Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра штучного інтелекту та програмного забезпечення

ЗВІТ
З ПРАКТИЧНОЇ РОБОТИ №10

дисципліна: «Алгоритми та структур и даних»

Виконала: студентка групи КС-22
Узенкова Дар’я
Перевірив: Олешко Олег

Харків
2024

Завдання. Дано орієнтований незважений граф. Необхідно визначити, чи є в ньому цикли, і якщо є, то вивести будь-який з них.

Вводимо два натуральні числа — кількість вершин і ребер у графі відповідно.

Далі вводимо ребра графа. Кожне задається парою чисел – номерами початкової та кінцевої вершин відповідно.

Якщо у графі немає циклу, то вивести NO, інакше вивести YES і потім перерахувати вершини в порядку обходу циклу.

У процесі обходу графа виводити кольори вершин за кроками:

- білий - ми ще не відвідували вершину
- сірий - відвідали вершину і не вийшли з неї (зациклилися)
- чорний - відвідали вершину і вийшли з неї

Результати виконання завдання 1 наведено:

1. У лістингу 1 – вихідний код програми.
2. У малюнках 1, 2, 3 – результати виконання програми.

```c
#include <stdio.h>
#include <stdlib.h>

#define WHITE 0 // Вершина ще не відвідана
#define GRAY 1  // Вершина в обробці
#define BLACK 2 // Вершина оброблена

int *color;    // Масив кольорів для вершин
int *parent;   // Масив батьків для відстеження циклу
int cycle_start, cycle_end; // Змінні для зберігання початку та кінця циклу
int found_cycle = 0;        // Прапорець для перевірки, чи був знайдений хоча б
один цикл

// Функція додавання ребра в граф
void add_edge(int **graph, int u, int v) {
    graph[u][v] = 1;
}

// Функція для виведення стану всіх вершин
void print_vertex_colors(int n) {
    printf("Стан вершин: ");
    for (int i = 0; i < n; i++) {
        if (color[i] == WHITE) {
            printf("Вершина %d: біла, ", i + 1);
        } else if (color[i] == GRAY) {
            printf("Вершина %d: сіра, ", i + 1);
        } else if (color[i] == BLACK) {
```

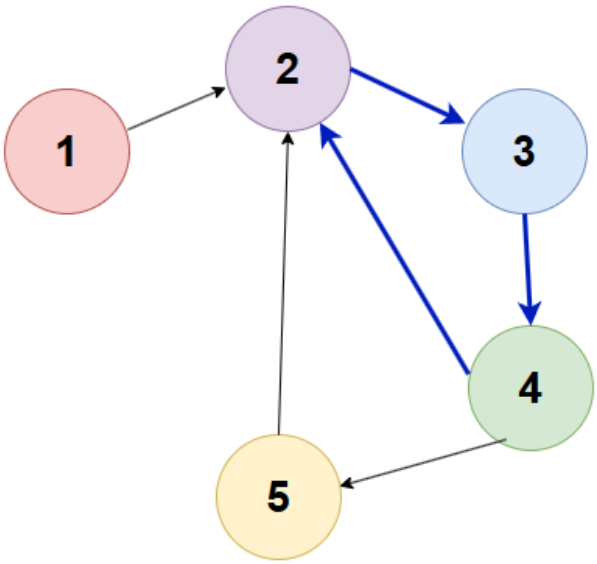
```
            printf("Вершина %d: чорна, ", i + 1);
        }
    }
    printf("\n");
}

// Функція обходу в глибину (DFS)
int dfs(int v, int **graph, int n) {
    color[v] = GRAY;
    printf("\nВершина %d стала сірою\n", v + 1);
    print_vertex_colors(n); // Виводимо стан всіх вершин

    for (int u = 0; u < n; u++) {
        if (graph[v][u]) { // Якщо існує ребро з v до u
            if (color[u] == WHITE) {
                parent[u] = v;
                if (dfs(u, graph, n)) {
                    return 1;
                }
            } else if (color[u] == GRAY) {
                // Знайдений цикл
                cycle_start = u;
                cycle_end = v;
                found_cycle = 1;
                return 1;
            }
        }
    }

    color[v] = BLACK;
    printf("\nВершина %d стала чорною\n", v + 1);
    print_vertex_colors(n); // Виводимо стан всіх вершин
    return 0;
}

// Функція для пошуку першого циклу в графі
void find_first_cycle(int **graph, int n) {
    color = (int *)malloc(n * sizeof(int));
    parent = (int *)malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        color[i] = WHITE;
        parent[i] = -1;
    }

    for (int v = 0; v < n; v++) {
        if (color[v] == WHITE && !found_cycle) {
            if (dfs(v, graph, n)) {
                // Якщо знайдено цикл, виводимо його та завершуємо пошук
                printf("\nYES\n");
                int cycle_vertices[100];
```

```
                int cycle_length = 0;
                int u = cycle_end;
                printf("Знайдений цикл: ");
                while (u != cycle_start) {
                    cycle_vertices[cycle_length++] = u + 1;
                    u = parent[u];
                }
                cycle_vertices[cycle_length++] = cycle_start + 1;
                for (int i = cycle_length - 1; i >= 0; i--) {
                    printf("%d ", cycle_vertices[i]);
                }
                printf("%d\n", cycle_vertices[cycle_length - 1]); // Додаємо
початкову вершину в кінці для завершення циклу
                break; // Виходимо після першого знайденого циклу
            }
        }
    }

    if (!found_cycle) {
        printf("\nNO\n");
    }

    free(color);
    free(parent);
}

int main() {
    int n, m;
    printf("Введіть кількість вершин і ребер: ");
    scanf("%d %d", &n, &m);

    int **graph = (int **)malloc(n * sizeof(int *));
    for (int i = 0; i < n; i++) {
        graph[i] = (int *)calloc(n, sizeof(int));
    }

    printf("Введіть ребра (початок кінець):\n");
    for (int i = 0; i < m; i++) {
        int u, v;
        scanf("%d %d", &u, &v);
        if (u > 0 && u <= n && v > 0 && v <= n) {
            add_edge(graph, u - 1, v - 1);
        } else {
            printf("Невірні індекси ребер!\n");
            i--;
        }
    }

    find_first_cycle(graph, n);

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

```
        free(graph[i]);
    }
    free(graph);

    return 0;
}
```

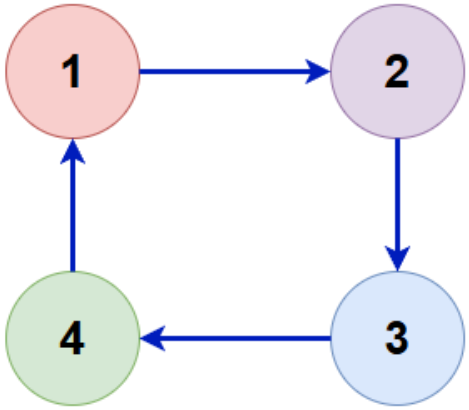
Лістинг 1 – вихідний код програми

```
Введіть кількість вершин і ребер: 5 6
Введіть ребра (початок кінець):
1 2
2 3
3 4
4 2
4 5
5 2

Вершина 1 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: біла, Вершина 3: біла, Вершина 4: біла, Вершина 5: біла,

Вершина 2 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: біла, Вершина 4: біла, Вершина 5: біла,

Вершина 3 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: сіра, Вершина 4: біла, Вершина 5: біла,

Вершина 4 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: сіра, Вершина 4: сіра, Вершина 5: біла,

YES
Знайдений цикл: 2 3 4 2
PS C:\Users\uzenk>
```



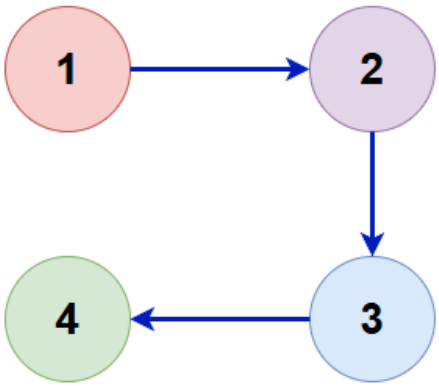
Малюнок 1 – результат виконання програми

```
Введіть кількість вершин і ребер: 4 4
Введіть ребра (початок кінець):
1 2
2 3
3 4
4 1

Вершина 1 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: біла, Вершина 3: біла, Вершина 4: біла,

Вершина 2 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: біла, Вершина 4: біла,

Вершина 3 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: сіра, Вершина 4: біла,

Вершина 4 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: сіра, Вершина 4: сіра,

YES
Знайдений цикл: 1 2 3 4 1
PS C:\Users\uzenk>
```



Малюнок 2 – результат виконання програми

```
Введіть кількість вершин і ребер: 4 3
Введіть ребра (початок кінець):
1 2
2 3
3 4

Вершина 1 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: біла, Вершина 3: біла, Вершина 4: біла,

Вершина 2 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: біла, Вершина 4: біла,

Вершина 3 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: сіра, Вершина 4: біла,

Вершина 4 стала сірою
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: сіра, Вершина 4: сіра,

Вершина 4 стала чорною
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: сіра, Вершина 4: чорна,

Вершина 3 стала чорною
Стан вершин: Вершина 1: сіра, Вершина 2: сіра, Вершина 3: чорна, Вершина 4: чорна,

Вершина 2 стала чорною
Стан вершин: Вершина 1: сіра, Вершина 2: чорна, Вершина 3: чорна, Вершина 4: чорна,

Вершина 1 стала чорною
Стан вершин: Вершина 1: чорна, Вершина 2: чорна, Вершина 3: чорна, Вершина 4: чорна,

NO
PS C:\Users\uzenk>
```



Малюнок 3 – результат виконання програми