Лекція 25. Дерева

25.1. Основні означення та властивості

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

Означення 25.1. **Деревом** називають зв'язний граф без простих циклів. Граф, який не містить простих циклів і складається з k компонент зв'язності, називають **лісом** з k дерев.

З означення випливає, що дерева й ліси являють собою прості графи.

На рис. 25.1 зображені приклади дерев.

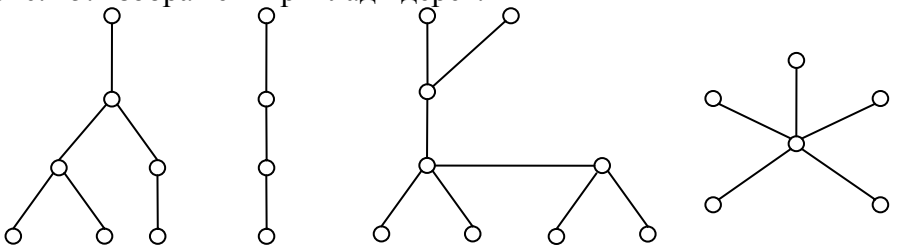


Рис. 25.1.

Теорема 25.1. Нехай граф G має n вершин. Тоді такі твердження еквівалентні:

1) граф G – дерево;
2) граф G не містить простих циклів і має (n-1) ребро;
3) граф G зв'язний і має (n-1) ребро;
4) граф G зв'язний, але вилучення довільного ребра робить його незв'язним;
5) довільні дві вершини графа G з'єднані точно одним простим маршрутом;
6) граф G не містить простих циклів, але, додавши до нього нове ребро, ми отримаємо точно один простий цикл.

Доведення. Методом математичної індукції за кількістю вершин. У разі n=1 твердження тривіальні. Припустимо, що твердження виконуються для n-1. Доведемо їх для n.

(1) $\rightarrow$ (2). За означенням G не містить простих циклів. Отже, вилучивши довільне ребро, ми одержимо два графи, кожний з яких являє собою дерево з меншою, ніж у G, кількістю вершин. Нехай, граф G_1 містить n_1 вершин, а G_2 – n_2: $n_1 + n_2 = n$. За припущенням індукції кількість ребер у кожному з отриманих дерев на 1 менша за кількість вершин, тобто в графі G_1 (n_1-1) ребер, а у графі G_2 (n_2-1) ребер. Отже у графі G було (n_1-1) + (n_2-1) + 1 = (n-1) ребро.

(2) $\rightarrow$ (3). Припустимо, що граф G незв'язний. Тоді кожна його компонента являє собою зв'язний граф без простих циклів (за означенням), тобто дерево. Звідси випливає, що кількість вершин у кожній компоненті на одиницю більша від кількості ребер. Отже, загальна кількість вершин графа G більша за кількість ребер принаймні на 2. Але це суперечить тому, що граф має (n-1) ребро.

(3) $\rightarrow$ (4). Вилучимо довільне ребро, отримаємо граф з n вершинами та (n-2) ребрами. Припущення про зв'язність такого графа суперечить теоремі про оцінку (знизу) кількості ребер звичайного графа (теорема 23.3).

(4) $\rightarrow$ (5). Оскільки граф G зв'язний, то кожну пару його вершин з'єднано принаймні одним простим маршрутом (теорема 23.1). Якщо якусь пару вершин з'єднано двома простими маршрутами, вони замикаються в простий цикл. Але це суперечить тому, що вилучення довільного ребра робить граф G незв'язним.

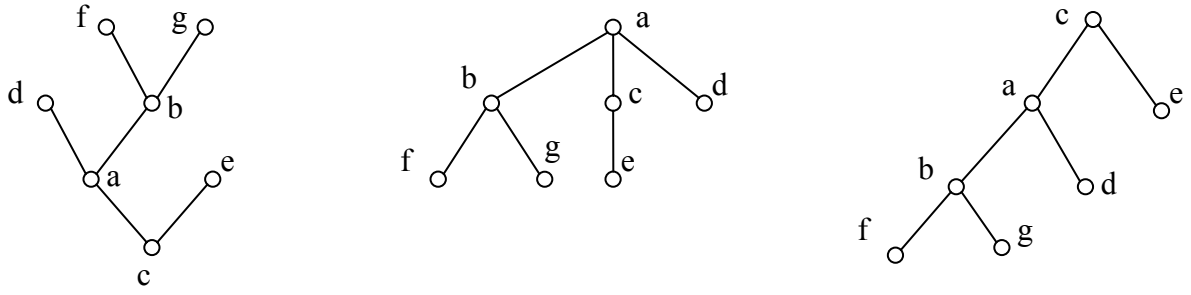
(5) $\rightarrow$ (6). Припустимо, що граф G містить простий цикл. Тоді довільні дві вершини цього циклу з'єднано принаймні двома простими шляхами, що суперечить твердженню (5). Додавши тепер до графа G ребро e, отримаємо єдиний простий цикл, бо інцидентні ребру e вершини вже з'єднано в графі G точно одним простим шляхом.

(6) → (1). Припустимо, що граф G незв'язний. Тоді додавання будь-якого ребра, що з'єднує вершину однієї компоненти з вершиною іншої, не зумовлює утворення простого циклу, що суперечить твердженню (6).►

Наслідок з твердження (2). Ліс з k дерев, який містить n вершин, має (n-k) ребер.

У багатьох застосуваннях певну вершину дерева означають як **корінь**. Тоді можна природно приписати напрямок кожному ребру. Оскільки існує єдиний простий маршрут від кореня до кожної вершини графа, то можна орієнтувати кожне ребро в напрямку від кореня. Отже, дерево разом із виділеним коренем утворює орієнтований граф, який називають **кореневим деревом**.

Різні способи вибору кореня дають змогу утворити різні кореневі дерева. Наприклад, на рис. 25.2, а зображено дерево, а на рис 25.2, б, в – кореневі дерева з коренями відповідно у вершинах a та c.



а б в

Рис. 25.2.

Означення 25.2. Нехай G – кореневе дерево. Якщо v – його вершина, відмінна від кореня, то її **батьком** називають єдину вершину u таку, що є орієнтоване ребро (u, v). Якщо u – батько, то v – **син**. Аналогічно за генеалогічною термінологією можна означити інших пращурів і нащадків вершини v. Вершини дерева, які не мають синів, називаються **листками**. Вершини, які мають синів, називаються **внутрішніми**. Нехай *a* – вершина дерева. Тоді **піддеревом** із коренем *a* називають підграф, який містить *a* та всі вершини – нащадки вершини *a*, а також інцидентні їм ребра.

Означення 25.3. Кореневе дерево називають **m-арним**, якщо кожна його внутрішня вершина має не більше ніж m синів. Дерево називають **повним m-арним**, якщо кожна його внутрішня вершина має точно m синів. У разі m=2 дерево називають **бінарним**.

Означення 25.4. Кореневе дерево, у якому сини кожної внутрішньої вершини впорядковано, називають **упорядкованим.** Таке дерево зображають так, щоб сини кожної вершини були розміщені зліва направо. Якщо внутрішня вершина впорядкованого бінарного дерева має двох синів, то першого називають **лівим**, а другого – **правим**.

Теорема 25.2. Повне m-арне дерево з r внутрішніми вершинами містить n = mr + 1 вершин.

Доведення. Кожна вершина, окрім кореня, - син внутрішньої вершини. Оскільки кожна з r внутрішніх вершин має m синів, то всього є, якщо не враховувати корінь, mr вершин, а з урахуванням кореня їх mr + 1. ►

Означення 25.5. **Рівнем вершини** v в кореневому дереві називають довжину простого шляху від кореня до цієї вершини (цей шлях, очевидно, єдиний). Рівень кореня вважають нульовим. **Висотою** кореневого дерева називають максимальний із рівнів його вершин. Інакше кажучи, висота кореневого дерева – це довжина найдовшого простого шляху від кореня до будь-якої вершини. Повне m-арне дерево, у якого всі листки на одному рівні, називають **завершеним**.

Кореневе m-арне дерево з висотою h називають **збалансованим**, якщо всі його листки на рівнях h або h-1.

Теорема 25.3. Нехай m-арне дерево має висоту h. Тоді в ньому не більше ніж m^k листків.

Доведення. За методом індукції по h. У разі h=1 твердження очевидне. Припустимо, що воно справджується для всіх m-арних дерев з висотою h-1. Доведемо її для дерева G з висотою h. Тоді його листки – це листки піддерев, отриманих із G вилученням ребер, які з'єднують корінь дерева G з кожною вершиною рівня 1, тобто з кожним сином кореня.

Кожне з цих піддерев має не більшу висоту, ніж h-1. За індуктивним припущенням всі вони мають не більше ніж m^{h-1} листків. Позаяк таких піддерев не більше ніж m, то загальна кількість листків у дереві G не перевищує $mm^{h-1} = m^h$. ►

Наслідок. Якщо m-арне дерево з висотою h має t листків, то $h \geq \lceil \log_m t \rceil$. Якщо m-арне дерево повне та збалансоване, то $h = \lceil \log_m t \rceil$. $\lceil x \rceil$ - найменше ціле число, яке більше чи дорівнює x.

Доведення. За теоремою 25.3 $t \leq m^h$. Прологарифмуємо цю нерівність за основою m: $\log_m t \leq h$. Оскільки h – ціле, то $h \geq \lceil \log_m t \rceil$. Тепер припустимо, що дерево повне та збалансоване. Вилучимо всі листки на рівні h (разом з інцидентними їм ребрам). Отримаємо завершене m-арне дерево висотою h-1. Воно має m^{h-1} листків. Отже, $m^{h-1} < t \leq m^h$. Звідси випливає, що $h - 1 < \log_m t \leq h$, тобто $h = \lceil \log_m t \rceil$. ►

25.2. Обхід дерев

Чимало задач можна моделювати з використанням кореневих дерев. Поширене таке загальне формулювання задачі: виконати задану операцію D з кожною вершиною дерева. Тут D – параметр загальнішої задачі відвідування всіх вершин, або так званого **обходу дерева**. Розглядаючи розв'язання цієї задачі як єдиний послідовний процес відвідування вершин дерева в певному порядку, можна вважати їх розміщеними одна за одною. Опис багатьох алгоритмів істотно спрощується, якщо можна говорити про наступну вершину дерева, маючи на увазі якесь упорядкування. Є три принципи впорядкування вершин, які природно випливають зі структури дерева. Як і саму деревоподібну структуру, їх зручно формулювати за допомогою рекурсії.

Звертаючись до бінарного дерева, де R – корінь, A та B – ліве та праве піддерева (рис. 25.3), можна означити такі впорядкування.

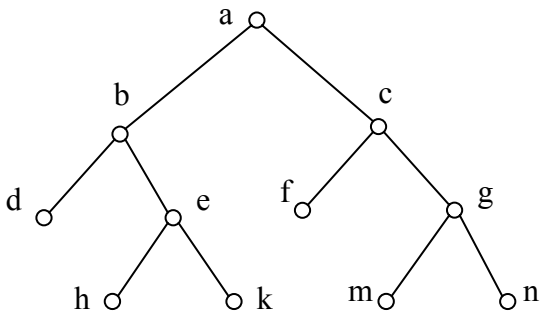
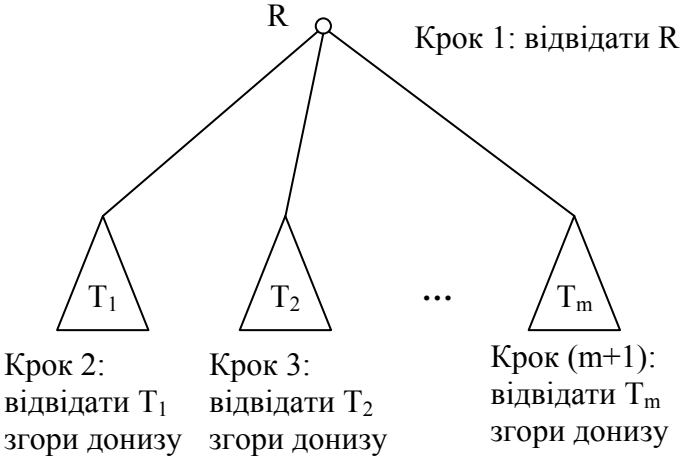


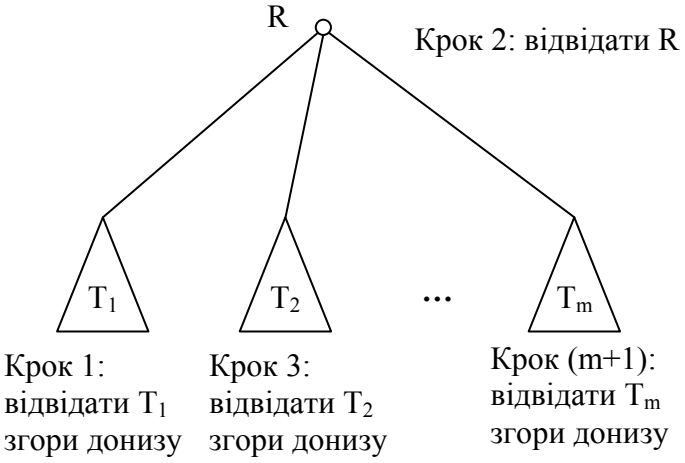
Рис. 25.3.

1. Обхід **у прямому порядку** (preorder), або **згори донизу**: R, A, B (корінь відвідують до обходу піддерев). Для дерева з рис. 25.3 послідовність вершин буде така: a, b, d, e, h, k, c, f, g, m, n.
2. Обхід у **внутрішньому порядку** (inorder), або **зліва направо**: A, R, B. Для дерева з рис. 25.3 послідовність: d, b, h, e, k, a, f, c, m, g, n.
3. Обхід у **зворотному порядку** (postorder), або знизу догори: A, B, R (корінь відвідують після обходу піддерев). Для дерева з рис. 25.3 послідовність:d, h, k, e, b, f, m, n, g, c, a.

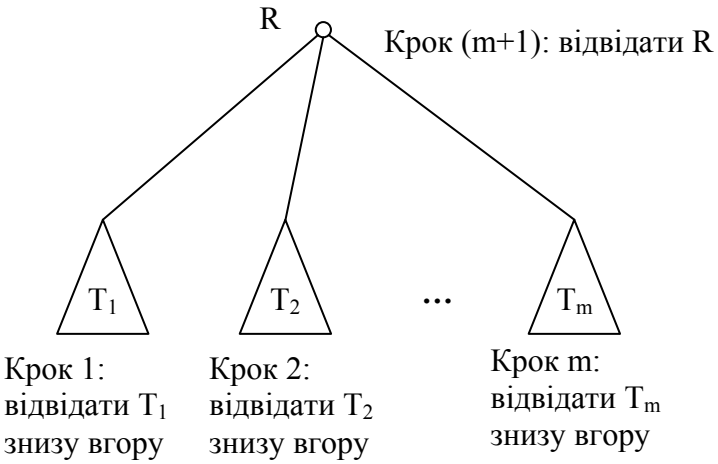
Зазначені способи обходу бінарних дерев можна узагальнити й на довільні m-арні дерева. Обхід таких дерев у прямому порядку (згори донизу) схематично зображено на рис. 25.4, а; у внутрішньому порядку (зліва направо) – на рис. 25.4, б; у зворотному (знизу вгору) – на рис. 25.4, в.



а



б



в

Рис. 25.4.

25.3. Приклад застосування дерев

Надзвичайно поширене в інформатиці застосування обходу дерев – зіставлення виразам (арифметичним, логічним тощо) дерев і побудова на цій основі різних форм запису виразів. Суть справи зручно пояснити на прикладі. Розглянемо арифметичний вираз:

$$\left(a+\frac{b}{c}\right)\times\left(d-e\times f\right).$$

Подамо його у вигляді дерева (рис. 25.5). Внутрішнім вершинам цього дерева відповідають символи операцій, а листкам – операнди.

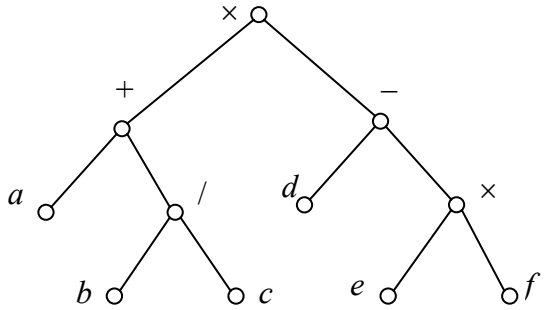


Рис. 25.5.

Обійдемо це дерево, записуючи символи у вершинах у тому порядку, у якому вони зустрічаються в разі заданого способу обходу. Отримаємо такі три послідовності:

- у разі обходу в прямому порядку – **префіксний** (**польський**) **запис**
$$\times + a / bc - d \times ef ;$$
- у разі обходу у внутрішньому порядку – **інфіксний запис** (поки що без дужок, які потрібні для визначення порядку операцій)
$$a + b / c \times d - e \times f ;$$
- у разі обходу в зворотному порядку – **постфіксний** (**зворотний польський**) **запис**
$$abc / + def \times - \times .$$

Звернімося спочатку до інфіксної форми запису виразу. Без дужок вона неоднозначна: один запис може відповідати різним деревам. Щоб уникнути неоднозначності інфіксної форми, використовують круглі дужки щоразу, коли зустрічають операцію. Повністю «одужкований» вираз, одержаний під час обходу дерева у внутрішньому порядку, називають **інфіксною формою запису**. Отже, для дерева з рис. 25.5 інфіксна форма така: $((a + (b / c)) \times (d - (e \times f)))$.

Наведені міркування свідчать, що інфіксна форма запису виразів незручна. На практиці використовують префіксну та постфіксну форми, бо вони однозначно відповідають виразу й не потребують дужок. Ці форми запису називаються **польським записом** (на честь польського математика Яна Лукасевича).

Для обчислення значення виразу в польському записі його проглядають справа наліво та знаходять два операнди разом зі знаком операції перед ними. Ці операнди та знак операції вилучають із запису, виконують операцію, а її результат записують на місце вилучення символів.

Для прикладу, обчислимо значення виразу в польському записі (^ - означає піднесення до степеня): «+ × – 4 2 5 ^ 2 / 9 3». За сформульованим правилом виділимо «/ 9 3», ці символи вилучимо й обчислимо 9 / 3 = 3; результат запишемо на місце вилучення символів «+ × – 4 2 5 ^ 2 3». Продовжимо обчислення. Динаміку процесу відображено в табл. 25.1.

Крок	**Вираз**	**Виділені символи**	**Виконання операції**
1	+ × – 4 2 5 ^ 2 / 9 3	/ 9 3	9 / 3 = 3
2	+ × – 4 2 5 ^ 2 3	^ 2 3	2^3 = 8
3	+ × – 4 2 5 8	– 4 2	4 – 2 = 2
4	+ × 2 5 8	× 2 5	2 × 5 = 10
5	+ 10 8	+ 10 8	10 + 8 = 18
6	18		

Табл. 25.1.

Для обчислення значення виразу в зворотному польському записі його переглядають зліва направо та виділяють два операнди разом зі знаком операції після них. Ці операнди та знак операції вилучають, а її результат записують на місце вилучених символів.

Оскільки польські записи однозначні та їх значення можна легко обчислити без сканування назад і вперед, їх широко використовують у комп'ютерних науках, особливо для конструювання компіляторів.

Наприклад, розглянемо можливий спосіб компіляції булевої функції (представлення функції у польському записі та обчислення значень функції залежно від значень операндів):

$$(\neg x \wedge (y \sim \neg x)) \vee \neg y.$$

Подамо його у вигляді дерева (рис. 25.6).

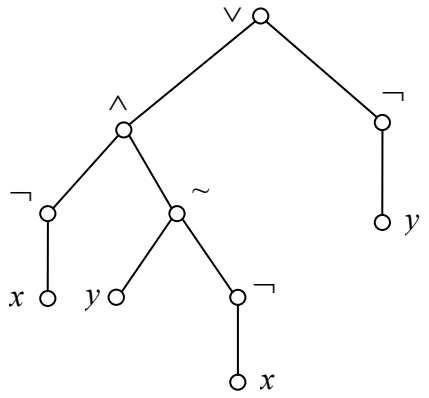


Рис. 25.6.

Згідно з прямим та зворотним порядками обходу дерева отримаємо відповідно пряму та зворотну польські форми записів: «$\vee \wedge \neg x \sim y \ \neg x \neg y$» та «$x \neg y x \neg \sim \wedge y \neg \vee$».

Відмітимо один факт, який стосується булевих виразів. Як відомо, в булевій алгебрі є одна унарна операція – заперечення «$\neg$». Тож ми не можемо сліпо виконувати правила, зазначені раніше, про заміну двох операндів та операції, яка йде перед чи після операндів, на результат операції. Потрібно ввести обмеження на використання цих правил, зазначивши, що вони виконуються тільки для бінарних операцій. Для унарних же операцій (заперечення «$\neg$») правила мають вигляд: замінювати операнд та операцію заперечення «$\neg$», яка стоїть *перед* операндом, на результат операції (для прямої польської форми) та замінювати операнд та операцію заперечення «$\neg$», яка стоїть *після* операнда, на результат операції (для оберненої польської форми).

Отже, для прямої польської форми отримаємо таке загальне правило обчислення результату булевих виразів. Переглядаємо послідовність справа наліво та виділяємо один операнд та операцію заперечення «$\neg$», яка стоїть перед цим операндом, або два операнди та операцію, відмінну від заперечення та яка стоїть перед цими операндами. Вибрані операнд(и) та операцію замінюємо в послідовності на результат застосування операції до операнда(ів). Аналогічно формулюється правило обчислення результату булевих виразів у зворотній польській формі.

Обчислимо значення функції на наборі (1, 0), використовуючи прямий польський запис. Для цього замість символів x та y підставляємо 1 та 0 відповідно й застосовуємо вказане правило доки не отримаємо кінцевий результат (табл. 25.2).

Проведемо аналогічні обчислення на наборі (0, 0), але використовуючи вже зворотній польській запис (табл. 25.3).

Крок	**Вираз**	**Виділені символи**	**Виконання операції**
1	$\vee \wedge \neg 1 \sim 0 \neg 1 \neg 0$	$\neg 0$	$\neg 0 = 1$
2	$\vee \wedge \neg 1 \sim 0 \neg 1 1$	$\neg 1$	$\neg 1 = 0$
3	$\vee \wedge \neg 1 \sim 0 0 1$	$\sim 0 0$	$0 \sim 0 = 1$
4	$\vee \wedge \neg 1 1 1$	$\neg 1$	$\neg 1 = 0$
5	$\vee \wedge 0 1 1$	$\wedge 0 1$	$0 \wedge 1 = 0$
6	$\vee 0 1$	$\vee 0 1$	$0 \vee 1 = 1$
7	1		

Табл. 25.2.

Крок	**Вираз**	**Виділені символи**	**Виконання операції**
1	$0 \neg 0 0 \neg \sim \wedge 0 \neg \vee$	$0 \neg$	$\neg 0 = 1$
2	$1 0 0 \neg \sim \wedge 0 \neg \vee$	$0 \neg$	$\neg 0 = 1$
3	$1 0 1 \sim \wedge 0 \neg \vee$	$0 1 \sim$	$0 \sim 1 = 0$
4	$1 0 \wedge 0 \neg \vee$	$1 0 \wedge$	$1 \wedge 0 = 0$
5	$0 0 \neg \vee$	$0 \neg$	$\neg 0 = 1$
6	$0 1 \vee$	$0 1 \vee$	$0 \vee 1 = 1$
7	1		

Табл. 25.3.