2.7. ПРИКЛАДИ ЗАСТОСУВАННЯ ДЕРЕВ

2.7.1. Упорядковування (сортування) даних. Прошиті бінарні дерева

Дерева пошуку

Цей вид бінарного дерева дуже широко використовується, тому що він дає великі переваги при створенні структур, які постійно формуються і паралельно в них відбувається пошук даних. Принцип формування вузлів такого дерева полягає у тому, що нове дане розташовується у ліве піддерево, якщо воно менше, ніж значення кореня, і в праве, навпаки. Обхід вузлів дерева відбувається за алгоритмом ***LVR***. Розглянемо приклад дерева, що постійно зростає і ніколи не зменшується (прикладом може бути формування частотного словника). У цій задачі задано послідовність слів і необхідно визначити частоту появи кожного слова у тексті. Це означає, що кожне слово шукається у дереві. Якщо слово знайдено, лічильник його вживання збільшується на одиницю, якщо ні, то формується новий вузол: лівого піддерева, якщо новий рядок менше кореня, або правого – у протилежному випадку. В цей вузол вставляється це слово і значення лічильника стає рівним 1. Це задача пошуку по дереву з включенням. Кількість порівнянь при пошуку дорівнює log_2N. Якщо записати будь-яку послідовність даних у таке дерево, то вона буде відсортованою. На жаль, видалення даних з уже сформованого дерева пошуку набагато складніша операція, ніж вставка. Але, якщо ця операція буде зустрічатися достатньо рідко, то у структурі вузла можна передбачити поле, у якому буде відмічатися дійсність цього значення, а при пошуку даних це поле не буде враховуватися.

Продемонструємо це на прикладі набору цифрових даних. Хай маємо такий набір *<44, 19, 21, 56, 38, 66, 88, 12, 64, 14>*. Будуємо бінарне дерево таким чином: перший елемент набору є коренем, а до всіх інших членів набору застосовується таке правило: він створює ліве піддерево тієї вершини, значення якої більше, або праве піддерево тієї вершини, значення якої менше. Для наведеного вище набору побудоване бінарне дерево зображено на рис. 2.25.

Якщо таке дерево обійти способом ***LVR***, то отримаємо упорядкований за збільшенням набір *<12, 14, 19, 21, 38, 44, 56, 64, 66, 88>*, а якщо способом ***RVL***, то упорядкований за зменшенням набір *<88, 66, 64, 56, 44, 38, 21, 19, 14, 12>*.

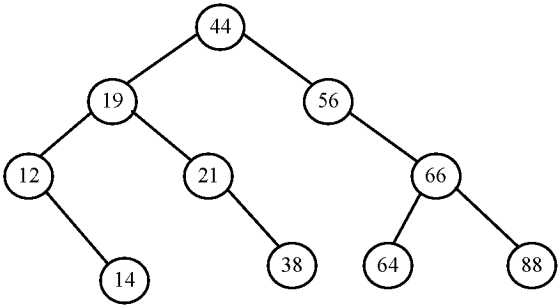

Рис. 2.25. Бінарне дерево для набору числових даних

Прошиті дерева

Бінарне дерево можна реалізувати за допомогою списку, елементи якого містять інформаційне поле (поля), та два поля з вказівниками на вершину лівого *(LPT – Left Part of Tree)* та правого *(RPT – Right Part of Tree)* піддерев (рис. 2.26).

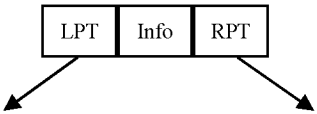

Рис. 2.26. Структура елемента списку для представлення бінарного дерева

Якщо вершина дерева не має правого чи лівого піддерева, то відповідні вказівники елемента списку мають значення `Null`. Так, список для дерева, що на рис. 2.25, буде мати такий вигляд (рис. 2.27). На рисунку значенню вказівника `Null` відповідає діагональна лінія прямокутника. Як вже наводилося, обхід дерева способом *LVR* дає упорядкований за збільшенням набір *<12, 14, 19, 21, 38, 44, 56, 64, 66, 88>*. Але фактично програма проходить деякі вершини двічі, і другий раз вони не включаються до набору за рахунок мічення пройденої вершини, якщо її значення вже було включено. Так, насправді обхід вершин буде таким *<12, 14,* ***12,*** *19, 21, 38,* ***21, 19,*** *44, 56, 64, 66, 88>*, але вершини із значеннями *12, 19* та *21* вдруге не включені до набору. Для скорочення кількості проходів по вершинах дерева «прошивають», використовуючи незадіяні поля вказівників. Так, для способу обходу *LVR* використовують незадіяні вказівники *RPT*. Замість значення `Null` вони мають показувати на вершину повернення, обминаючи пройдену вершину. Прошите дерево, що на рис. 2.27, представлено на рис. 2.28.

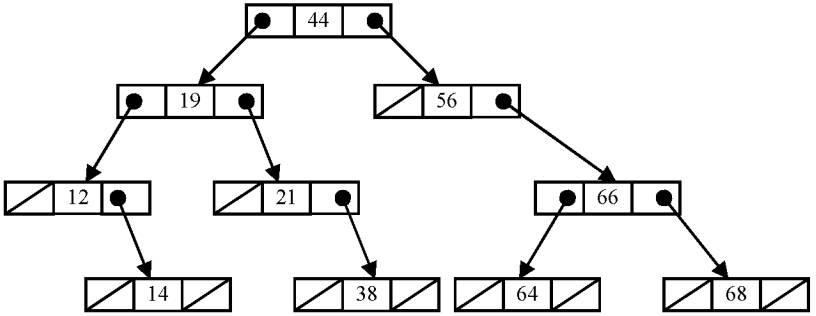

Рис. 2.27. Схема представлення дерева на рис. 2.25 з використанням списку

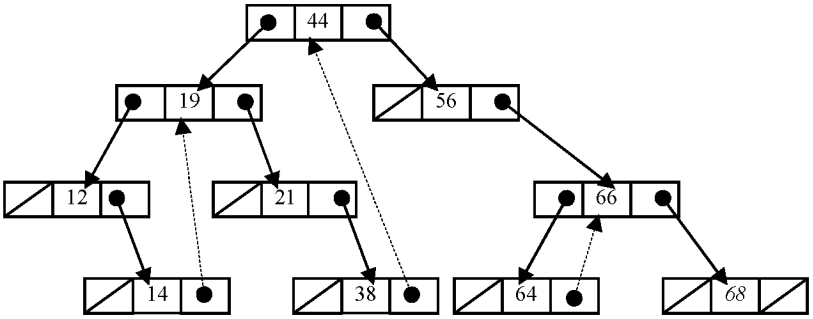

Рис. 2.28. Симетрично прошите справа дерево, що зображено на рис. 2.27

Таке дерево називається *симетрично прошитим справа*. Для універсальності алгоритму задіяні й вказівники `RPT` для вершин, після обробки якої здійснюється повернення на один рівень вище (в наведеному прикладі – із вершини із значенням `64` до вершини із значенням `66`. Крім того, крайня права вершина залишається із значенням вказівника правого піддерева `(RPT) Null`.

Аналогічно будується *симетрично прошите зліва* бінарне дерево для способу обходу *RVL*, при цьому використовуються незадіяні вказівники *LPT*. На рис. 2.29 наведено бінарне дерево, обхід якого способом *RVL* дає такий набір даних <*C, J, F, A, I, E, Y, B, D, G*>. У цьому випадку задіються всі вказівники *LPT* окрім крайньої лівої вершини.

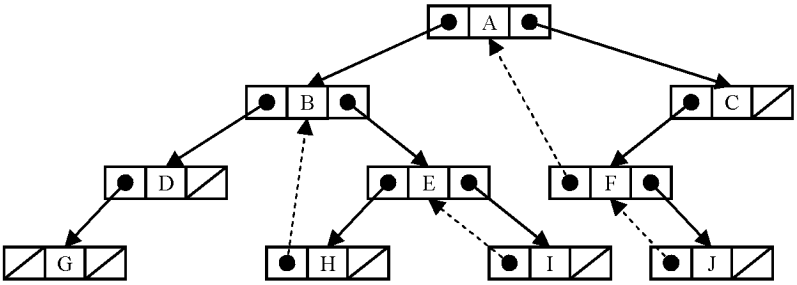

Рис. 2.29. Симетрично прошите зліва дерево

2.7.2. Конструкторсько-технологічні графи виробів

Однією із класичних задач конструкторської підготовки машинобудівного або приладобудівного виробництва є обробка специфікацій до креслень і проведення розрахунків матеріальних норм (матеріалів та комплектувальних покупних предметів) на виготовлення того чи іншого виробу. Склад виробу або будь якої складальної одиниці можна представити у вигляді конструкторсько-технологічного графа (КТГ), який представлятиме m-арне дерево. На рис. 2.30 зображено приклад такого КТГ для виробу A.

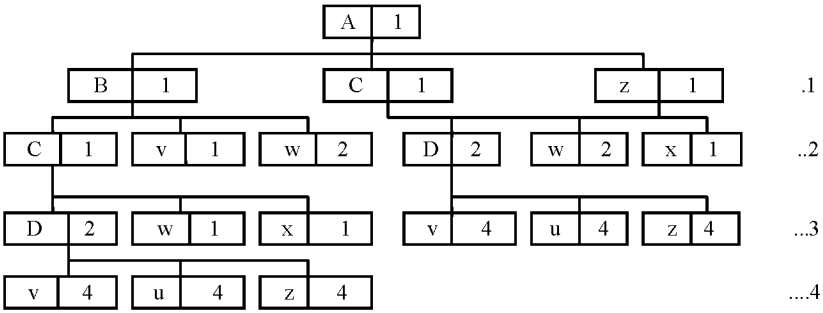

Рис. 2.30. Конструкторсько-технологічний граф виробу A

Кожна вершина КТГ, яка представлена прямокутником, має два інформаційних поля: у лівій частині прямокутника указаний код предмету, а в правій частині – кількість предметів, що входить до складу предмету, який представляє «батьківську» вершину дерева. Так, наприклад,

до складу предмета *C* входить два предмета *D*, один предмет *w* і один предмет *x*. На рисунку прийняті такі позначення предметів:

- Великими літерами початку латинської абетки позначені складальні одиниці, тобто сам виріб та вузли (агрегати, збірки тощо). Їм відповідають вершина та вузли дерева.
- Маленькими літерами кінця латинської абетки позначені деталі, яким відповідають листя дерева.
- Справа показано рівні входження предметів до виробу.

Структура дерева залежить як від конструкції предмету, так і від технологій складання виробу. Крайнім випадком є однорівневе дерево, яке відповідає кустарній технології, коли подаються всі деталі одночасно і складальник збирає їх у виріб на свій розсуд. Але при індустріальному виробництві деталі складаються у вузли, вузли в зборки більш високого рівня і так далі до готового виробу. Але КТГ виробів у вигляді дерева треба побудувати на основі специфікацій предметів, які відповідають однорівнему розвузлуванню.

Однорівневе розвузлування якогось виробу (збірки) представляє собою перелік тих компонент, що входять у нього безпосередньо, тобто на наступному нижньому рівні. По своєї суті – це специфікація виробу або проміжної збірки. Нижче у табл. 2.15 приведена специфікація збірки *B*.

Таблиця 2.15

Однорівневе розвузлування предмету *B*

Код збірки	Код компонента	Кількість
B	C	1
B	v	1
B	w	2

Після отримання всіх специфікацій та їх об'єднання в теоретико-множинному сенсі отримаємо такий набір даних (див. табл. 2.16).

Таблиця 2.16

Специфікації складених предметів з КТГ, що на рис. 2.30

Код збірки	Код компонента	Кількість
A	B	1
A	C	2
A	z	1
B	C	1
B	v	1
B	w	2
C	D	2

Закінчення таблиці 2.16

C	w	1
C	x	1
D	v	4
D	u	4
D	z	4

Таке комп'ютерне представлення специфікацій виробів дозволяє розв'язувати задачу структурного розвузлування виробів.

При *структурному розвузлуванні* перелічуються всі компоненти, які входять в даний виріб або складальну одиницю і розташовані на усіх нижчих рівнях структурного дерева даного виробу (збірки).

Нижче у таблиці 2.17 наведено результат структурного (зубчатого) розвузлування виробу *A*, при цьому рівень входження відповідає глибіні дерева.

Таблиця 2.17

Структурне розвузлування предмету *A*

Код виробу	Код предмета-компонента	Кількість компонентів на даному рівні, шт.	Кількість компонентів у виробі, шт.	Рівень входження
A	B	1	1	.1
A	C	1	1	..2
A	D	1	1	...3
A	v	4	8	4
A	u	4	8	4
A	z	4	8	4
A	w	1	1	...3
A	x	1	1	...3
A	v	1	1	..2
A	w	2	2	..2
A	C	2	2	.1
A	D	2	4	..2
A	v	4	16	...3
A	u	4	16	...3
A	z	4	16	...3
A	w	2	4	..2
A	x	1	2	..2
A	z	1	1	.1

Ця таблиця повністю відповідає дереву КТГ, що зображене на рис. 2.30. В таблиці наведено ще один атрибут – «Кількість компонентів у виробі», дані про які відсутні на дереві. Справа в тому, що ця величина дорівнює

кількості компонентів, що вказана у специфікації, помноженої на кількість самих збірок у збірці найближчого верхнього рівня, помноженої на кількість збірки цього рівня у збірці наступного верхнього рівня і т. д. аж до першого рівня. Так, до складу предмету *D* входить 4 предмета *v*, але предмет *D* входить до складу предмету *C* у кількості 2 шт., тому до складу предмету *C* через предмет *D* входить вже 4*2=8 предметів, оскільки подальші входження дорівнюють 1 шт. А загальне входження предмету *u* через ланцюжок входжень *u*←*D*←*C* дорівнює 4*2*2=16 шт.

Далі структурне розвузлування використовують для сумарного розвузлування, що знати, скільки яких предметів потрібно для виготовлення даного виробу.

При *сумарному (повному) розвузлуванні* представляється підсумковий перелік усіх компонент, що входять прямо (безпосередньо) або побічно (через проміжні збірки) до складу даної складальної одиниці. При цьому бажано указати найнижчій рівень входження компонентів. Така інформація корисна для попередньої орієнтації в терміновості виготовлення або закупівлі даного предмету. Так, якщо предмет входить до складу виробу на найнижчому рівні, то він потрібен уже на самому початку його виготовлення.

У таблиці 2.18 наведено сумарне розвузлування виробу *A*.

Таблиця 2.18

Структурне розвузлування предмету *A*

Код виробу	Код предмета-компонента	Загальна кількість компонентів, шт.	Найнижчий рівень входження
A	B	1	1
A	C	3	2
A	D	5	3
A	v	25	4
A	w	7	3
A	x	3	3
A	u	24	4
A	z	25	4

Зрозуміло, що якщо в наборі даних будуть специфікації кількох виробів, то можна розрахувати потребу в компонентах для всіх цих виробів.