Obraz zawierający tekst

Opis wygenerowany automatycznie

**Dariusz Strojny**

Algorytmy i Struktury Danych

Projekt zaliczeniowy nr 3

Rzeszów, 2023

# Spis treści

[1 Spis treści 3](#_Toc125372620)

[2 Spis ilustracji 4](#_Toc125372621)

[3 Temat 5](#_Toc125372622)

[4 Analiza, projektowanie 6](#_Toc125372623)

[4.1 Zasada działania programu 6](#_Toc125372624)

[4.2 Struktury danych 6](#_Toc125372625)

[4.3 Metodyka 7](#_Toc125372626)

[4.3.1 Funkcje użyte w programie 7](#_Toc125372627)

# Spis ilustracji

**Nie można odnaleźć pozycji dla spisu ilustracji.**

# Temat

**Zadanie 9.** Napisz program, który dla zadanego grafu skierowanego reprezentowanego przy pomocy macierzy sąsiedztwa wyznaczy i wypisze następujące informacje:

1. Wszystkich sąsiadów dla każdego wierzchołka grafu (sąsiad wierzchołka to ten wierzchołek, do którego prowadzi krawędź z
2. Wszystkie wierzchołki, które są sąsiadami każdego wierzchołka
3. Stopnie wychodzące wszystkich wierzchołków
4. Stopnie wchodzące wszystkich wierzchołków
5. Wszystkie wierzchołki izolowane
6. Wszystkie pętle
7. Wszystkie krawędzie dwukierunkowe

Każdy z powyższych podpunktów powinien być realizowany jako oddzielna funkcja. W funkcji main() należy przedstawić działanie napisanej przez siebie biblioteki na reprezentatywnym przykładzie. Kod powinien być opatrzony stosownymi komentarzami.

# Analiza, projektowanie

## Zasada działania programu

Program ten ma za zadanie wykonywanie zbioru określonych akcji na grafie skierowanym reprezentowanym przy pomocy macierzy sąsiedztwa. Macierz ta ma zostać wypełniona wartościami określającymi istnienie krawędzi pomiędzy poszczególnymi wierzchołkami tego grafu.

## Struktury danych

Najważniejszą strukturą danych w programie jest macierz sąsiedztwa reprezentująca wprowadzany do niej graf skierowany. Macierz sąsiedztwa posiada wymiary n x n, gdzie n jest ilością wierzchołków w danym grafie. Każde pole w macierzy sąsiedztwa reprezentuje informację o istnieniu pewnej krawędzi poprzez wartość 1 lub jej brak przez wartość 0. Pola znajdujące się na przekątnej reprezentują istnienie krawędzi tworzących pętle na poszczególnych wierzchołkach. Pozostałe pola definiują krawędzie między wierzchołkami, których numery są odpowiednio zależne od numeru wiersza i kolumny, na których to te pola się znajdują.

Na przykład: Pole o numerze wiersza 3 i kolumny 2 w macierzy sąsiedztwa, reprezentuje istnienie krawędzi skierowanej z wierzchołka 3 do wierzchołka 2, natomiast pole o numerze wiersza 1 i kolumny 1 definiuje istnienie pętli na tym wierzchołku.

Dodatkowo utworzoną przeze mnie strukturą danych jest struktura grafu, przechowująca w kodzie tablicę będącą macierzą sąsiedztwa oraz ilość wierzchołków w grafie. Jej definicja wygląda następująco:

// struktura grafu przechowująca macierz sąsiedztwa oraz ilość wierzchołków grafu

struct {

// tablica dwuwymiarowa przechowująca informacje o krawędziach grafu

int\*\* array;

// ilość wierzchołków grafu

int size;

} typedef graph;

## Metodyka

### Funkcje użyte w programie

• graph **new\_graph**(int size)

Tworzy nową strukturę typu graph i inicjuje w niej pamięć dla macierzy sąsiedztwa

Parametry

*size* – ilość wierzchołków w tworzonym grafie

Zwraca

Zainicjowany obiekt typu graph

• void **free\_graph**(graph &g)

Zwalnia pamięć struktury typu graph

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_add\_edge**(graph& g, int a, int b)

Ustawia w macierzy sąsiedztwa grafu skierowanego informację o obecności krawędzi od wierzchołka a do wierzchołka b

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

*a* – numer wierzchołka, z którego wychodzi krawędź

*b* – numer wierzchołka, do którego wchodzi krawędź

• void **graph\_add\_edge**(graph& g, const std::vector<std::pair<int, int>>& edges)

Ustawia w macierzy sąsiedztwa grafu skierowanego informację o obecności krawędzi pomiędzy każdą z par przekazanych w tablicy

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

*edges* – tablica zawierająca pary numerów wierzchołków, między którymi mają powstać krawędzie

• void **print\_graph**(const graph& g)

Wypisuje zawartość macierzy sąsiedztwa grafu

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_print\_neighbours\_of\_each\_vertex**(const graph& g)

Wypisuje wszystkich sąsiadów dla każdego wierzchołka grafu

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_print\_each\_vertex\_that\_is\_neighbour**(const graph& g)

Wypisuje wszystkie wierzchołki, które są sąsiadami każdego wierzchołkaParametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_print\_vertex\_outdegrees**(const graph& g)

Wypisuje stopnie wychodzące wszystkich wierzchołków

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_print\_vertex\_indegrees**(const graph& g)

Wypisuje stopnie wchodzące wszystkich wierzchołków

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_print\_isolated\_vertexes**(const graph& g);

Wypisuje wszystkie wierzchołki izolowane

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_print\_loops**(const graph& g)

Wypisuje wszystkie pętle

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego

• void **graph\_print\_multi\_edges**(const graph& g)

Wypisuje wszystkie krawędzie dwukierunkowe

Parametry

*g* – obiekt typu graph przechowujący macierz sąsiedztwa i ilość wierzchołków grafu skierowanego