КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ І МЕХАНІКИ

ТЕМА 11. *ДЕРЕВА*.

План

Література

1. *Нікольський Ю.В., Пасічник В.В., Щербина Ю.М.* Дискретна математика. – Л.: “Магнолія – 2006”, 2016. – 432 с.

§ 1. Основні означення та властивості.

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.

Означення. *Деревом* називають зв'язний граф без простих циклів.

Означення. Граф, який не містить простих циклів і складається з k компонент, називають *лісом* із k дерев.

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

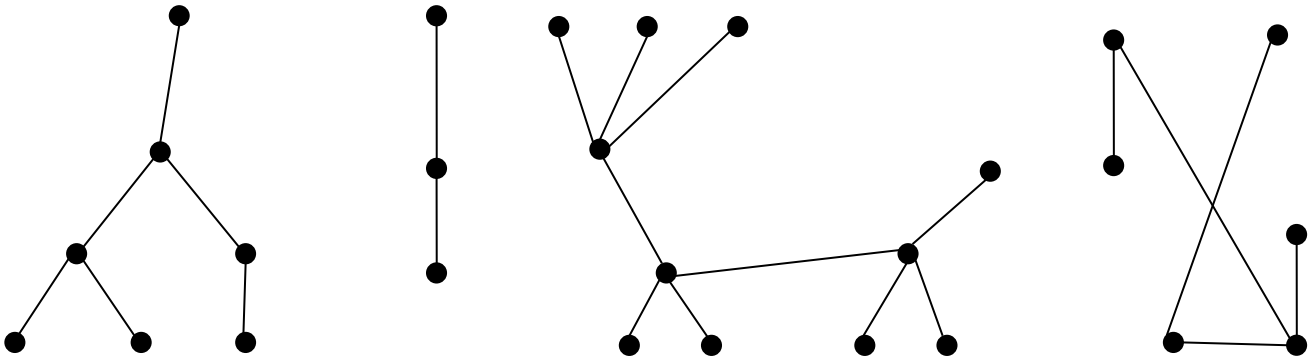


Рис. 1

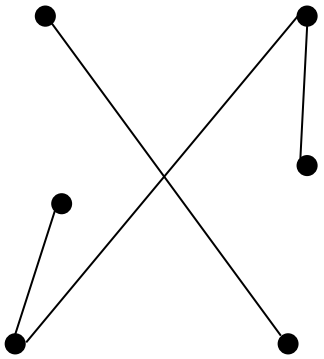


Рис. 2

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

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

I. граф T – дерево;

II. граф T не містить простих циклів і має $(n-1)$ ребро;

III. граф T зв'язний і має $(n-1)$ ребро;

IV. граф T зв'язний, але вилучення довільного ребра робить його незв'язним;

V. довільні дві вершини графа T з'єднані точно одним простим шляхом;

VI. граф T не містить простих циклів, але, додавши до нього довільне нове ребро (без додавання вершин), ми отримаємо точно один простий цикл.

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

(I) → (II). 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 T має $(n - 1)$ ребро.

(II) → (III). 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.

(III) → (IV).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 3 з лекції 14).

(IV) → (V). Оскільки граф T зв'язний, то кожну пару його вершин з'єднано принаймні одним простим шляхом (теорема 1 з лекції 14). Якщо припустити, що якусь пару вершин з'єднано двома простими шляхами, то вони замикаються в простий цикл. Але це суперечить тому, що вилучення довільного ребра робить граф T незв' язним.

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

(VI) → (I). 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 (VI).

Наслідок. Ліс із k дерев, який містить n вершин, має $(n - k)$ ребер.

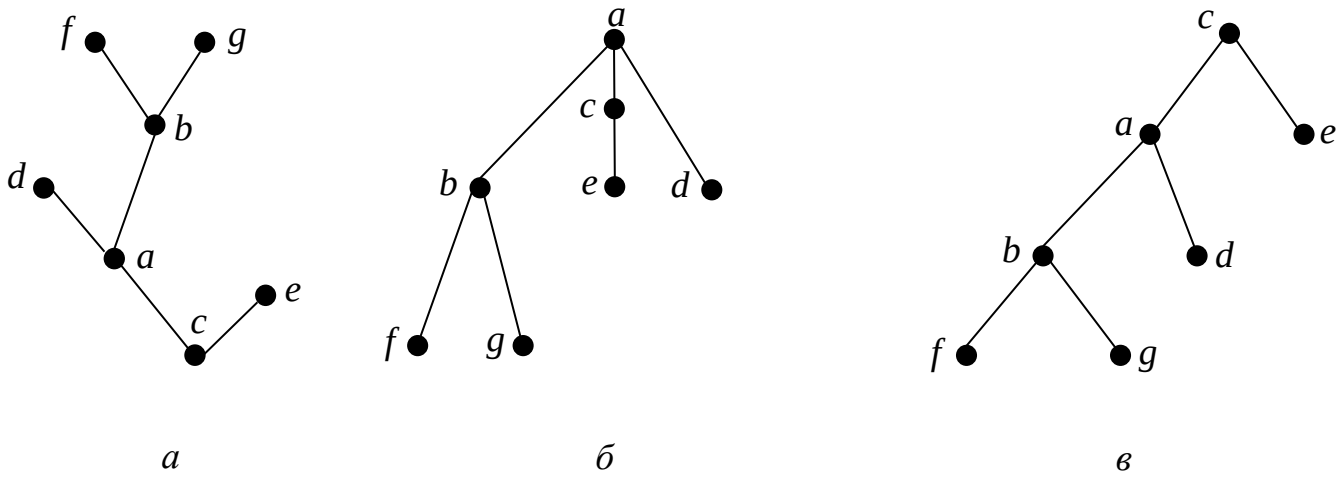
Розглянемо поняття кореневого дерево.

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.

Означення. Дерево разом із виділеним коренем утворює орієнтований граф, який називають *кореневим деревом.*

Зазначимо, що різні способи вибору кореня утворюють 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.*



а *б* *в*

Рис. 3

Означення. Нехай T – кореневе дерево. Якщо v – його вершина, відмінна від кореня, то *батько* v – це єдина вершина u така, що є орієнтоване ребро (u, v). Якщо u – батько, то v – *син.* Аналогічно за генеалогічною термінологією можна означити інших предків і нащадків вершини v.

Означення. Вершини дерева, які не мають синів, називають *листками.*

Означення. Вершини, які мають синів, називають *внутрішніми.* Зазначимо, що корінь належить до внутрішніх вершин.

Означення. Якщо a – вершина дерева, то *піддерево* з коренем a – це підграф, що містить a та всі вершини – нащадки вершини a, а також інцидентні їм ребра.

Означення. Кореневе дерево називають *m-арним,* якщо кожна його внутрішня вершина має не більше ніж m синів.

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

Означення. *Упорядковане кореневе дерево* – це кореневе дерево, у якому сини кожної внутрішньої вершини впорядковано.

На рисунку таке дерево зображають так, щоб сини кожної вершини були розміщені зліва направо.

Означення. Якщо внутрішня вершина впорядкованого бінарного дерева має двох синів, то першого називають *лівим* сином, а другого – *правим.*

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

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

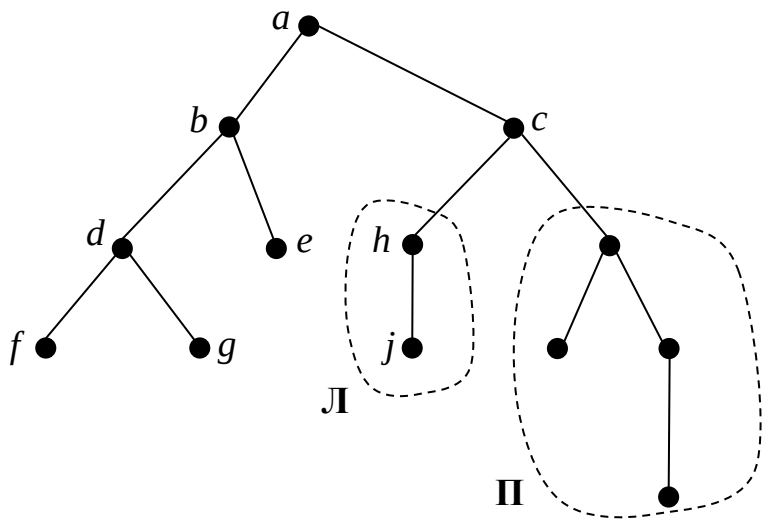


Рис. 4

Теорема 2. Повне *m*-арне дерево з *i* внутрішніми вершинами містить $n = m\,i + 1$ 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, *m i* вершин. З урахуванням кореня всього вершин $n = m\,i + 1$, що й потрібно було довести.

Означення. *Рівнем* вершини *v* в кореневому дереві називають довжину простого шляху від кореня до цієї вершини (цей шлях, очевидно, єдиний).

Рівень кореня вважають нульовим.

Означення. *Висотою* кореневого дерева називають максимальний із рівнів його вершини.

Інакше кажучи, висота кореневого дерева – це довжина найдовшого простого

шляху від кореня до будь-якої вершини.

Означення. Повне m-арне дерево, у якого всі листки на одному рівні, називають *завершеним m-арним деревом.*

Означення. Кореневе m-арне дерево з висотою h називають *збалансованим,* якщо всі його листки 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.

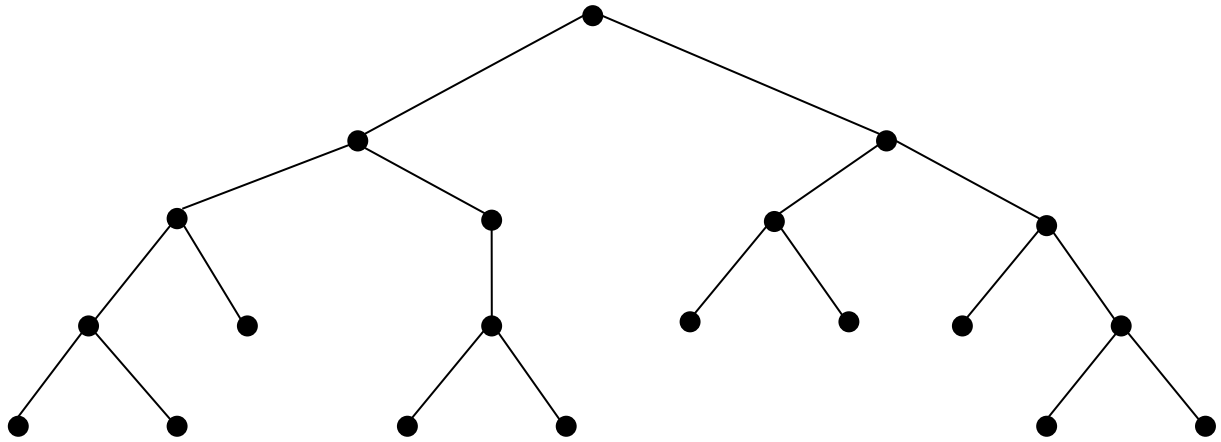
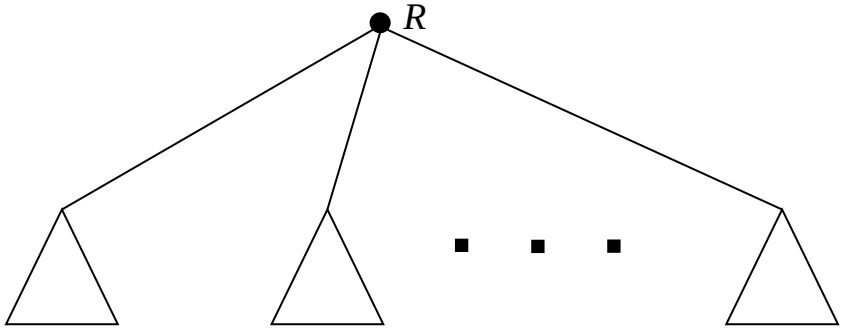


Рис. 5

Зазначимо, що є й інші означення збалансованості.

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. Листки дерева T – це листки піддерев, які отримують з T вилученням ребер, що з'єднують корінь дерева T із кожною вершиною рівня 1 (рис. 6).



1-е піддерево з висотою $h-1$ | 2-е піддерево з висотою $h-1$ | m-е піддерево з висотою $h-1$

Рис. 6

Кожне з цих піддерев має не більшу висоту, ніж $h-1$. За індуктивною гіпотезою кожне з них має не більше ніж m^{h-1} листків. Позаяк таких піддерев не більше ніж m, то загальна кількість листків у дереві T не перевищує $m \cdot m^{h-1} = m^h$. Теорему доведено.

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 $h = \lceil \log_m l \rceil$. Нагадаємо, що $\lceil x \rceil$ – це найменше ціле, яке більше чи дорівнює x.

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

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

Означення. Об’єкт називають *рекурсивним*, якщо він містить сам себе, або означений за допомогою самого себе.

Рекурсія – потужний засіб у математичних означеннях.

В попередньому параграфі було дано означення повного бінарного дерева. Дамо його означення рекурсивно:

а) (ізольована вершина) – *повне бінарне дерево*;

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

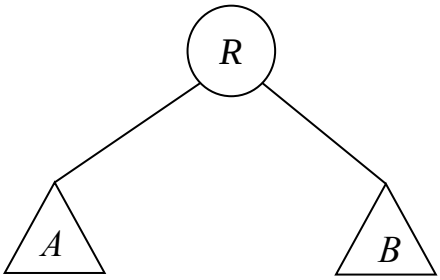


Рис. 7

Приклад 5. Рекурсивне означення функції $n!$ для невід'ємних цілих чисел має такий вигляд:

а) $0! = 1$;

б) якщо $n > 0$, то $n! = n\,(n-1)!$

Очевидно, що потужність рекурсії пов'язана з тим, що вона дозволяє означити нескінченну множину об'єктів за допомогою скінченного висловлювання. Так само нескінченні обчислення можна описати за допомогою скінченної рекурсивної програми, навіть якщо ця програма не містить явних циклів. Проте, краще за все використовувати рекурсивні алгоритми в тих випадках, коли задача, яку розв'язують, або функція, яку обчислюють, або дані, які обробляють, задано за допомогою рекурсії.

Чимало задач можна моделювати з використанням кореневих дерев. Поширене таке загальне формулювання задачі: *виконати задану операцію* ***обробити*** *з кожною вершиною дерева*. Тут *обробити* – параметр загальнішої задачі відвідування всіх вершин, або так званого *обходу дерева*. Розглядаючи розв'язування цієї задачі як єдиний послідовний процес відвідування вершини 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. Існує 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* (тобто корінь відвідують після обходу піддерев).

Нижче наведено рекурсивні алгоритми обходу бінарних дерев. У кожному способі обходу використано команду **обробити**(*v*), де *v* – вершина. Цей термін залишено невизначеним, бо його значення залежить від того, що ми хочемо зробити під час проходження вершин. Для наших цілей достатньо використати звичайну команду друку символу.

Алгоритм обходу в прямому порядку позначено як **ОПП**(*x*), у внутрішньому – як **ОВП**(*x*) і в зворотному – як **ОЗП**(*x*). У всіх трьох алгоритмах *x* – це корінь дерева, яке обходять. Лівого сина вершини *v* позначено як лс(*v*), а правого – як пс(*v*).

Алгоритм обходу дерева в прямому порядку – **ОПП**(*корінь*)

- **Обробити**(*корінь*)
- Якщо лс(*корінь*) існує, то **ОПП**(лс(*корінь*))
- Якщо пс(*корінь*) існує, то **ОПП**(пс(*корінь*))

Алгоритм обходу дерева у внутрішньому порядку – **ОВП**(*корінь*)

- Якщо лс(*корінь*) існує, то **ОВП**(лс(*корінь*))
- Обробити(*корінь*)
- Якщо пс(*корінь*) існує, то **ОВП**(пс(*корінь*))

Алгоритм обходу дерева у зворотному порядку – **ОЗП**(*корінь*)

- Якщо лс(*корінь*) існує, то **ОЗП**(лс(*корінь*))
- Якщо пс(*корінь*) існує, то **ОЗП**(пс(*корінь*))
- Обробити(*корінь*)

Приклад 6. 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 q*;

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*.

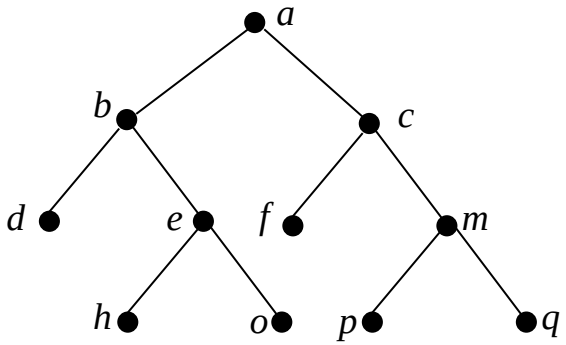
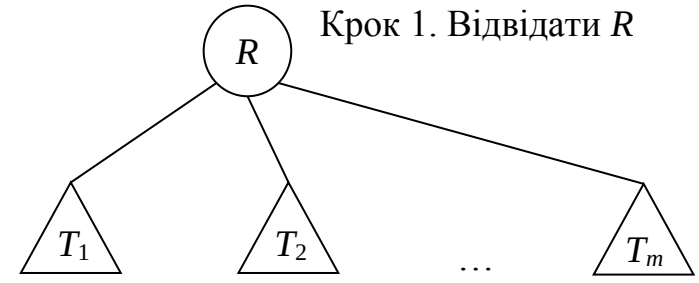


Рис. 8

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 4.11

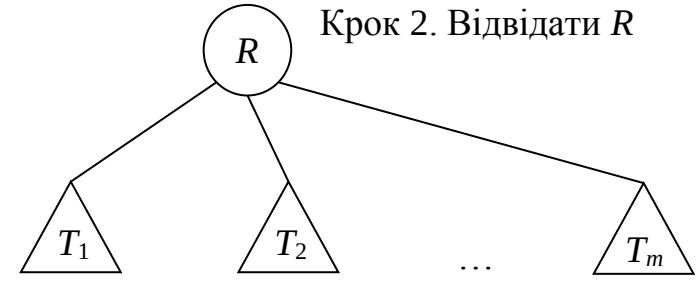


Крок 2: Відвідати T_1 зверху вниз

Крок 3: Відвідати T_2 зверху вниз

Крок (m+1): Відвідати T_m зверху вниз

Рис. 9



Крок 1: Відвідати T_1 зліва направо

Крок 3: Відвідати T_2 зліва направо

Крок (m+1): Відвідати T_m зліва направо

Рис. 10

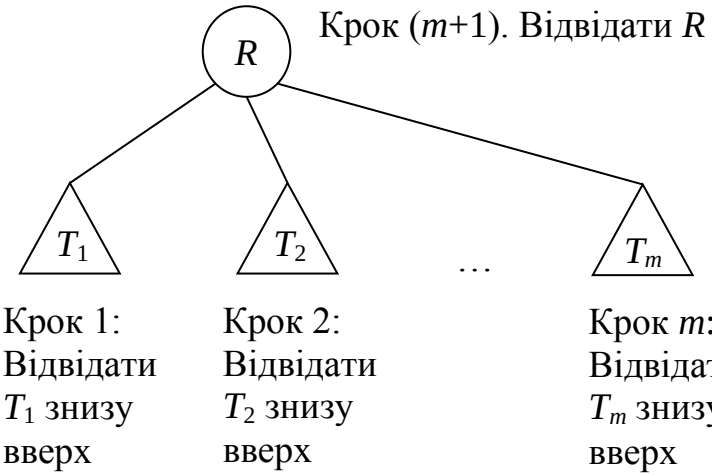


Рис. 11

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

Приклад 7. Розглянемо арифметичний вираз (зірочкою позначено операцію множення) $\left(a+\frac{b}{c}\right)*(d-e*f)$. 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.

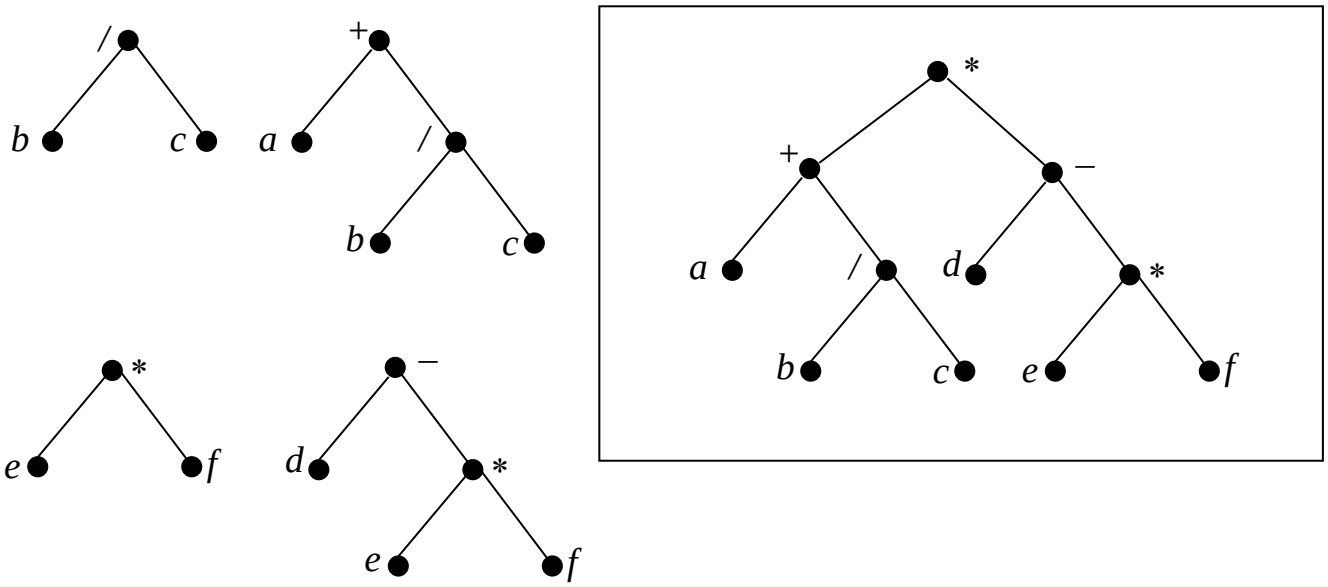


Рис. 12

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

- у разі обходу в прямому порядку – *префіксний* або *польський запис:*

$$*+a/bc-d*ef\,.$$

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

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

- у разі обходу в зворотному порядку – *постфіксний* або *зворотний польський запис:*

$$abc/+def*-*.$$

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 $a+b/c*d-e*f$, що й дереву на рис. 12 (у рамці), хоча на цих рисунках зображено різні дерева.

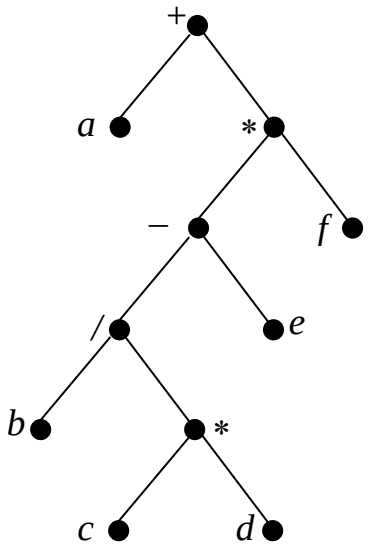


Рис. 13

Щоб уникнути неоднозначності інфіксної форми, використовують круглі дужки щоразу, коли зустрічають операцію. Повністю "одужкований" вираз, отриманий під час обходу дерева у внутрішньому порядку, називають *інфіксною формою* запису.

Отже, для дерева з рис.12 інфіксна форма така: $((a+(b/c))*(d-(e*f)))$; для дерева, зображеного на рис. 13, інфіксна форма має такий вигляд: $(a+(((b/(c*d))-e)*f))$.

Наведені міркування свідчать, що інфіксна форма запису виразів незручна. На практиці використовують префіксну та постфіксну форми запису, бо вони однозначно відповідають виразу й не потребують дужок. Ці форми запису називають *польськими*

записами (на честь польського математика й логіка Яна Лукасевича).

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

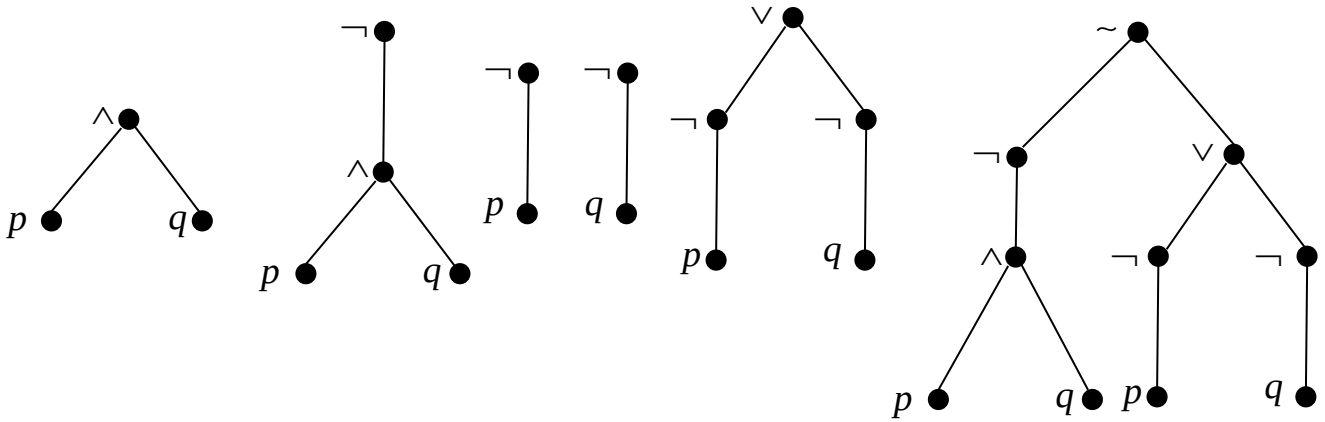


Рис. 14

Отримаємо такі форми запису виразу:

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

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

- 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 8. Обчислимо значення виразу в польському записі (стрілка означає піднесення до степеня) $+ - * 235 / \uparrow 234$.

За сформульованим правилом виділимо $\uparrow 23$, ці символи вилучимо й обчислимо $2 \uparrow 3 = 8$, результат записуємо на місце вилучених символів: $+ - * 235 / 84$.

Продовжимо обчислення. Динаміку процесу відображено в табл. 1.

Таблиця 1

Крок	Вираз	Виділені символи	Виконання операції
1	$+-*235/\uparrow 234$	$\uparrow 23$	$2\uparrow 3=8$
2	$+-*235/84$	$/84$	$8/4=2$
3	$+-*2352$	$*23$	$2*3=6$
4	$+-652$	-65	$6-5=1$
5	$+12$	$+12$	$1+2=3$
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 $723*-4\uparrow 93/+$.

Динаміку обчислень відображено в табл. 2.

Таблиця 2

Крок	Вираз	Виділені символи	Виконання операції
1	$723*-4\uparrow 93/+$	$23*$	$2*3=6$
2	$76-4\uparrow 93/+$	$76-$	$7-6=1$
3	$14\uparrow 93/+$	$14\uparrow$	$1\uparrow 4=1$
4	$193/+$	$93/$	$9/3=3$
5	$13+$	$13+$	$1+3=4$
6	4		

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

Контрольні запитання

1. Дати визначення дерева.

2. Що називають лісом з k дерев?
3. Нехай граф T має n вершин. Сформулювати еквівалентні твердження.
4. Що називають кореневим деревом?
5. Дати визначення батька, сина та листка у кореневому дереві.
6. Які вершини у кореневому дереві називають внутрішніми?
7. Дати визначення піддерева.
8. Що називають m-арним кореневим деревом та повним m-арним кореневим деревом? Дати визначення бінарного кореневого дерева.
9. Що називають упорядкованим кореневим деревом?
10. Дати визначення лівого та правого синів у бінарному дереві.
11. Дати визначення лівого та правого піддерев у певній вершині.
12. Як знайти кількість вершин у повному m-арному дереві з i внутрішніми вершинами?
13. Дати визначення рівня певної вершини у кореневому дереві.
14. Що називають висотою кореневого дерева?
15. Яке дерево називають завершеним m-арним деревом?
16. Яке кореневе m-арне дерево називають збалансованим?
17. Скільки листків у m-арному дереві з висотою h?
18. Який об’єкт називають рекурсивним?
19. Які існують три принципи впорядкування вершин у дереві?
20. Як здійснюється обхід у прямому, у внутрішньому та зворотному порядках? Зобразити схематично.
21. Як визначають інфіксну форму запису виразу та як уникнути в ній неоднозначності?
22. Як визначають префіксну форму запису виразу та як обчислюють значення виразу, що записаний у цій формі?
23. Як визначають постфіксну форму запису виразу та як обчислюють значення виразу, що записаний у цій формі?

Розробила
доцент кафедри прикладної математики і механіки
к. ф.-м. наук Чмир О.Ю.