Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп’ютерних наук та штучного інтелекту

ЗВІТ

З ПРАКТИЧНОЇ РОБОТИ №10

дисципліна: «Алгоритмізація та програмування»

Виконав: студент 2 курсу групи КС22
Спеціальності 122 «Комп’ютерні науки»
Скрипняк Тарас Артемович
Прийняв: викладач
Олешко О.І.

Харків – 2024

Завдання №1: 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

```
#include <stdio.h>

#define MAX_NODES 1000
#define MAX_EDGES 2000
#define WHITE 0
#define GRAY 1
#define BLACK 2

int adjacency_list[MAX_NODES][MAX_NODES];
int adjacency_count[MAX_NODES];
int color[MAX_NODES];
int parent[MAX_NODES];
int stack[MAX_NODES];
int stack_size;
int n, m;
int cycle_start = -1, cycle_end = -1;

void push(int v) {
    stack[stack_size++] = v;
}

int pop() {
    return stack[--stack_size];
}

void add_edge(int u, int v) {
    adjacency_list[u][adjacency_count[u]++] = v;
}

int dfs_iterative(int start) {
    stack_size = 0;
    push(start);

    while (stack_size > 0) {
        int v = pop();
        if (color[v] == WHITE) {
            color[v] = GRAY;
            printf("Вершина %d: сірий\n", v);
            push(v);

            for (int i = 0; i < adjacency_count[v]; i++) {
                int u = adjacency_list[v][i];
                if (color[u] == WHITE) {
```

```
                    parent[u] = v;
                    push(u);
                } else if (color[u] == GRAY) {
                    cycle_start = u;
                    cycle_end = v;
                    return 1;
                }
            }
        } else if (color[v] == GRAY) {
            color[v] = BLACK;
            printf("Вершина %d: чорний\n", v);
        }
    }
    return 0;
}

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

    printf("Вершини спочатку: білі\n");

    for (int i = 0; i < n; i++) {
        if (color[i] == WHITE) {
            if (dfs_iterative(i)) break;
        }
    }

    if (cycle_start == -1) {
        printf("NO\n");
    } else {
        printf("YES\n");
        int temp = cycle_end;
        printf("%d ", cycle_start);
        while (temp != cycle_start) {
            printf("%d ", temp);
            temp = parent[temp];
        }
        printf("%d\n", cycle_start);
    }
}

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


    for (int i = 0; i < n; i++) {
        adjacency_count[i] = 0;
    }

    printf("Введіть ребра (початкова вершина і кінцева вершина):\n");
    for (int i = 0; i < m; i++) {
        int u, v;
        scanf("%d %d", &u, &v);
        add_edge(u, v);
    }

    find_cycle();

    return 0;
}
```

Лістинг - вихідний код програми

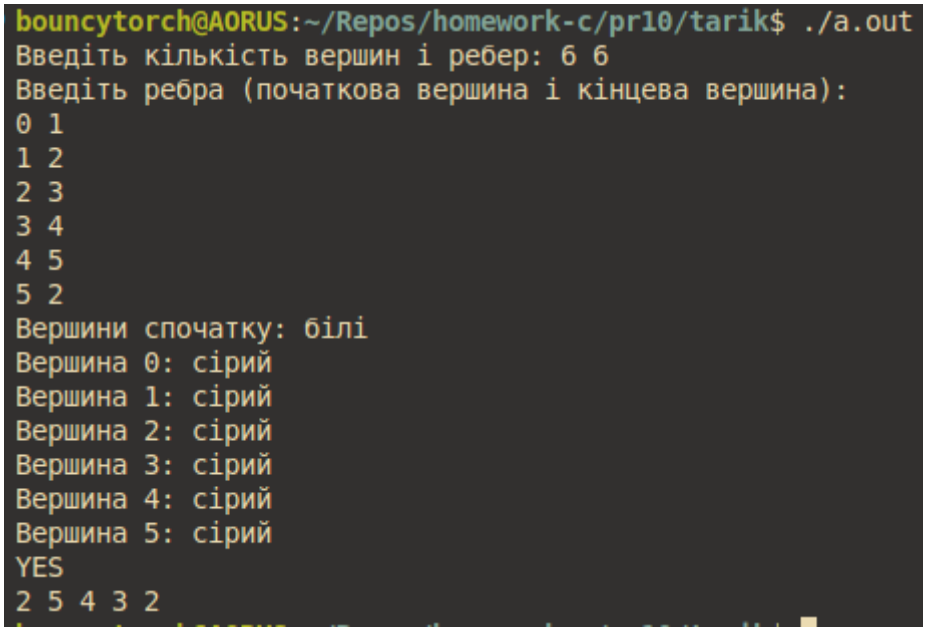

Рисунок 1 - результат виконання програми