AKADEMIA GÓRNICZO-HUTNICZA im. Stanisława Staszica w Krakowie

Algorytmy grafowe – reprezentacja, algorytmy przeszukiwania

Stanisław Olech - 412023

Automatyka i Robotyka

EAIiIB

Zad. 1

Macierz sąsiedztwa

+ proste sprawdzenie istnienia krawędzi

+ prosta implementacja wag do grafu

- duża złożoność obliczeniowa

- nieopłacalna dla macierzy rzadkiej

Listy sąsiedztwa
+ mała złożoność pamięciowa

+ dobra dla macierzy rzadkich

Zad. 2

Algorytm szukania wszerz.

Graf nieskierowany będzie niespójny gdy przy przejściu całego grafu nie odwiedzimy któregoś z wierzchołków.

Graf nie będzie mieć cyklu jeśli nigdy nie będziemy mieli drogi z dowolnego wierzchołka z powrotem do niego samego. Jest to łatwe w implementacji bo możemy dodać tylko zmianę na true gdy będzie powtórzenie.

```
#include <iostream>
#include <vector>
#include <deque>
#include <algorithm>
#include <tuple>

std::tuple <std::vector<int>, std::vector<int>, std::vector<bool>>
BFS(const std::vector<int>* graf, int s, int size){
    // 1. Inicjalizacja zmiennych
    std::deque<int> FIFO;
    int v;
    int no;
    std::vector<int> NO;
    std::vector<int> old;
    std::vector<bool> property = {false, false};  // spójny/niespójny,
acykliczny/z cyklami

    // 2. Nadanie wierzchołkowi v=s numeru No=1
    no = 1;
    v = s;
    old.push_back(v);
    NO.push_back(no);
```

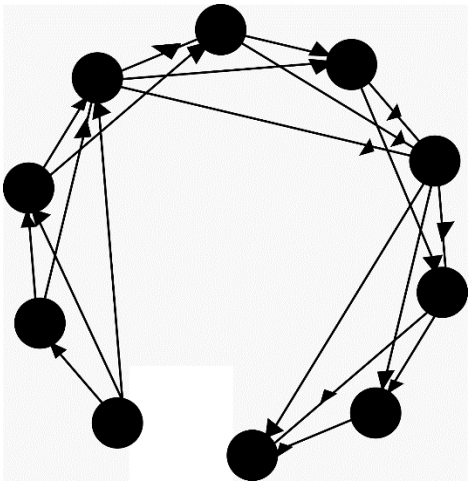
```
    // 3. Umieszczenie w FIFO sąsiadów v
    for (auto x: graf[v]){
        FIFO.push_back(x);
    }

    // 4. Dopóki FIFO nie jest pusta
    while (!FIFO.empty()){
        // 4a pobranie z FIFO wierzchołka v (z usunięciem)
        v = FIFO.front();
        FIFO.pop_front();

        // 4b Nadanie kolejnego numeru ++No
        no++;
        NO.push_back(no);
        old.push_back(v);

        // 4c Dodanie nieponumerowanych sąsiadów v do FIFO
        for (auto x: graf[v]){
            if (!std::count(old.begin(), old.end(), x)){
                // eliminacja powtórzeń w kolejce.
                if (!std::count(FIFO.begin(), FIFO.end(), x)){
                    FIFO.push_back(x);
                }
            }
            else{
                property[1] = true;
            }
        }
    }

    // 5. Analiza No[] - Wyjście: numeracja i własności grafu
    if (size != no++){property[0] = true;}
    return {NO, old, property};
}

int main() {

    // lista sąsiedztwa
    int nodes = 5;
    std::vector<int> graph[nodes]; // Tablica std::vectorów
    graph[0].push_back(1);
    graph[0].push_back(4);
    graph[1].push_back(0);
    graph[1].push_back(2);
    graph[1].push_back(4);
    graph[2].push_back(1);
    graph[2].push_back(3);
    graph[3].push_back(2);
    graph[3].push_back(4);
    graph[4].push_back(0);
    graph[4].push_back(1);
    graph[4].push_back(3);

    auto ans = (BFS(graph, 1, nodes));
    std::vector<int> path = std::get<0>(ans);
    auto property = std::get<2>(ans);

    for (auto x: path){
        std::cout << x << std::endl;
    }
```

```
    std::cout << std::boolalpha << std::endl;
    std::cout << "jest niespojny: " << property[0] << std::endl;
    std::cout << "ma cykl: " << property[1] << std::endl;
    return 0;
}
```

Kod. 1 kod źródłowy algorytmu szukania wszerz.

Zad 3.

a) Graf spójny acykliczny

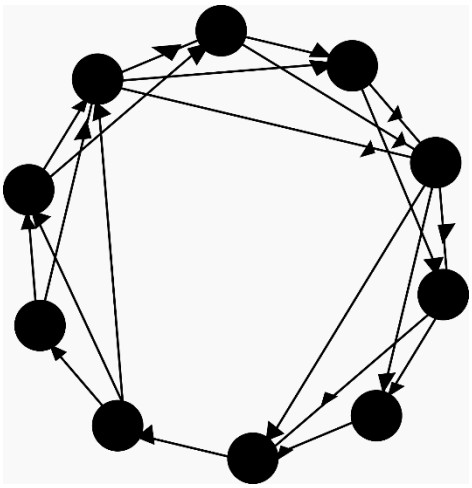


Rys. 1 reprezentacja grafu spójnego acyklicznego

```
std::vector<int>
graph1[nodes]; // Tablica
std::vectorów
graph1[0].push_back(1);
graph1[0].push_back(2);
graph1[0].push_back(3);
graph1[1].push_back(2);
graph1[1].push_back(3);
graph1[2].push_back(3);
graph1[2].push_back(4);
graph1[3].push_back(4);
graph1[3].push_back(5);
graph1[3].push_back(6);
graph1[4].push_back(5);
graph1[4].push_back(6);
graph1[5].push_back(6);
graph1[5].push_back(7);
graph1[6].push_back(7);
graph1[6].push_back(8);
graph1[6].push_back(9);
graph1[7].push_back(8);
graph1[7].push_back(9);
graph1[8].push_back(9);
```

kod. 2 reprezentacja moim sposobem w c ++

b) Graf spójny z cyklami

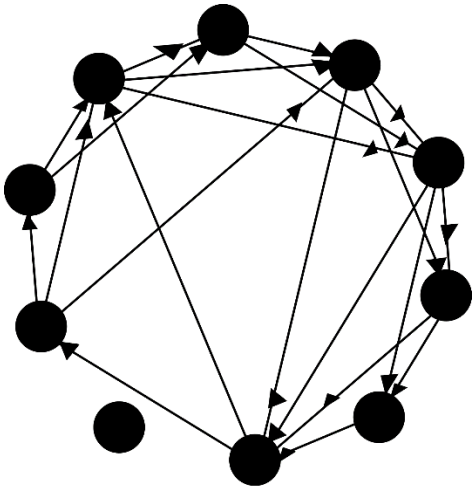


Rys. 2 reprezentacja grafu spójnego z cyklami

```
std::vector<int>
graph1[nodes]; // Tablica
std::vectorów
graph1[0].push_back(1);
graph1[0].push_back(2);
graph1[0].push_back(3);
graph1[1].push_back(2);
graph1[1].push_back(3);
graph1[2].push_back(3);
graph1[2].push_back(4);
graph1[3].push_back(4);
graph1[3].push_back(5);
graph1[3].push_back(6);
graph1[4].push_back(5);
graph1[4].push_back(6);
graph1[5].push_back(6);
graph1[5].push_back(7);
graph1[6].push_back(7);
graph1[6].push_back(8);
graph1[6].push_back(9);
graph1[7].push_back(8);
graph1[7].push_back(9);
graph1[8].push_back(9);
graph1[9].push_back(0);
```

kod. 3 reprezentacja moim sposobem w c ++

c) Graf niespójny z cyklami



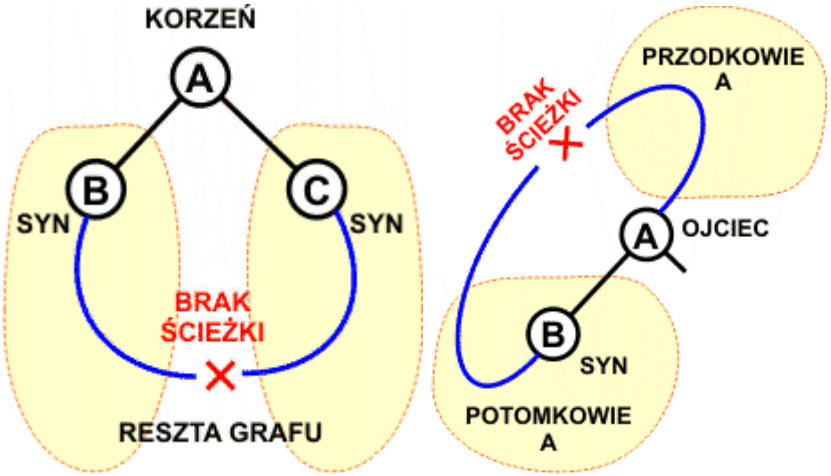
Rys. 3 reprezentacja grafu niespójnego z cyklami

```
 std::vector<int>
graph1[nodes]; // Tablica
std::vectorów
graph1[1].push_back(2);
graph1[1].push_back(3);
graph1[2].push_back(3);
graph1[2].push_back(4);
graph1[3].push_back(4);
graph1[3].push_back(5);
graph1[3].push_back(6);
graph1[4].push_back(5);
graph1[4].push_back(6);
graph1[5].push_back(6);
graph1[5].push_back(7);
graph1[5].push_back(8);
graph1[5].push_back(9);
graph1[6].push_back(7);
graph1[6].push_back(8);
graph1[6].push_back(9);
graph1[7].push_back(8);
graph1[7].push_back(9);
graph1[8].push_back(9);
graph1[9].push_back(1);
```

kod. 4 reprezentacja moim sposobem w c ++

Zad. 4

Moim zdaniem najprostszy sposób na znalezienie Wierzchołka rozpajającego graf to napisanie programu znajdującego liczbę spójnych składowych grafu a następnie po kolei usuwa każdy z wierzchołków. – nie jest to najlepsze rozwiązanie. Odpowiedz, którą znalazłem na Internecie to modyfikacja DFS by sprawdzać między gałęziami oraz między gałęziami a przodkami czy są jakieś połączenia- jak nie ma to jest to taka krawędź.



Rys. 4 Wierzchołek rozpajający graf. Źródło: https://eduinf.waw.pl

Aby znaleźć centum grafu stosuje się algorytm Dijkstry.

Można badać parametry takie jak na przykład czy graf jest:

- Pusty
- Regulowany
- Eulerowskim

Można też badać:

- stopień grafu (stopnie wierzchołków też)
- obwód
- indeks chromatyczny (zmienne mówiące o użytych kolorach)

Wnioski.

Laboratorium pozwoliło mi przypomnieć sobie podstawowe pojęcia związane z grafami. Umożliwiło mi też przypomnienie sobie podstaw c++.