## Робота №06

*Неорієнтовані графи*

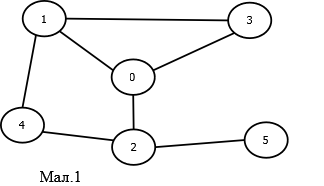
Неорієнтований граф G = (V, E) складається зі скінченної множини вершин V = {v1, v2, …,vn} та скінченної множини ребер E = {e1, e2, …, em}. Ребро визначається парою вершин (v,w), які воно з`єднує. Кожне неорієнтоване ребро графа (v,w) можна пройти в двох напрямках: від v до w i від w до v, тому в завданні графа неорієнтоване ребро (v,w) задається двома орієнтованими (v,w) і (w,v).

Неорієнтований граф не має петель, ребер виду (v,v), і не являється мультиграфом, тобто дві вершини може з»єднувати не більше ніж одне ребро.

*Маршрут* (*шлях*) в графі – це список ребер [(v0, v1), (v1, v2), (v2, v3), …, (vk-1, vk)], що веде з початкової вершини v0 в кінцеву vk, в подальшому позначається як список вершин [v0, v1, …, vk]. *Довжина шляху* – кількість дуг k. *Ланцюг* - маршрут, у якого дуги попарно різні. *Простий ланцюг* - маршрут, у якого вершини попарно різні. *Цикл* - замкнений маршрут v0==vk. *Простий цикл (контур)* - замкнений простий маршрут.

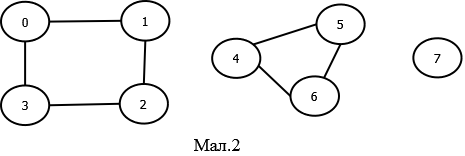
Граф зв`язний. якщо існує шлях між любою парою вершин. Компонента зв"язності графа - зв"язний підграф, що не є власним підграфом жодного іншого зв"язного підграфа (тобто максимальний зв"язний підграф). Кожний граф є пряма сума своїх компонент зв"язності.

Зв`язний графом G має ряд характеристик, пов`язаних з відстанями між вершинами.

*Відстань* між двома вершинами - довжина найкоротшого простого ланцюга, що їх з"єднує. *Ексцентриситет вершини* v: e(v)- найбільша відстань між вершиною v і всіма іншими вершинами, тобто

e(v) = max {d(v,w) | w – вершина G}

*Діаметр графу* D(G) - максимальна з ексцентриситет. *Радіус графу* R(G) - мінімальна з ексцентриситет. Вершина v - *центральна*, якщо e(v) = R(G). *Центр* - множина всіх центральних вершин.

В роботі розглядаються неорієнтовані графи, вершини яких цілі числа від 0 до m - 0..m.

Стандартне представлення графа з вершинами 0.. m (дане типу GraphS) - кортеж: максимальний номер вершини m і список ребер, в якому кожне ребро неорієнтованого графа (v,w) вказується двічі (v,w) і (w,v).

По іншому графи (дані типу Graph) задаються списками суміжності. Граф з (m+1) вершиною – це список довжини (m+1), i-ий список (0≤i≤ m) – це список всіх вершин графа, суміжних з вершиною i.

***type*** GraphS = (Int,[(Int,Int)])

***type*** Graph = [[Int]]

Кортеж gr1S і список gr1 задають неорієнтований граф (Мал.1):

gr1S =(5, [(0,1),(0,2),(0,3), (1,0),(1,3),(1,4), (2,0),(2,4),(2,5), (3,0),(3,1), (4,1),(4,2), (5,2)) ))]) .

gr1 = [[1,2,3], [0,3,4], [0,4,5], [0,1],[1,2],[2]] .

Кортеж gr2S і список gr2 задають неорієнтований граф (Мал.2):

gr2S =(7, [(0,1),(0,3), (1,0),(1,2), (2,1), (2,3), (3,0),(3,2), (4,5),(4,6), (5,4),(5,6), (6,4),(6,5)]) .

gr2 = [[1,3],[0,2],[0,2],[5,6],[4,6],[4,5],[]].

На основі допоміжного файлу, котрий включає визначення типів і тестових даних, створити файл, в якому надати визначення наступних функцій.

1. Предикат *isOrdinary gr*, що перевіряє чи задає список gr типа Graph неорієнтований граф.
   * isOrdinary gr1 = True
   * isOrdinary [[1,2],[],[0]] = False
2. Функція *fromGraph gr*, що перетворює представлення графу списком суміжності *gr* в стандартне.
   * fromGraph gr1 = gr1S
3. Функція *toGraph grS* , що перетворює стандартне представлення графу *grS* в задання списком суміжності.
   * toGraph gr2S = gr2
4. Функція *shortWay gr a b*, котра знаходить в неорієнтованому *gr* графі найкоротший шлях, що з’єднує дві вершини графа *a* і *b*. Якщо в графі не існує шляху, що з’єднує вершини *a* і *b,* то функція повертає порожній список [].
   * shortWay gr1 1 5 = [1,0,2,5]
   * shortWay gr2 1 5 = []
5. Предикат *isConnecting gr*, що перевіряє чи є неорієнтований граф *gr* – зв’язним.
   * isConnecting gr1 = True
   * isConnecting gr2 = False
6. Функція *components gr* , що знаходить всі зв’язні компоненти неорієнтованого графу *gr.*
   * components gr1 = [[0,1,2,3,4,5]]
   * components gr2 = [[0,1,2,3],[4,5,6],[7]]
7. Функція *eccentricity gr v,* що обчислює ексцентриситету вершини *v* неорієнтованого зв’язного графа *gr.*
   * eccentricity gr1 0 = 2
   * eccentricity gr1 5 = 3
8. Функції *findDiameter gr* і *findRadius gr*, котрі обчислюють, відповідно, діаметр і радіус неорієнтованого зв’язного графа *gr*.
   * findDiameter gr1 = 3
   * findRadius gr1 = 2
9. Функція *findCenter gr*, що знаходить список вершин, котрі утворюють центр неорієнтованого зв’язного графа *gr*.
   * findCenter gr1 = [0,2,4]
10. Функція *shortWayAll gr a b*, котра знаходить в неорієнтованому *gr* графі всі різні найкоротші шляхи, що з’єднують дві вершини графа *a* і *b* .
    * shortWays gr1 1 5 = [[1,0,2,5],[1,4,2,5]]
    * shortWays gr2 1 5 = []

*isOrdinary* :: Graph -> Bool

*fromGraph*  :: Graph -> GraphS

*toGraph*  :: GraphS -> Graph

*shortWay* :: Graph -> Int -> Int -> [Int]

*isConnecting*:: Graph -> Bool

*components*  :: Graph -> [[Int]]

*eccentricity* :: Graph -> Int -> Int

*findDiameter*:: Graph -> Int

*findRadius* :: Graph -> Int

*findCenter* :: Graph -> [Int]

*shortWayAll* :: Graph -> Int -> Int -> [ [Int]]

Зауваження:

Назва файлу Family06.hs (Family – прізвище студента). Файл включає модуль Family06 і створюється на основі файла-заготовки HWP06.hs