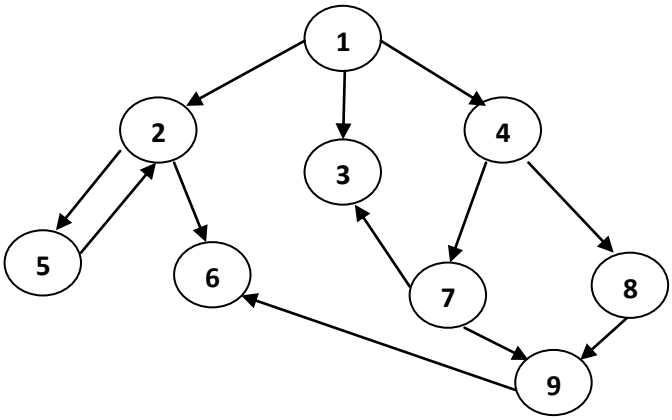
Algorytmy i złożoność obliczeniowa

Laboratorium 6: Grafy. Macierzowa reprezentacja grafu. Algorytm BFS

1. **Przeszukiwanie grafu wszerz (Breadth-First Search, BFS).**

   Przeszukiwanie grafu polega na odwiedzaniu jego wierzchołków zgodnie z relacją sąsiedztwa. W zależności od wybranego algorytmu kolejność odwiedzania wierzchołków może być różna. Owa kolejność w przypadku przeszukiwania wszerz (BFS) jest taka, że po odwiedzeniu danego wierzchołka przechodzimy do **wszystkich** jego sąsiadów, a dopiero potem do innych wierzchołków (sąsiadów owych odwiedzonych sąsiadów). Aby zapewnić taką kolejność przetwarzania algorytm korzysta z kolejki FIFO przechowującej wierzchołki. Poniższy rysunek przedstawia przykładowy graf, którego wierzchołki zostały ponumerowane zgodnie z kolejnością odwiedzania przez algorytm BFS, przyjmując że sąsiadów danego wierzchołka przeglądamy kolejno od lewej do prawej.



2. **Algorytm BFS, wersja podstawowa.**

   Algorytm przeszukiwania grafu wszerz rozpoczyna działanie od wyznaczonego wierzchołka s. Wierzchołek ten jest oznaczany jako odwiedzony, żeby zapobiec powtórnemu przetwarzaniu tego wierzchołka przez algorytm, a następnie dodawany do listy Q. Na tej liście znajdują się wierzchołki, które będą przetwarzane w kolejnych krokach algorytmu.

   W danym kroku algorytmu pobierany jest pierwszy wierzchołek z listy Q, oznaczony przez v. Następnie odwiedzane są wszystkie wierzchołki sąsiadujące z wybranym wierzchołkiem v, które nie są jeszcze oznaczone jako odwiedzone (odwiedzenie wierzchołka polega na oznaczeniu go jako odwiedzonego oraz dodaniu go na koniec listy Q). Następnie wierzchołek v zostaje oznaczony jako przeszukany (gdy wszyscy sąsiedzi v są już odwiedzeni. Algorytm kończy działanie, gdy lista Q jest pusta.

3. **Klasy potrzebne do rozwiązania zadań.**

   Dana jest klasa abstrakcyjna MGraph oraz interfejs IBfsSearchable, które należy pobrać bezpośrednio ze strony WWW. Do przechowywania wierzchołków można użyć własnej implementacji listy lub skorzystać z klas Java Collections Framework, np. interfejsu Queue i dowolnej jego implementacji, np. LinkedList. Poniższy fragment kodu pokazuje przykładowe wykorzystanie kolejki.

```
Queue<Integer> Q = new LinkedList<>(); //utworzenie pustej kolejki
Q.offer(1)  //wstawienie 1 na koniec kolejki (aut. konwersja na Integer)
int v = Q.poll(); // pobranie 1. el. z kolejki i aut. konwersja z Integer na int
if (Q.isEmpty()) { ... } //sprawdzenie czy kolejka jest pusta
```

4. Zadania.

1. Utwórz nowy projekt Java w środowisku NetBeans lub Eclipse. Utwórz klasę **TGraph** dziedziczącą z klasy **aiz.graph.MGraph**. Zaimplementuj dziedziczone metody abstrakcyjne. Napisz krótki program, w którym definiowany jest graf zawierający minimum 7 wierzchołków i 12 krawędzi. [2p]

2. Utwórz klasę **BFSGraph** dziedziczącą z klasy **TGraph** implementującą interfejs **aiz.graph.IBfsSearchable**. Zaimplementuj metodę **bfs** realizującą przeszukiwanie grafu wszerz tak, aby podczas przeszukiwania zostały wypisane wierzchołki grafu zgodnie z kolejnością ich odwiedzenia. [3p]

3. Utwórz klasę **BFSGraph2** dziedziczącą z klasy **BFSGraph**. Nadpisz implementację metody **bfs** tak, aby po przeszukaniu grafu zostały wypisane odległości od wierzchołka źródłowego do pozostałych wierzchołków w grafie. [2p]

4. Zaimplementuj wyszukiwanie najkrótszej drogi w grafie z wybranego wierzchołka źródłowego do wybranego wierzchołka docelowego z wykorzystaniem algorytmu BFS. [2p]
    a. Wypisz odległość i drogę z wierzchołka źródłowego do docelowego (kolejne sąsiadujące ze sobą wierzchołki) [1p]

Praca domowa:

1. Zaimplementować metodę, która korzystając z algorytmu BFS zwróci graf wejściowy obcięty do drzewa przeszukiwań. Wynikiem ma być zatem graf wejściowy, z którego usunięto wszystkie wierzchołki nieosiągalne z wierzchołka źródłowego i pozostawiono tylko krawędzie umożliwiające dotarcie do wszystkich wierzchołków najkrótszą drogą.
2. Zaproponować struktury danych dla listowej reprezentacji grafu ważonego.
3. Zaimplementować algorytm DFS używając stosu zamiast kolejki.

Na następnych zajęciach:
Grafy. Algorytm DFS. Silnie spójne składowe.