ДЕРЕВА (НЕЛІНІЙНІ СПИСКИ)

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

1. Основні означення та властивості дерев

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

Приклад. На рис. 1 зображено приклади дерев. Граф, зображений на рис. 2 - не дерево, бо він незв'язний.

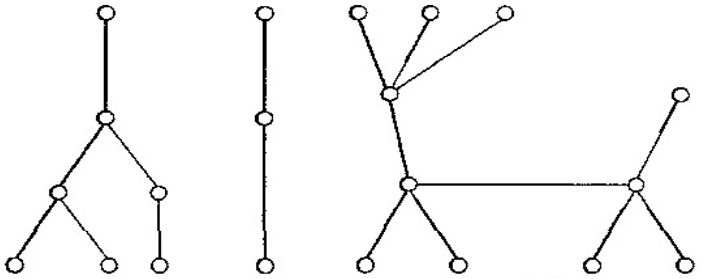

Рис. 1

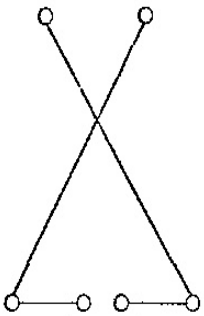

Рис. 2

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

Теорема 1. Нехай граф T має n вершин. Тоді такі твердження еквівалентні:

1. граф T - дерево;
2. граф T не містить простих циклів і має $(n - 1)$ ребро;
3. граф T зв'язний і має $(n - 1)$ ребро;
4. граф T зв'язний, але вилучення довільного ребра робить його незв'язним;
5. довільні дві вершини графа T з'єднані точно одним простим шляхом;
6. граф T не містить простих циклів, але, додавши до нього довільне нове ребро, ми отримаємо точно один простий цикл.

Доведення (математичною індукцією). У разі $n = 1$ твердження тривіальні; припустимо, що $n \geq 2$.

$1 \rightarrow 2$. За означенням T не містить простих циклів. Отже, вилучивши довільне ребро, ми одержимо два графи, кожний з яких являє собою дерево з меншою, ніж у T, кількістю вершин. За припущенням індукції кількість ребер у кожному з отриманих дерев на 1 менша за кількість вершин. Звідси випливає, що граф Т має $(n - 1)$ ребро.

$2 \rightarrow 3$. Припустимо, що граф T незв'язний. Тоді кожна його компонента являє собою зв'язний граф без простих циклів, тобто дерево. Звідси випливає, що кількість вершин у кожній компоненті на одиницю більша від кількості ребер. Отже, загальна кількість вершин графа T більша за кількість ребер принаймні на 2. Але це суперечить тому, що граф T 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 (див. розділ „Шляхи та цикли. Зв’язність”).

$4 \rightarrow 5$. Оскільки граф T зв'язний, то кожну пару його вершин з'єднано принаймні одним простим шляхом (див. розділ „Шляхи та цикли. Зв’язність”). Якщо якусь пару вершин з'єднано двома простими шляхами, вони замикаються в простий цикл. Але це суперечить тому, що вилучення довільного ребра робить граф T незв'язним.

$5 \rightarrow 6$. Припустимо, що граф T містить простий цикл. Тоді довільні дві вершини цього циклу з'єднано принаймні двома простими шляхами, що суперечить твердженню (5). Додавши тепер до графа T ребро e, одержимо єдиний простий цикл, бо інцидентні ребру e вершини вже з'єднано в графі T точно одним простим шляхом.

$6 \rightarrow 1$. Припустимо, що граф T 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 із твердження (2). Ліс із 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 шлях 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.

Приклад 2. На рис. 3 *а* зображено дерево, а на рис. 3 *б*, *в* - кореневі дерева з коренями відповідно у вершинах *a* та *c*.

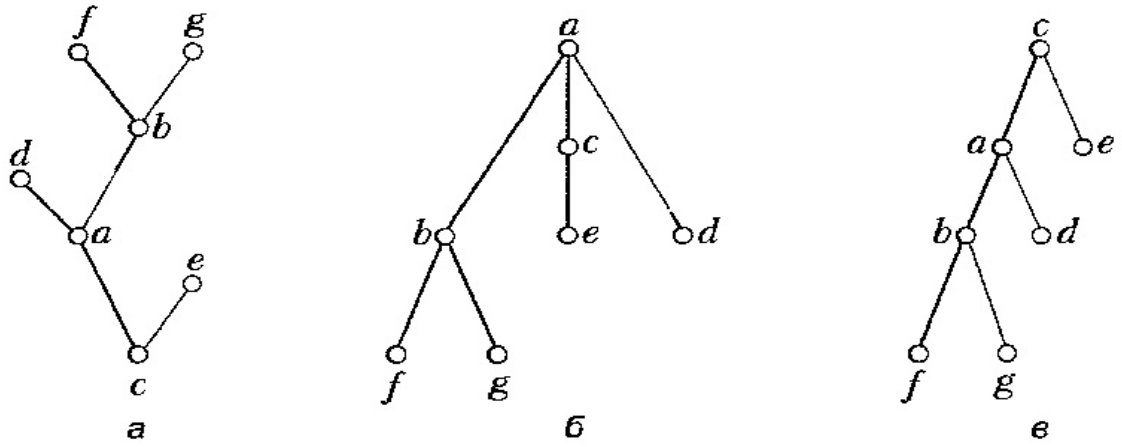

Рис. 3

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

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*.

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*. Піддерево з коренем у вершині, яка являє собою лівого сина вершини *v*, називають *лівим піддеревом у цій вершині*. Якщо корінь піддерева - правий син вершини *v*, то таке піддерево називають *правим піддеревом у цій вершині*.

Приклад 3. У дереві, зображеному на рис. 4, Л і П - відповідно ліве та праве піддерева у вершині *c*.

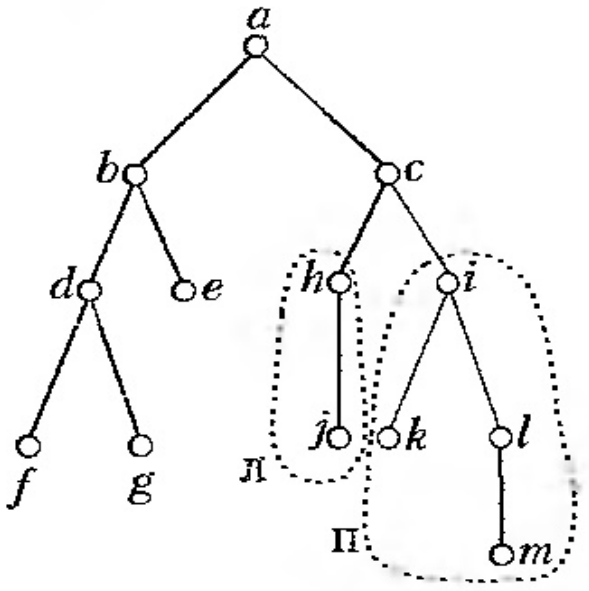

Рис. 4

Теорема 2. Повне *m*-арне дерево з *i* внутрішніми вершинами містить $n = mi + 1$ вершин.

Доведення. Кожна вершина, окрім кореня, — син внутрішньої вершини. Оскільки кожна з *i* внутрішніх вершин має *m* синів, то всього є, якщо не враховувати корінь, *mi* вершин, а з урахуванням кореня їх $mi + 1$. ■

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 *h* 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 знаходяться на рівнях *h* або *h* - 1.

Приклад 4. На рис. 5 зображено збалансоване бінарне дерево, яке має висоту 4: усі його листки на рівнях 3 та 4.

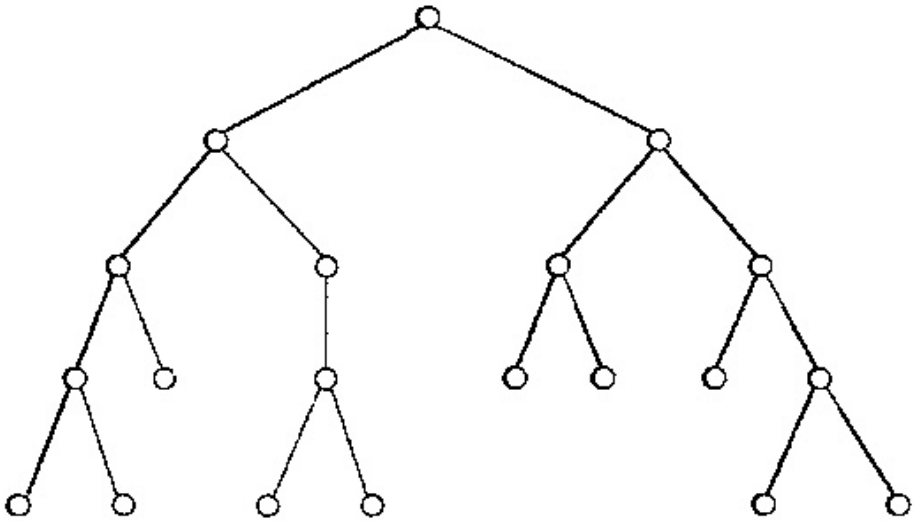

Рис. 5

Теорема 3. Нехай m-арне дерево має висоту *h*. Тоді в ньому не більше ніж m^h листків.

Доведення. Застосуємо математичну індукцію за *h*. У разі $h = 1$ твердження очевидне. Припустимо, що воно справджується для всіх *m*-арних дерев із меншою висотою, ніж *h*. Нехай *T* - *m*-арне дерево з висотою *h*. Тоді його листки - це листки піддерев, отриманих із *T* вилученням ребер, що з'єднують корінь дерева *T* з кожною вершиною рівня 1 (рис. 6).

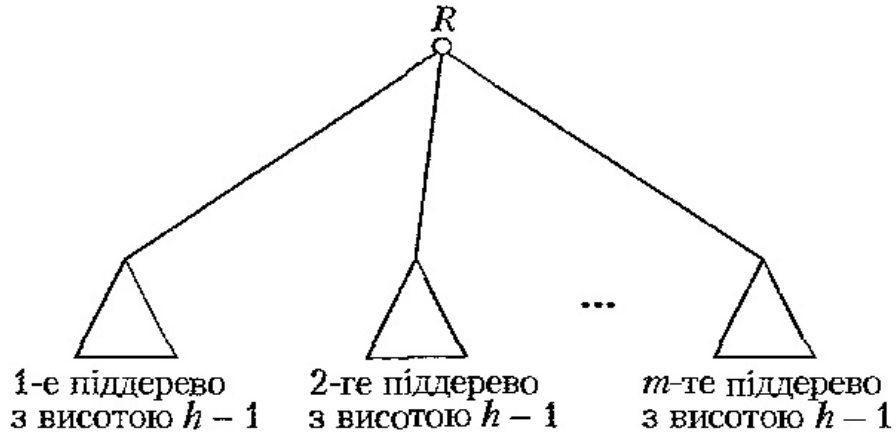

Рис. 6

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

■

Наслідок. Якщо m-арне дерево з висотою h має l листків, то $h \geq \lceil \log_m l \rceil$. Якщо m-арне дерево повне та збалансоване, то $h = \lceil \log_m l \rceil$.

Доведення. За теоремою 3 $l \leq m^h$. Прологарифмуємо цю нерівність за основою m: $\log_m l \leq h$. Оскільки h - ціле, то $h \geq \lceil \log_m l \rceil$. Тепер припустимо, що дерево повне та збалансоване. Вилучимо всі листки на рівні h (разом з інцидентними їм ребрами). Одержимо завершене m-арне дерево висотою h - 1. Воно має m^{h-1} листків. Отже, $m^{h-1} < 1 \leq m^h$. Звідси випливає, що $h-1 < \log_m l \leq h$, тобто $h = \lceil \log_m l \rceil$.

2. Рекурсія. Обхід дерев. Префіксна та постфіксна форми запису виразів

Об'єкт називають *рекурсивним*, якщо він містить сам себе чи його означено за допомогою самого себе. Рекурсія - потужний засіб у математичних означеннях.

Приклад 5. У підрозділі 1 сформульовано означення повного бінарного дерева. Тепер означимо його рекурсивно:

а) о (ізольована вершина) - *повне бінарне дерево*:

б) якщо A та B - повні бінарні дерева, то конструкція, зображена на рис. 7, - *повне бінарне дерево*.

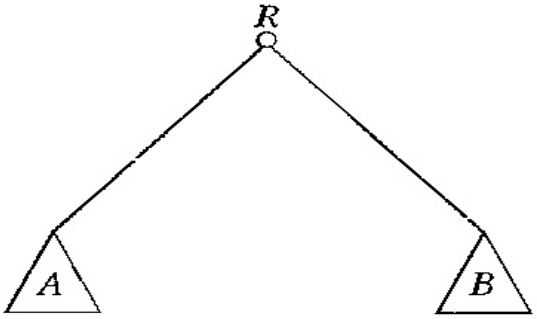

Рис. 4.7

Приклад 6. Рекурсивне означення функції $n!$ для невід'ємних цілих чисел має такий вигляд:

а) 0! = 1;

б) якщо $n > 0$, то $n! = n \times (n-1)!$.

Очевидно, що важливість рекурсії пов'язана з тим, що вона дає змогу означити нескінченну множину об'єктів за допомогою скінченного висловлювання. Так само нескінченні обчислення можна описати за допомогою скінченної рекурсивної програми, навіть якщо вона не містить явних циклів. Проте найдоцільніше використовувати рекурсивні алгоритми тоді, коли розв'язувану задачу, обчислювану функцію чи оброблювані дані задано за допомогою рекурсії.

Чимало задач можна моделювати з використанням кореневих дерев. Поширене таке загальне формулювання задачі: виконати задану операцію $\Re$ з кожною вершиною дерева. Тут $\Re$ - параметр загальнішої задачі відвідування всіх вершин, або так званого *обходу дерева*. Розглядаючи розв'язування 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. 7), можна означити такі впорядкування.

1. Обхід у *прямому порядку* (*preorder*), або *зверху вниз*: *R, A, B* (корінь відвідують до обходу піддерев).
2. Обхід у *внутрішньому порядку* (*inorder*), або *зліва направо*: *A, R, B*.
3. Обхід у *зворотному порядку* (*postorder*), або *знизу вверх*: *A, B, R* (корінь відвідують після обходу піддерев).

Приклад 7. На рис. 8 зображено бінарне дерево. Різні обходи дадуть такі послідовності вершин:

- обхід у прямому порядку: *a b d e h o c f m p g*;
- обхід у внутрішньому порядку: *d b h e o a f c p m q*;
- обхід у зворотному порядку: *d h o e b f p q m c a*.

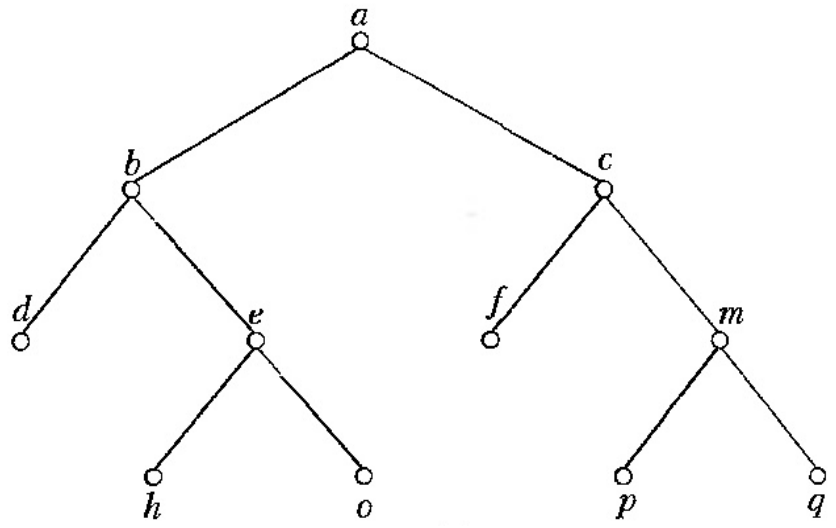

Рис. 8

Зазначені способи обходу бінарних дерев можна узагальнити й на довільні *m*-арні дерева. Обхід таких дерев у прямому порядку (зверху вниз) схематично зображено на рис. 9, у внутрішньому порядку (зліва направо) - на рис. 10, у зворотному (знизу вверх) — на рис. 11.

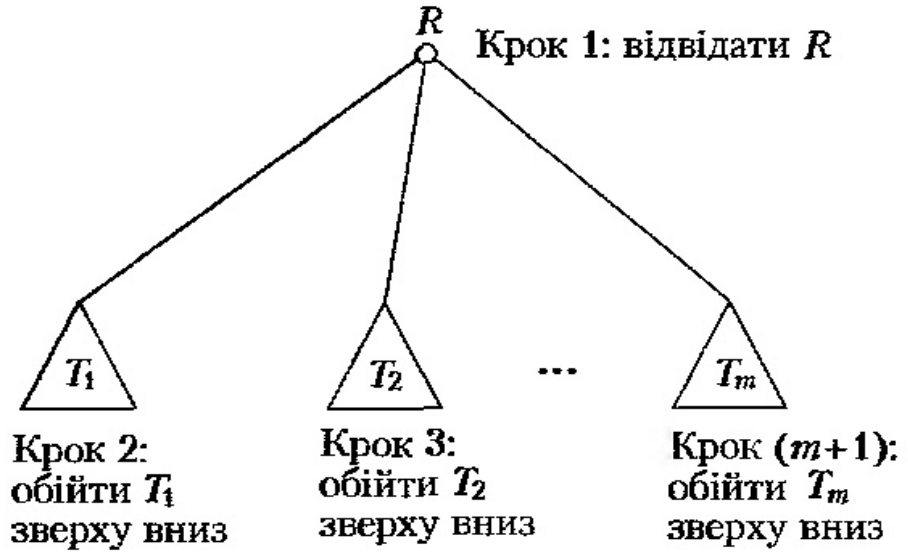

Рис. 9

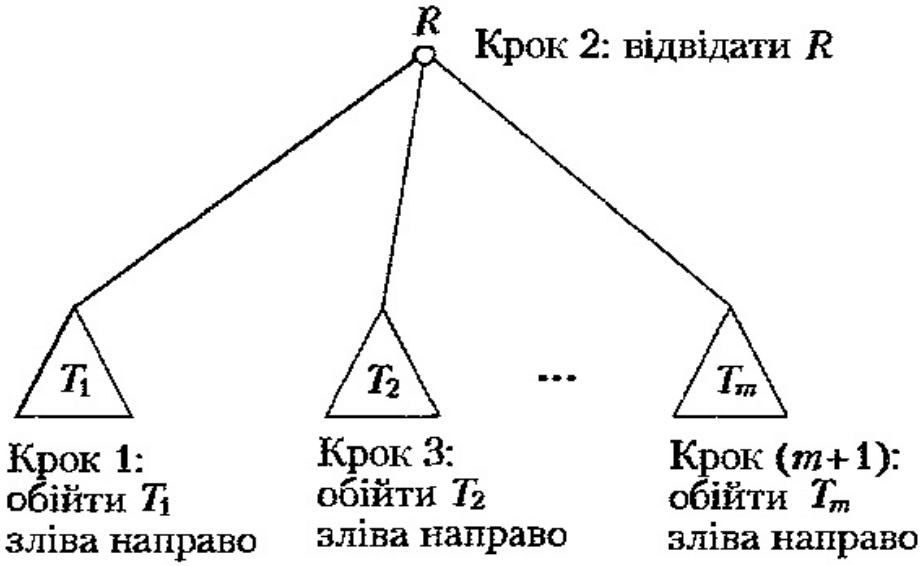

Рис. 10

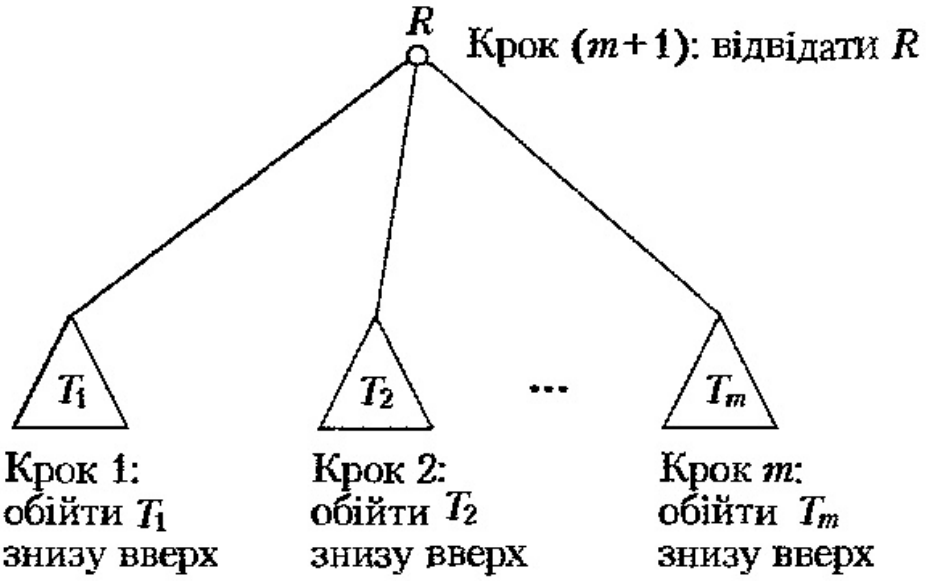

Рис. 11

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. Послідовність дій відтворено на рис. 12. Рамкою на ньому обведено дерево, яке відповідає заданому арифметичному

виразу. Внутрішнім вершинам цього дерева відповідають символи операцій, а листкам - операнди.

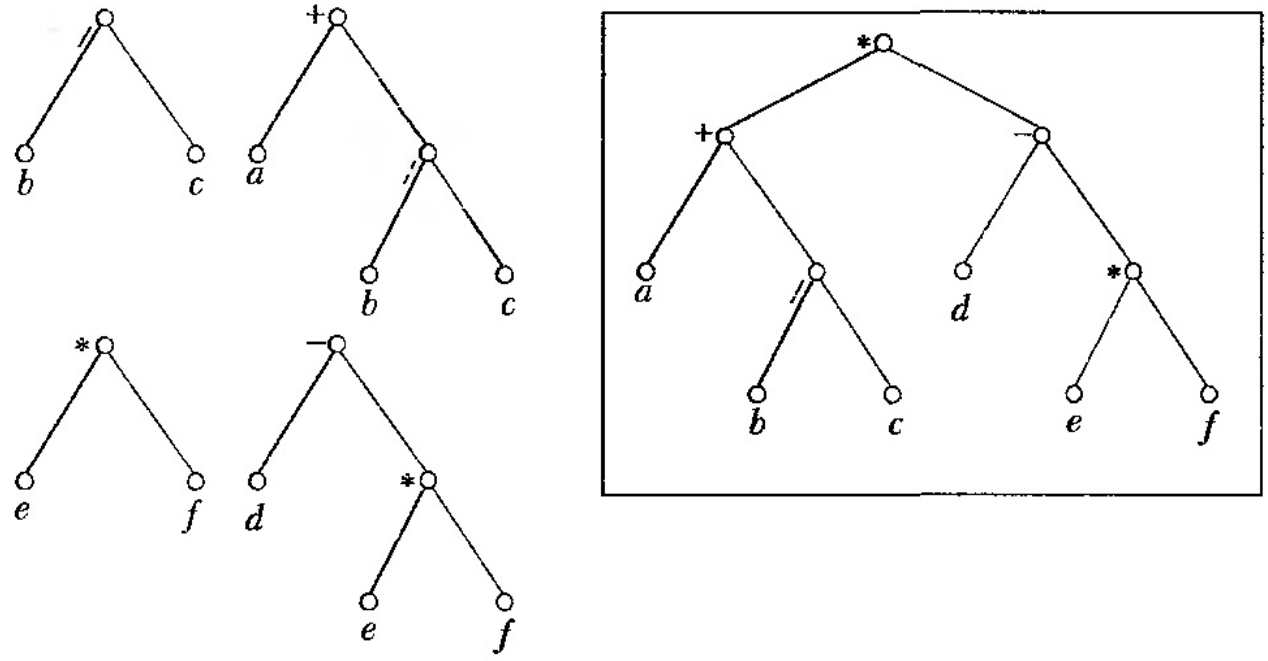

Рис. 12

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*

$$\times + a\ / b c - d \times e f\ ;$$

- у разі обходу у внутрішньому порядку — *інфіксний запис* (поки що без дужок, потрібних для визначення порядку операцій)

$$a + b\ / c \times d - e \times f\ ;$$

- у разі обходу в зворотному порядку - *постфіксний (зворотний польський) запис*

$$a b c\ / + d e f \times - \times .$$

Звернімося спочатку до інфіксної форми запису виразу. Без дужок вона неоднозначна: один запис може відповідати різним деревам. Наприклад, дереву, зображеному на рис. 13, у разі обходу зліва направо відповідає той

самий вираз $a+b/c\times d-e\times f$, що й дереву на рис. 12 (у рамці), хоча на цих рисунках зображено різні дерева. 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. 12 інфіксна форма така: $\big((a+(b/c))\times(d-(e\times f))\big)$; для дерева, зображеного на рис. 13, інфіксна форма має такий вигляд: $\Big(a+\big(\big((b/(c\times d))-e\big)\times f\big)\Big)$.

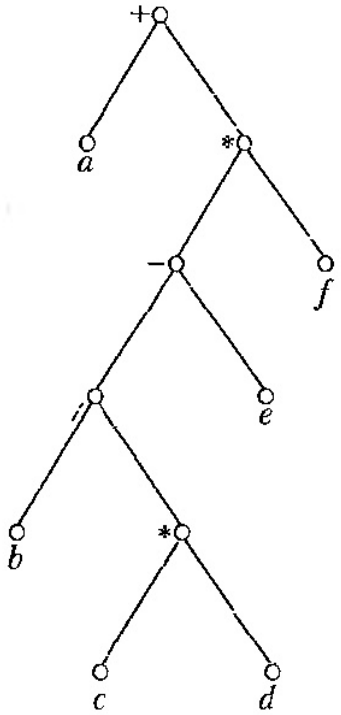

Рис. 13

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 називають *польським записом* (на честь польського математика й логіка Яна Лукасевича, українця за походженням).

Приклад 8. Розглянемо логічний вираз $\big(\neg(p\wedge q)\big)\sim(\neg p\vee\neg q)$. Послідовні етапи побудови відповідного бінарного дерева зображено на рис. 14.

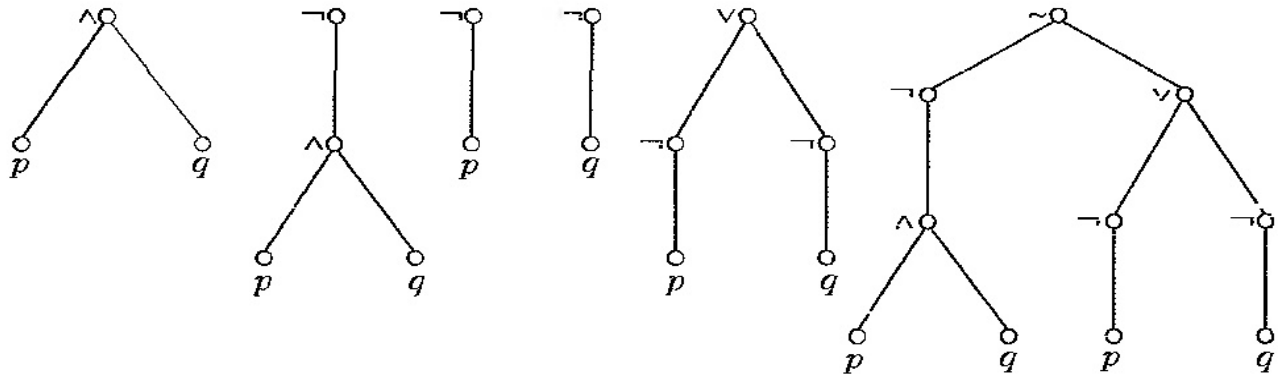

Рис. 14

Отримаємо такі форми запису виразу:

- інфіксна форма запису (відповідає обходу дерева виразу у внутрішньому порядку):

$$\big(\big((p \wedge q)\neg\big) \sim \big((p\neg) \vee (q\neg)\big)\big);$$

- польський запис (відповідає обходу дерева виразу в прямому порядку):

$$\sim \neg \wedge pq \vee \neg p \neg q;$$

- зворотний польський запис (відповідає обходу дерева виразу в зворотному порядку):

$$pq \wedge \neg p \neg q \neg \vee \sim .$$

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 вилучених символів.

Приклад 9. Обчислимо значення виразу в польському записі (^ означає піднесення до степеня)

$$+ - \times 235 / \wedge 234 .$$

За сформульованим правилом виділимо $\wedge 23$, ці символи вилучимо й обчислимо $2 \wedge 3 = 8$; результат запишемо на місце вилучених символів:

$$+ - \times 2\ 3\ 5 / 84 .$$

Продовжимо обчислення. Динаміку процесу відображено в таблиці 1.

Таблиця 1

Крок	**Вираз**	**Виділені символи**	**Виконання операцій**
1	+ - × 2 3 5 / ^ 2 3 4	^ 2 3	2 ^ 3 = 8
2	+ - × 2 3 5 / 8 4	/ 8 4	8 / 4 = 2
3	+ - × 2 3 5 2	× 2 3	2 × 3 = 6
4	+ - 6 5 2	- 6 5	6 – 5 = 3
5	+ 1 2	+ 1 2	1 + 2 = 3
6	3		

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

Приклад 10. Обчислимо значення виразу в зворотному польському записі

7 2 3 × - 4 ^ 9 3 / +.

Динаміку обчислень відображено в таблиці 2.

Таблиця 2

Крок	**Вираз**	**Виділені символи**	**Виконання операцій**
1	7 2 3 × - 4 ^ 9 3 / +	2 3 ×	2 × 3 =6
2	7 6 - 4 ^ 9 3 / +	7 6 -	7 - 6 = 1
3	1 4 ^ 9 3 / +	1 4 ^	1 ^ 4 = 1
4	1 9 3 / +	9 3 /	9 / 3 = 3
5	1 3 +	1 3 +	1 + 3 = 4
6	4		

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.