Algorytmy grafowe – reprezentacja, algorytmy przeszukiwania

Zadanie 1:

**Macierz sąsiedztwa:**

Zalety:

* Sprawdzenie czy istnieje krawędź oraz sprawdzenie kosztu danej krawędzi ma złożoność czasową O(1)
* Złożoność usunięcia krawędzi O(1)
* Szybki sposób sprawdzenia czy dany graf jest skierowany
* Jeżeli graf jest symetryczny można używać macierzy trójkątnej górnej (nie ma potrzeby zapisu dwukrotnie tej samej krawędzi)

Wady:

* Złożoność pamięciowa O(|V|2) gdzie |V| to moc zbioru wierzchołków

**Lista sąsiedztwa:**

Zalety:

* Złożoność czasowa dodania krawędzi O(1)
* Znacznie mniejsza złożoność pamięciowa O(|V|+|E|) gdzie |V| to moc zbioru wierzchołków a |E| to moc zbioru krawędzi

Wady:

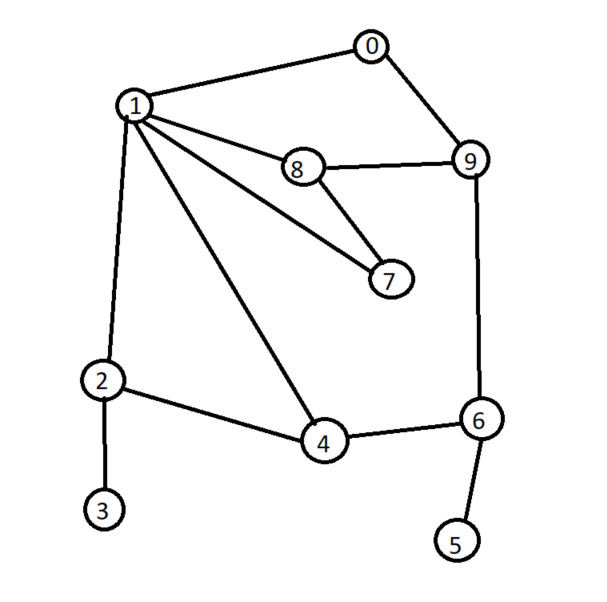
* Większa złożoność sprawdzenia czy istnieje lub usunięcie krawędzi O(|E|) gdzie |E| to moc zbioru krawędzi
* Przy grafach nieskierowanych potrzeba zapisu krawędzi w dwóch miejscach.

Podsumowując zapis macierzowy grafu jest opłacalny w przypadku bardzo gęstych grafów. Jeżeli chodzi o grafy rzadkie o dużo lepszym wyborem zapisu byłaby macierz sąsiedztwa. Z uwagi na pewne ograniczenia sprzętowe pod względem pamięciowym do zapisu grafów można posłużyć się zapisem przewidzianych dla macierzy rzadkich takich jak: Lista list (LIL), Współrzędne i wartość (COO), Format Yale.

Zadanie 2

Zdefiniowany graf:

Obraz zawierający tekst

Opis wygenerowany automatycznie 

**Rys. 1.** Reprezentacja grafu za pomocą list sąsiedztwa **Rys. 2.** Rysunek przykładowego grafu

Zadanie 3

Algorytm przeszukiwania wszerz:

Warunek na spójność grafu:

* Za pomocą BFS zostały odwiedzone wszystkie wierzchołki.

Warunek na acykliczność grafu:

Jeżeli graf jest spójny:

* Należy badać czy w trakcie fazy 4c (Dodanie nieponumerowanych sąsiadów v do FIFO) algorytmu BFS natknięto się na ponumerowany wierzchołek. Jeżeli tak to czy jest on poprzednikiem przerabianego aktualnie wierzchołka jeżeli tak to należy kontynuować sprawdzanie acykliczności. Jeżeli nie to w danym grafie istnieje cykl

Jeżeli graf jest niespójny i nie wykryto cyklu w pierwszym kawałku grafu(jeżeli znaleziono to automatycznie dany graf nie jest acykliczny):

* Należy wykonać BFS z odpowiednią flagą acykl\_check = True dla wszystkich nieodwiedzonych wierzchołków i sprawdzić czy istnieją jakieś cykle jak dla grafów spójnych. Dodatkowo można bardziej zoptymalizować dany kod pod względem współdzielenia pewnej listy z rejestrem odwiedzonych wierzchołków dla różnych instancji wywoływanego algorytmu BFS z flagą acykl\_check = True.

Kod źródłowy:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Rys. 3.** Pierwsza część programu[[1]](#footnote-1)

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Rys. 4.** Druga część programu

Wynik dla przykładowego grafu (**Rys. 2.**). kolejne wartości z listy określają kolejne odwiedzone wierzchołki:



**Rys. 5.** Wynik operacji BFS dla przykładowego grafu

Zadanie 4

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Rys. 6.** Reprezentacja grafów dla różnych przypadków

Graf acykliczny spójny:

Obraz zawierający zegarek

Opis wygenerowany automatycznie

**Rys. 7.** Graf acykliczny spójny

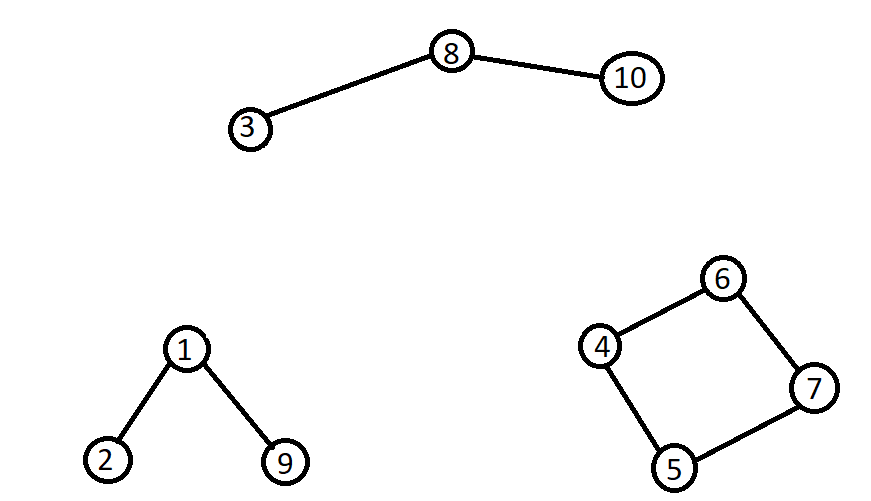
Graf spójny zawierający cykle:

Obraz zawierający zegarek

Opis wygenerowany automatycznie

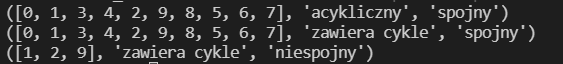
**Rys. 7.** Graf spójny zawierający cykle

Graf nie spójny zawierający cykle:



**Rys. 8.** Graf nie spójny zawierający cykle

Kolejno wyniki dla danych grafów(**Rys. 6.** ,**Rys. 7.** ,**Rys. 8.**):



**Rys. 9.** Wyniki algorytmu dla poszczególnych grafów.

Zadanie 5

* Wierzchołki rozpajające grafu możemy znaleźć w następujący sposób:

dla każdego wierzchołka v z V -zbiór wierzchołków grafu G wykonujemy bfs lub dfs lecz pomijamy istnienie wierzchołka v. (tzn. pomijamy każdą krawędź incydentną z wierzchołkiem v) Następnie sprawdzamy czy wszystkie wierzchołki (bez v) zostały odwiedzone. Jeżeli tak się nie stało to wierzchołek v jest wierzchołkiem rozpajającym.

* Centrum grafu możemy znaleźć w następujący sposób:

dla każdego wierzchołka v z V -zbiór wierzchołków grafu G wykonujemy bfs przy czym każda wartość w kolejce będzie miała dodatkową wartość która jest określana odległość od grafu początkowego. Gdy w bfs będzie rozpatrywany nowy wierzchołek z kolejki to wszystkie nowo dodane wierzchołki będą posiadały odległość (w kolejce) o 1 większą od rozpatrywanego wierzchołka. Przy układaniu listy odwiedzonych wierzchołków przypisujemy odległość danego wierzchołka do maksymalnego kosztu jeżeli ten jest większy od poprzedniego maksymalnego kosztu. Porównujemy wszystkie maksymalne koszty dla wierzchołków i wybieramy wierzchołek dla którego koszt wynosi minimum. Ten wierzchołek jest centrum grafu

* Za pomocą dfs można znajdować drogę pomiędzy dwoma wierzchołkami odpowiednio wprowadzając zapamiętywanie drogi
* Za pomocą dfs i bfs można sprawdzić czy graf zawiera cykle lub czy jest spójny.(zadanie 3)
* Za pomocą dfs (dla grafu nieważonego) można stworzyć minimalne drzewo rozpinające odpowiednio dodając odwiedzone krawędzie pomiędzy poprzednikiem a rozpatrywanym wierzchołkiem do pewnego zbioru.

1. Ucięty komentarz „nadanie wierzchołkowi v=s numeru No = s (numeracja) oraz dodanie poprzednika (potrzebne przy wyznaczaniu cykli)” [↑](#footnote-ref-1)