графу (ціле число, більше нуля), m – кількість ребер графу (ціле число, більше нуля), vi та ui – початкова та кінцева вершина ребра i (1vin, 1uin, цілі числа). Індексація вершин у файлі ведеться з 1. Вважається, що граф є орієнтованим.
Таким чином можна сказати, що граф задається у файлі списком ребер.

2. Вивід матриць інцидентності та суміжності. За вимогою користувача програма повинна виводити матриці інцидентності та суміжності (окремі функції) на екран та/або у текстовий файл, який вказує користувач.

2. Результати виконання програми

Вхідні дані:



Результат виводу в консоль:

```
матриця інцидентності:
▼ (6) [Array(8), Array(8), Array(8), Array(8), Array(8), Array(8)]
  ▶ 0: (8) [1, 0, 0, 0, -1, 0, -1, 0]
  ▶ 1: (8) [0, -1, 0, 0, 0, 0, 0, -1]
  ▶ 2: (8) [-1, 1, -1, 0, 0, 0, 0, 0]
  ▶ 3: (8) [0, 0, 1, -1, 0, 0, 0, 1]
  ▶ 4: (8) [0, 0, 0, 1, 1, 1, 0, 0]
  ▶ 5: (8) [0, 0, 0, 0, 0, -1, 1, 0]
    length: 6
  ▶ [[Prototype]]: Array(0)
----------------------
матриця суміжності:
▼ (6) [Array(6), Array(6), Array(6), Array(6), Array(6), Array(6)]
  ▶ 0: (6) [0, 0, 0, 0, 1, 1]
  ▶ 1: (6) [0, 0, 1, 1, 0, 0]
  ▶ 2: (6) [1, 0, 0, 1, 0, 0]
  ▶ 3: (6) [0, 0, 0, 0, 1, 0]
  ▶ 4: (6) [0, 0, 0, 0, 0, 0]
  ▶ 5: (6) [0, 0, 0, 0, 1, 0]
    length: 6
  ▶ [[Prototype]]: Array(0)
>
```

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 інцидентності ============
const incMatrix = []; // incidence matrix
let incColumn = [];

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


let k = 0;
for (let column of incMatrix) { // підстановка 1
   column[n_column[k]-1] = -1;
   column[m_column[k]-1] = 1;
   k++;
}


// ------- Matrix Render -------
const incMatrix_res = [];
let incRow_res = [];

for (let i = 0; i < n; i++) {
   incRow_res = [];
   for (let j = 0; j < m; j++) {
```

```
            incRow_res.push(incMatrix[j][i]);
        }
        incMatrix_res.push(incRow_res);
    }

    console.log(`матриця інцидентності:`);
    console.log(incMatrix_res);
    // ========================




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