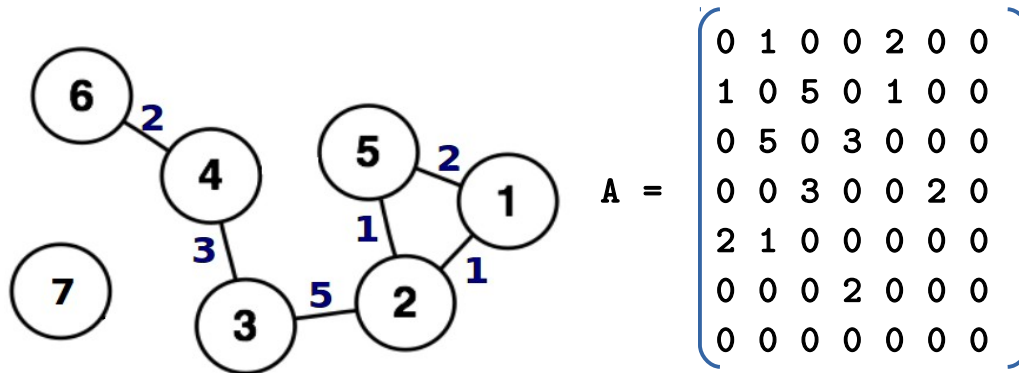


Języki Programowania, Zadanie dodatkowe

Wyszukiwanie ścieżki w grafie. Macierz sąsiedztwa. Algorytm Dijkstry. Czytanie pseudokodu.

W teorii grafów **macierz sąsiedztwa** grafu G jest kwadratową macierzą w której a_{ij} oznacza wagę krawędzi pomiędzy wierzchołkami i i j . Dla przykładu – graf z 7 wierzchołkami:



1. Należy stworzyć klasę graf używając dwuwymiarowej tablicy (macierz sąsiedztwa) do przechowywania informacji o jej krawędziach. Klasa powinna mieć następujące metody:

`Graf(int max)` – konstruktor ; `max` – maksymalna ilość wierzchołków w danym grafie. W przypadku osiągnięcia tej liczby dodanie nowego wierzchołka nie będzie możliwe.

`void init()` - zeruje wszystkie pola macierzy sąsiedztwa

`void add()` - dodanie wierzchołka

`void remove(int r)` - usunięcie wierzchołka o numerze `r`

`void display()` - wyświetlenie grafu (macierzy sąsiedztwa)

`void add_edge(int n, int m, int w=1)` - dodanie krawędzi pomiędzy wierzchołkami o numerach `m` i `n` (oraz wagą `w`)

`void remove_edge()` - usunięcie krawędzi (dwa numery wierzchołków powinny być wprowadzane z klawiatury po wywołaniu metody)

oraz destruktor.

W funkcji `main` stworzyć graf odpowiadający podanemu powyżej w przykładzie.

Macierz sąsiedztwa powinna być zaimplementowana jako dwuwymiarowa tablica z dynamicznie alokowaną pamięcią.

2. Należy zaimplementować algorytm djkstry do wyszukiwania najkrótszej ścieżki w grafie jako metodę klasy `graf` na podstawie podanego na następnej stronie pseudokodu.

`void dijkstra(int source, int target)` - wyszukiwanie najkrótszej ścieżki od `source` do `target`

Oraz przetestować go dla stworzonego wcześniej grafu, oraz ścieżek: (1,3), (6,7), (6, 5).

3. Rozszerzyć klasę graf w ten sposób, by maksymalna liczba wierzchołków w grafie zmieniała się, jeśli będziemy chcieli dodać kolejny wierzchołek, przekraczający tę liczbę (odpowiednie zwalnianie i alokacja pamięci!).

```

function Dijkstra(Graph, source, target):
    //deklarujemy wszystkie tablice

    for each vertex v in Graph: // Inicjalizacje, dla każdego wierzchołka.
        distance[v] := infinity ; // Odległość od source do v (nieznana, zakładamy nieskończoność)
        previous[v] := -1 ; // Tablica zapisująca wierzchołki na najkrótszej ścieżce
        VISITED[v] := 0; // Żaden z wierzchołków nie był odwiedzony. Np. jeśli
    end for ; //wierzchołek „i” odwiedzony: VISITED[i]=1.

    distance[source] := 0 ; //Odległość od punktu początkowego = 0
    if source = target // Jeśli source jest równe target - ten sam wierzchołek
        print "The same vertex"
        exit //Zakończyć działanie programu

    int tmp := infinity; //nieskończoność to np. 999999999
    int visited_nodes := 0; //licznik odwiedzonych wierzchołków
    int current = source; //current = obecny. Startujemy od wierzchołka początkowego source

    while visited_nodes < Graph size //póki nie sprawdzimy wszystkich wierzchołków w grafie
        for each node x of Graph
            if node x was not visited and there exist edge between x and current
                if distance[current] + edge weight between current and x < distance[x]
                    distance[x] := distance[current] + distance between current and x
                    previous[x] := current
            end for

        VISITED[current] := 1 //Wierzchołek current odwiedzono
        visited_nodes := visited_nodes + 1 //zwiększyć licznik odwiedzonych wierzchołków

        for each vertex v in Graph //znajdujemy nowy wierzchołek current
            if vertex v was not visited and distance[v] < tmp
                tmp := distance[v]
                current := v
            end for

        tmp := infinity

        if current = target //doszliśmy do punktu końcowego!
            print "Path found" //wypisać na ekran „ścieżka zaleźona” i wypisać ścieżkę
            int u := target;
            while previous[u] != -1
                print u
                u := previous[u]
            print source
        end while ;
        print "Path not found" //ścieżka pomiędzy source i target nie istnieje

    end Dijkstra

```

Słowniczek:

edge – krawędź (linia łącząca dwa wierzchołki), ma wagę (**weight**)

vertex – wierzchołek, charakteryzowany przez numer wierzchołka

source – punkt startowy

target – punkt końcowy