*Лекція 14 (закінчення)*

# Тема 4. Графи та їхні властивості

## План лекції

## Поняття дерева

## Обходи графів

## Поняття дерева

Поняття дерева широко використовують у багатьох розділах математики й інформатики. Наприклад, дерева використовують як інструмент під час обчислень, як зручний спосіб збереження даних, їх сортування чи пошуку.

*Деревом* називають неорієнтований зв'язний граф без простих циклів. Неорієнтований граф, який не містить простих циклів і складається з $k$ компонент, називають *лісом* із $k$ дерев.

**Приклад.** На рис. 8 зображено приклади дерев. Граф, який зображено на рис. 9 – не дерево, бо він незв'язний.

*Зауваження.* З означення випливає, що дерева й ліси є простими графами.

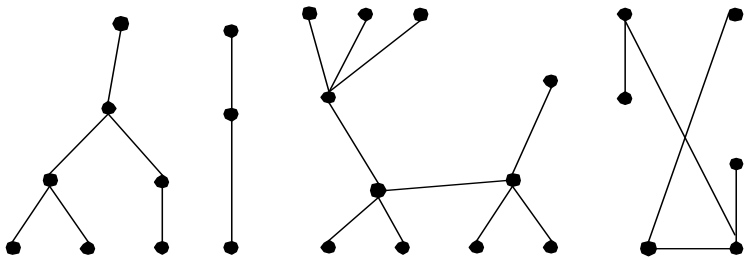


**Рис. 8**

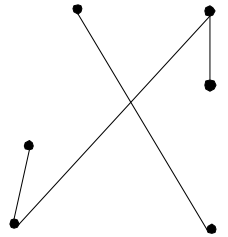


**Рис. 9**

**Теорема.** Дерево з $n$ вершинами містить точно $n$ – 1 ребро.

## Обходи графів

Існує багато алгоритмів на графах, які ґрунтуються на систематичному переборі їх вершин або обході вершин, під час якого кожна вершина одержує унікальний порядковий номер. Методи обходу вершин графа називають *методами пошуку.* Під час обходу графа ми певним чином позначаємо ребра, які утворять дерево обходу.

## Пошук углиб у простому зв'язному графі

Опишемо метод пошуку в простому зв'язному графі. Цей метод називають *пошуком вглиб*, або DFS-методом (від англ. Depth First Search).

Нехай $G = (V, E)$ – простий зв'язний граф, усі вершини якого позначені попарно різними символами. У процесі пошуку вглиб вершинам графа $G$ надають номери (DFS-номери), та певним чином позначають ребра. У ході роботи алгоритму використовують структуру даних для збереження множин, яку називають *стеком* (англ. stack – стіг).

Зі стеку можна вилучити тільки той елемент, котрий було додано до нього останнім: стек працює за принципом „останнім прийшов – першим вийшов" (last in, first out – скорочено LIFO). Інакше кажучи, додавання й вилучення елементів у стеку відбувається з одного кінця, який називають *верхівкою стеку*. DFS-номер вершини $x$ позначають як DFS($x$).

## Алгоритм пошуку вглиб у простому зв'язному графі

Крок 1. Почати з довільної вершини $v_s$. Виконати DFS($v_s$):=1. Включити вершину $v_s$ у стек.

Крок 2. Розглянути вершину, яка знаходиться у верхівці стеку, нехай це буде вершина $x$. Якщо для всіх вершин, суміжних із вершиною $x$, уже визначені DFS-номери, то перейти до кроку 4, інакше – до кроку 3.

Крок 3. Нехай $\{x, y\}$ – ребро, у якому номер DFS($y$) не визначений. Позначити це ребро потовщеною суцільною лінією (або внести в список ребер дерева), визначити DFS($y$) як черговий DFS-номер, включити вершину $y$ у стек й перейти до кроку 2.

Крок 4. Вилучити вершину $x$ зі стеку. Якщо стек порожній, то зупинитись, інакше – перейти до кроку 2.

Щоб вибір номерів був однозначним, доцільно домовитись, що вершини, суміжні з тією, яка вже отримала DFS-номер, аналізують за зростанням їхньої нумерації (або в алфавітному порядку). Динаміку роботи алгоритму зручно відображати за допомогою таблиці з чотирма стовпцями: вершина, DFS-номер, уміст стеку, список ребер дерева. Її називають *протоколом обходу* графа пошуком вглиб.

**Приклад.** Виконаємо обхід графа на рис. 10 пошуком вглиб, починаючи з вершини $b$. Розв'язок подано на рис. 11; протокол пошуку вглиб подано в таблиці поруч з рисунками. У цій таблиці в третьому стовпці вважаємо, що верхівка стеку праворуч. У четвертому стовпці таблиці накопичуються ребра дерева пошуку вглиб (його іноді називають *глибинним деревом*). На рисунку це дерево позначено потовщеними лініями.

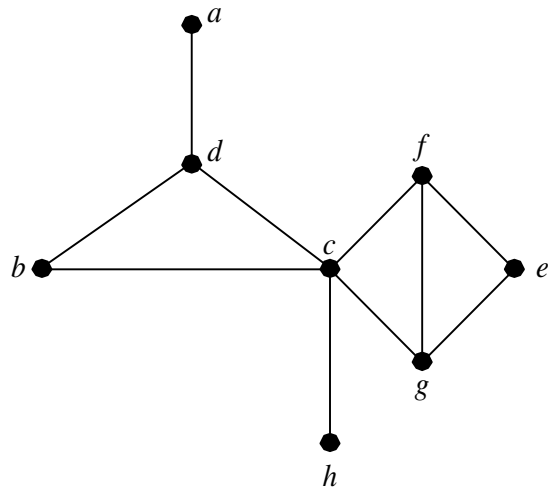



**Рис. 10**

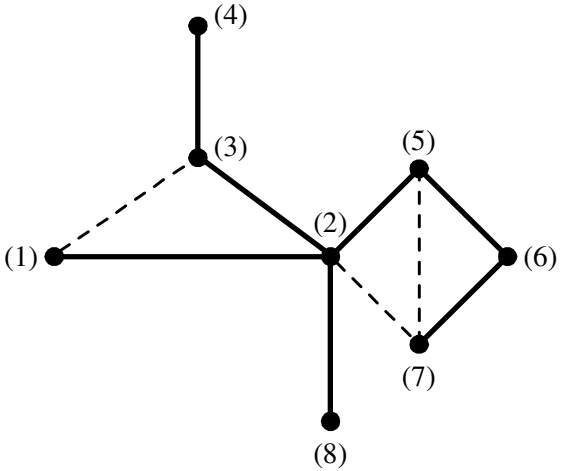
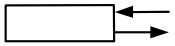



**Рис. 11**

**Таблиця. Протокол обходу графа пошуком вглиб**

| **Вершина** | **DFS-номер** | **Уміст стеку**  | **Список ребер дерева** |
|---|---|---|---|
| *b* | 1 | *b* | – |
| *c* | 2 | *bc* | {*b*, *c*} |
| *d* | 3 | *bcd* | {*c*, *d*} |
| *a* | 4 | *bcda* | {*d*, *a*} |
| – | – | *bcd* | – |
| – | – | *bc* | – |
| *f* | 5 | *bcf* | {*c*, *f*} |
| *e* | 6 | *bcfe* | {*f*, *e*} |
| *g* | 7 | *bcfeg* | {*e*, *g*} |
| – | – | *bcfe* | – |
| – | – | *bcf* | – |
| – | – | *bc* | – |
| *h* | 8 | *bch* | {*c*, *h*} |
| – | – | *bc* | – |
| – | – | *b* | – |
| – | – | ∅ | – |

## Пошук ушир у простому зв'язному графі

У процесі *пошуку вшир* вершини графа проглядають в іншій послідовності, ніж у методі пошуку вглиб, і їм надають BFS-номери (від англ. Breadth First Search). BFS-номер вершини $x$ позначають як BFS($x$). Назва пояснюється тим, що під час пошуку рухаються вшир, а не вглиб: спочатку проглядаються всі сусідні вершини, після цього – сусіди сусідів і так далі.

У ході реалізації алгоритму використовують структуру даних для збереження множин, яку називають *чергою* (англ. queue). Із черги можна вилучити тільки той елемент, який перебував у ній найдовше: працює принцип „першим прийшов – першим вийшов" (first in, first out – скорочено FIFO). Елемент включається в *хвіст* черги, а виключається з її *голови*. Пошук ушир, узагалі кажучи, відрізняється від пошуку вглиб заміною стеку на чергу. Після такої модифікації що раніше відвідується вершина (включається в чергу), то раніше вона використовується (і виключається із черги). Використання вершини полягає в перегляді одразу всіх іще не відвіданих її сусідів. Усю процедуру подано нижче.

## Алгоритм пошуку вшир у простому зв'язному графі

Крок 1. Почати з довільної вершини $v_s$. Виконати BFS($v_s$):=1. Включити вершину $v_s$ у хвіст черги.

Крок 2. Розглянути вершину, яка знаходиться в голові черги, нехай це буде вершина $x$. Якщо для всіх вершин, суміжних із вершиною $x$, уже визначені BFS-номери, то перейти до кроку 4, інакше – до кроку 3.

Крок 3. Нехай $\{x, y\}$ – ребро, у якому номер BFS($y$) не визначений. Позначити це ребро потовщеною суцільною лінією (або внести в список ребер дерева), визначити BFS($y$) як черговий BFS-номер, включити вершину $y$ у хвіст черги й перейти до кроку 2.

Крок 4. Вилучити вершину $x$ із голови черги. Якщо черга порожня, то зупинитись, інакше – перейти до кроку 2.

Щоб результат виконання алгоритму був однозначним, вершини, які суміжні з вершиною $x$, аналізують за зростанням їх нумерації (або в алфавітному порядку). Динаміку роботи алгоритму пошуку вшир також зручно відображати за допомогою протоколу обходу. Він аналогічний попередньому й відрізняється лише третім стовпцем: тепер це – уміст черги (уважаємо, що голова черги ліворуч, а хвіст – праворуч).

**Приклад.** Виконаємо обхід графа на рис. 12 пошуком вшир, починаючи з вершини $b$. Розв'язок зображено на рис. 13; протокол пошуку вшир подано в таблиці поруч із рисунками. У четвертому стовпці таблиці накопичуються ребра дерева пошуку вшир. На

рисунку це дерево позначено потовщеними лініями. Пошук углиб і пошук ушир – важлива складова проектування алгоритмів на графах.

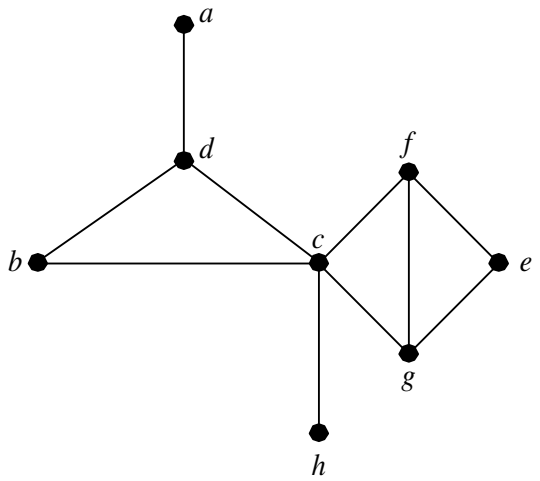



**Рис. 12**

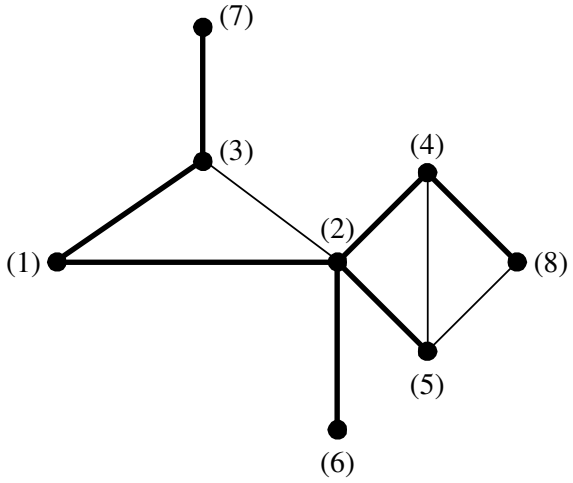



**Рис. 13**

**Таблиця. Протокол обходу графа пошуком вшир**

| **Вершина** | **BFS-номер** | **Вміст черги** ← ▭ ← | **Список ребер дерева** |
|---|---|---|---|
| *b* | 1 | *b* | – |
| *c* | 2 | *bc* | {*b*, *c*} |
| *d* | 3 | *bcd* | {*b*, *d*} |
| – | – | *cd* | – |
| *f* | 4 | *cdf* | {*c*, *f*} |
| *g* | 5 | *cdfg* | {*c*, *g*} |
| *h* | 6 | *cdfgh* | {*c*, *h*} |
| – | – | *dfgh* | – |
| *a* | 7 | *dfgha* | {*d*, *a*} |
| – | – | *fgha* | – |
| *e* | 8 | *fghae* | {*f*, *e*} |
| – | – | *ghae* | – |
| – | – | *hae* | – |
| – | – | *ae* | – |
| – | – | *e* | – |
| – | – | ∅ | – |