## Робота №8

*Графи*

Граф G = (V, E) складається зі скінченної множини вершин V = {v1, v2, …,vn} та скінченної множини ребер E = {e1, e2, …, em}. Кожне ребро визначається парою вершин (v,w), які воно з`єднує.

Граф називається: орієнтованим, якщо пара вершин кожного ребра (v,w) впорядкована; неорієнтованим, якщо ребра – неупорядковані пари, тобто (v,w) = (w, v).

Будуть розглядатися графи, вершини яких нумеруються послідовними числами від 1 до n (загальна кількість вершин графу). Для представлення графу використовуються списки суміжностей.

Орієнтований граф G = ({1,2,3,4},{(1,2), (2,1), (1,3), (3,1), (1,4), (4,1), (4,2), (2,4)}) має 4 вершини 1..4 і його можна представити списком асоціацій [(1,[2,3,4]), (2,[1,4]), (3,[1]), (4,[1,2])]. Кожний елемент списку (v,[v1, …, vk]) задає вершину графа v і список суміжностей (її сусідів) - [v1, …, vk].

Зауважимо, що цей же список задає також і неорієнтований граф G1 = ({1,2,3,4},{(1,2), (1,3), (1,4), (4,2)}). В традиційному представленні неорієнтованого графу кожна його дуга «представляється» двома дугами з напрямком.

Список асоціацій [(1,[2,3,4]), (2,[1]), (3,[]), (4,[2])] задає орієнтований граф G = ({1,2,3,4},{(1,2), (1,3), (1,4), (2,1), (4,2)}) і не задає жодного неорієнтованого графа.

В цій роботі для реалізації представлення графів списками суміжностей використовується тип

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

Масиви (Array) використовуються в Haskell для зберігання і доступу елементів, у яких кожний має унікальний *індекс*. Масиви не входять в Prelude, для їх використання потрібно імпортувати модуль Data.Array.

Шлях в орієнтованому чи неорієнтованому графі – це послідовність ребер L = <(v1, v2), (v2, v3), …, (vk-1, vk)>, де всі вершини v1, v2, v3, …, vk-1 різні, а v1 і vk можуть збігатися (в задачах шлях L будемо задавати списком [v1, v2, v3, …, vk-1, vk] ).

Цикл в графі – це шлях, у якого перша і остання вершини збігаються, тобто v1 = vk .

Граф, що не має циклів, називається ациклічним.

Орієнтований граф G = (V, E) називається транзитивним за виконанням умови: якщо (v, u) Є E і (u, w) Є E, то (v, w) Є E.

Граф G1 = (V1, E1) являє собою підграф G = (V, E), якщо V1 – підмножина V і E1 – підмножина E.

Неорієнтований граф G зв’язний, якщо існує шлях між кожною парою його вершин або якщо він складається з однієї ізольованої вершини. Зв’язним компонентом графа G є його максимальний зв’язний підграф. Незв’язний граф складається з кількох зв’язних компонентів.

Топологічне сортування ациклічного орієнтованого графу – це впорядкування його вершин згідно з частковим порядком, визначеним ребрами цього графа на множині його вершин. Тобто топологічне сортування – це список усіх вершин графа такий, що для кожного ребра (v,u) з вершини v у вершину u, вершина v зустрічається в списку раніше ніж u.

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

1. Функція *longWay gr a b*, котра знаходить в орієнтованому *gr* графі найдовший шлях, що з’єднує дві вершини графа *a* і *b*
2. Предикат *isNoCycle gr*, що перевіряє чи являється орієнтований граф *gr* - ациклічним.
3. Предикат *isTransitive gr*, що перевіряє чи являється орієнтований граф *gr* - транзитивним.
4. Предикат *isGraph gr*, що перевіряє чи задає список gr типа Graph неорієнтований граф.
5. Функція *shortWay gr a b*, котра знаходить в неорієнтованому *gr* графі найкоротший шлях, що з’єднує дві вершини графа *a* і *b* .
6. Предикат *isConnecting gr*, що перевіряє чи є неорієнтований граф *gr* – зв’язним.
7. Функція *allComponents gr* , що знаходить всі зв»язні компоненти неорієнтованого графу *gr.*
8. Функція *topolSorting gr,* котра виконує топологічне сортування орієнтованого графа *gr.*

*longWay*  :: Graph -> Int -> Int -> Maybe [Int]

*isNoCycle* :: Graph -> Bool

*isTransitive* :: Graph -> Bool

*isGraph* :: Graph -> Bool

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

*isConnecting* :: Graph -> Bool

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

*topolSorting* :: Graph -> Maybe [Int]

Зауваження:

Назва файлу Family08.hs (Family – прізвище студента). Файл включає модуль Family08 і створюється на основі файла-заготовки HWP08.hs