ЛЕКЦІЯ 7. РЕКУРЕНТНІ СПІВВІДНОШЕННЯ

Рекурентне співвідношення – це рівняння або нерівність, яка описує функцію із використанням самої себе, але тільки з меншими аргументами. Наприклад, час роботи процедури `MergeSort` $T(n)$ в найгіршому випадку описується за допомогою наступного рекурентного співвідношення:

$$T(n) = \begin{cases} \Theta(1), & \text{якщо } n = 1; \\ 2T(n/2) + \Theta(n), & \text{якщо } n > 1. \end{cases} \qquad (7.1)$$

Рішенням цього співвідношення є функція $T(n) = \Theta(n\lg n)$.

При розгляді рекурентних співвідношень зазвичай вводяться два спрощення. По-перше, приймається, що час роботи алгоритму $T(n)$ визначається лише для цілочислових n, оскільки в більшості алгоритмів кількість вхідних елементів виражається цілим числом.

По-друге, оскільки час роботи алгоритму з вхідними даними фіксованого розміру виражається константою, то в рекурентних співвідношеннях для достатньо малих n зазвичай справедлива тотожність $T(n) = \Theta(1)$. Тому для зручності граничні умови рекурентних співвідношень, як правило, відкидаються та приймається, що для малих n час роботи алгоритму є $T(n)$ константою. Тому попереднє рекурентне співвідношення зазвичай буде записуватись як:

$$T(n) = 2T(n/2) + \Theta(n), \qquad (7.2)$$

без явного зазначення $T(n)$ для малих n.

7.1. Метод підстановки

Метод підстановки, який застосовується для рішення рекурентних співвідношень, складається з двох етапів:

1. робиться припущення про вигляд розв’язку;
2. за допомогою методу математичної індукції визначаються константи та доводиться, що рішення вірне.

Назва методу пояснює, що припущене рішення підставляється у рекурентне

рівняння. Цей метод достатньо потужний, але може застосовуватись лише коли легко можна висунути припущення щодо вигляду розв'язку.

Метод підстановки можна використовувати для визначення або верхньої, або нижньої межі рекурентного співвідношення. В якості прикладу розглянемо верхню границю рекурентного співвідношення:

$$T(n) = 2T(\lfloor n/2 \rfloor) + n, \tag{7.3}$$

яке подібне до (7.2). Ми припускаємо, що розв'язок має вигляд $T(n) = O(n\lg n)$. Метод полягає в доведенні того, що при придатному виборі константи $c > 0$ виконується нерівність $T(n) \le cn\lg n$. Почнемо з того, що припустимо справедливість цієї нерівності для $\lfloor n/2 \rfloor$, тобто що виконується співвідношення $T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$. Після підстановки даного виразу в рекурентне співвідношення отримуємо:

$$\begin{aligned} T(n) &\le 2\left(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)\right) + n \le cn\lg(n/2) + n = cn\lg n - cn\lg 2 + n = \\ &= cn\lg n - cn + n \le cn\lg n \end{aligned},$$

де остання нерівність виконується при $c \ge 1$.

Тепер, згідно методу математичної індукції, необхідно довести, що цей розв'язок справедливий для граничних умов. Зазвичай для цього достатньо показати, що граничні умови є придатною базою для доведення за індукцією. В рекурентному співвідношенні (7.3) необхідно довести, що сталу c можна обрати достатньо великою для того, щоб співвідношення $T(n) \le cn\lg n$ виконувалось й для граничних умов. Така вимога інколи призводить до проблем. Припустимо, що $T(1) = 1$ – єдина гранична умова цього співвідношення. Для $n = 1$ співвідношення $T(n) \le cn\lg n$ дає $T(n) \le c \cdot 1 \cdot \lg 1 = 0$, що суперечить припущенню $T(1) = 1$. Відповідно, даний базис індукції нашого доведення не виконується.

Цю складність, яка виникає при доведенні припущення індукції для вказаної граничної умови, легко обійти. Наприклад, в рекурентному співвідношенні (7.3) можна використовувати переваги асимптотичних позначень, які вимагають довести нерівність $T(n) \le cn\lg n$ тільки для великих n ($n \ge n_0$, де n_0 – константа). Це дозволяє в доведенні за методом математичної індукції не враховувати

граничну умову $T(1) = 1$. Так, при $n > 3$ наведене рекурентне співвідношення явним чином від $T(1)$ не залежить. Таким чином, обравши $n_0 = 2$, в якості бази індукції можна розглядати не $T(1)$, а $T(2)$ та $T(3)$. З рекурентного співвідношення слідує, що $T(2) = 4$, а $T(3) = 5$. Тепер доведення за математичною індукцією співвідношення $T(n) \le cn\lg n$ для деякої константи $c \ge 1$ можна завершити, обравши її достатньо великою для того, щоб були справедливі нерівності $T(2) \le 2c\lg 2$ та $T(3) \le 3c\lg 3$. Для цього достатньо обрати $c \ge 2$. В більшості рекурентних співвідношень легко розширити граничні умови таким чином, щоб припущення індукції виявилось вірним для малих n.

На жаль, не існує єдиного рецепту як вгадати розв'язок рекурентного співвідношення. Для цього потрібний досвід, вдача та творче мислення. Проте існують певні евристичні прийоми, які можуть допомогти зробити правильну здогадку.

Якщо рекурентне співвідношення подібне до того, що ми щойно розглянули, то можна припустити, що розв'язки цих співвідношень будуть подібними. Наприклад, розглянемо рекурентне співвідношення:

$$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n,$$

яке виглядає більш складним, адже в аргументі функції T правої частини доданий доданок 17. Але інтуїтивно зрозуміло, що цей доданок не може вплинути на асимптотичну поведінку рішення. При достатньо великому n різниця між $T(\lfloor n/2 \rfloor)$ та $T(\lfloor n/2 \rfloor + 17)$ стає несуттєвою: обидва ці числа приблизно рівні половині числа n. Відповідно, можна припустити, що $T(n) = O(n\lg n)$ і в цьому випадку, та за допомогою методу підстановки перевірити це припущення.

Інший спосіб знайти розв'язок – зробити грубу оцінку верхньої та нижньої межі, а потім зменшити невизначеність до мінімуму. Наприклад, в рекурентному співвідношенні (7.3) в якості початкової нижньої межі можна було б обрати $T(n) = \Omega(n)$, оскільки в ньому присутній доданок n; можна також показати, що грубою верхньою межею буде $T(n) = O(n^2)$. Далі верхня межа поступово знижується, а нижня – піднімається доки не буде отримана правильна асимптотична поведінка

рішення $T(n) = O(n\lg n)$.

Інколи за допомогою простих алгебраїчних перетворень можна звести рекурентне співвідношення до вже відомого випадку. Наприклад, наступне співвідношення:

$$T(n) = 2T\left(\left\lfloor \sqrt{n} \right\rfloor\right) + \lg n$$

виглядає досить складним. Проте його можна спростити, виконавши заміну змінних. Для зручності ми не будемо турбуватись про округлення таких значень, як $\sqrt{n}$, до цілих чисел. Використаємо заміну $m = \lg n$:

$$T\left(2^m\right) = 2T\left(2^{m/2}\right) + m.$$

Тепер можна перейменувати функцію $S(m) = T\left(2^m\right)$, після чого отримується нове співвідношення:

$$S(m) = 2S(m/2) + m,$$

яке схоже на (7.3). Це рекурентне співвідношення має такий самий розв'язок – $S(m) = O(m\lg m)$. Зробивши обернену заміну $S(m)$ на $T(n)$, отримуємо:

$$T(n) = T\left(2^m\right) = S(m) = O(m\lg m) = O(\lg n \lg\lg n).$$

7.2. Метод дерев рекурсії

Як зазначалось, недоліком методу підстановки є необхідність робити припущення щодо розв'язку рекурентного співвідношення. Метод дерев рекурсії – прямий шлях до вдалого припущення. В дереві рекурсії кожний вузол представляє час, необхідний для виконання окремо взятої підзадачі, яка розв'язується при одному з багатьох рекурсивних викликах функції. Далі значення часу роботи окремих етапів підсумовується в межах кожного рівня, а потім – по всім рівням дерева, в результаті чого отримуємо повний час роботи алгоритму.

Дерева рекурсії краще за все підходять для того, щоб допомогти зробити припущення про вигляд розв'язку, який потім перевіряється методом підстановок. При цьому у припущенні часто дозволяються наявність невеликих неточностей,

оскільки воно потім все одно перевіряється. Якщо ж побудова дерева рекурсії та підсумування часу роботи по всім складовим відбувається достатньо ретельно, то саме дерево рекурсії може стати джерелом доведення коректності розв'язку.

Наприклад, розглянемо, як за допомогою дерева рекурсії можна здогадатись про вигляд розв'язку рекурентного співвідношення $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$. Почнемо з того, що оцінимо розв'язок згори. Враховуючи те, що округлення до цілої частини аргументів функцій є несуттєвим, то побудуємо дерево рекурсії для рекурентного співвідношення $T(n) = 3T(n/4) + cn^2$, де $c > 0$.

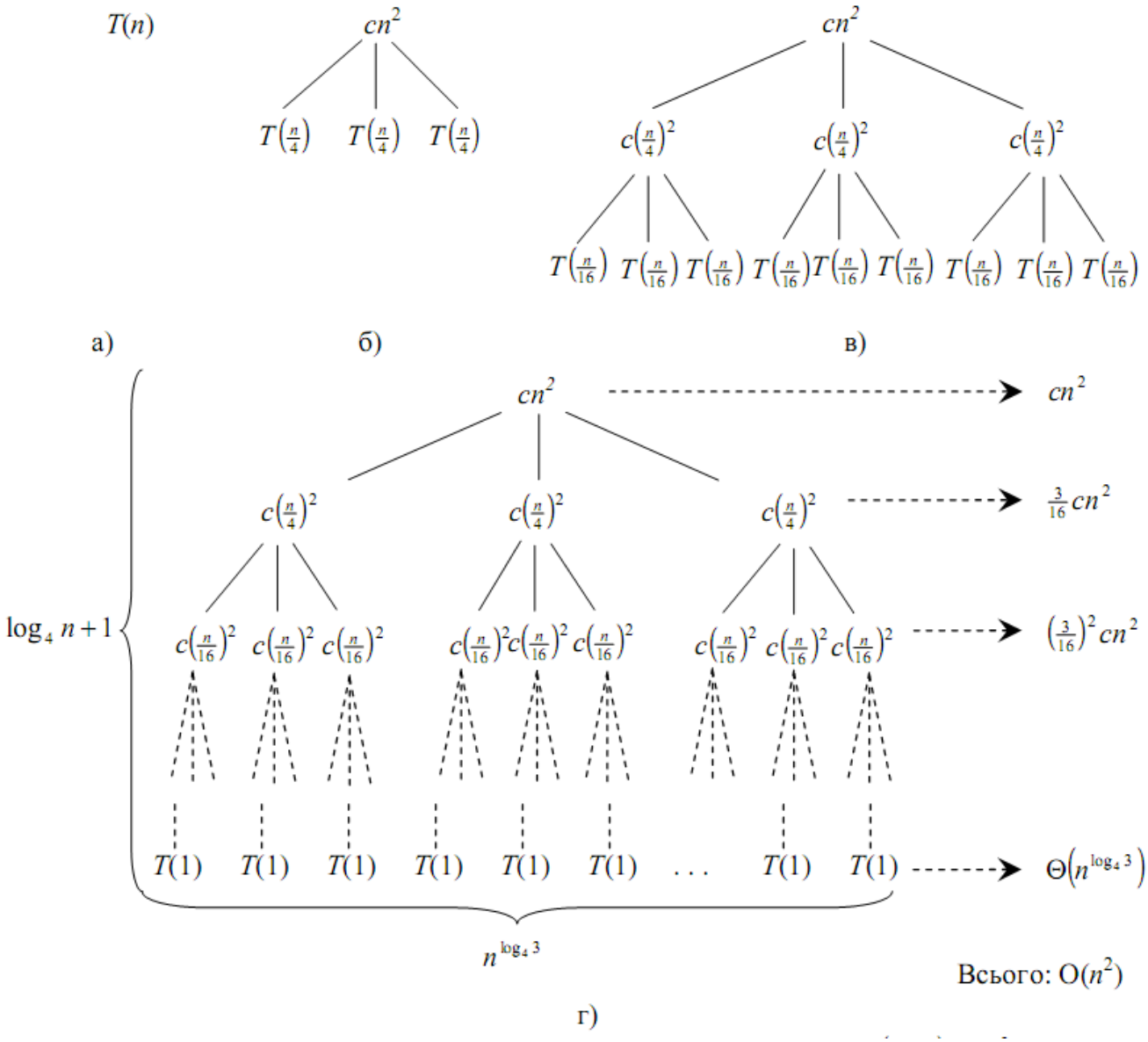

Рис. 7.1. Побудова дерева рекурсії для рівняння $T(n) = 3T(n/4) + cn^2$

На рис. 7.1 продемонстрований процес побудови дерева рекурсії для

рекурентного співвідношення $T(n) = 3T(n/4) + cn^2$. Для зручності вважатимемо, що n – ступінь четвірки. В частині *а* показана функція $T(n)$, яка потім все більш детально розкривається в частинах *б – г* у вигляді еквівалентного дерева рекурсії. Член cn^2 в корені дерева представляє час верхнього рівня рекурсії, а три піддерева, які починаються в корені, – час виконання підзадач розміром $n/4$.

По мірі віддалення від кореня розмір підзадач зменшується, зрештою ми дійдемо до граничних умов. Скільки рівнів дерева потрібно побудувати, щоб їх досягнути? Розмір допоміжної задачі, як відповідає рівню i, відповідає $n/4^i$. Таким чином, розмір підзадачі стає рівним 1, коли $n/4^i = 1$ або, що теж саме, коли $i = \log_4 n$. Отже, в дереві всього $\log_4 n + 1$ рівнів.

Далі необхідно визначити час виконання кожного рівня дерева. На кожному рівні в три рази більше вузлів, ніж на попередньому, тому кількість вузлів на рівні i дорівнює 3^i. Оскільки розміри допоміжних підзадач при спуску на один рівень зменшуються в чотири рази, час виконання кожного вузла на рівні i ($i = \overline{0, \log_4 n - 1}$) дорівнює $c\left(n/4^i\right)^2$.

Помноживши кількість вузлів на рівні на час виконання одного вузла, отримуємо, що сумарний час виконання допоміжних підзадач, які відповідають вузлам рівня i, дорівнює $3^i c\left(n/4^i\right)^2 = \left(3/16\right)^i cn^2$. Останній рівень, який знаходиться на глибині $\log_4 n$ має $3^{\log_4 n} = n^{\log_4 3}$ вузлів, кожний з яких дає у загальний час роботи внесок $T(1)$. Тому час роботи цього рівня дорівнює величині $n^{\log_4 3} T(1)$, яка поводить себе як $\Theta\left(n^{\log_4 3}\right)$.

Тепер необхідно підсумувати час роботи всіх рівнів дерева та визначити загальний час роботи дерева:

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \ldots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta\left(n^{\log_4 3}\right) =$$

$$= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) = \frac{\left(3/16\right)^{\log_4 n} - 1}{\left(3/16\right) - 1} cn^2 + \Theta\left(n^{\log_4 3}\right)$$

.

Враховуючи, що ми маємо справу в цій формулі із геометричною

прогресією, знаменник якої менше одиниці, а отже прогресія спадає. Таким чином, ми можемо обмежити згори суму у формулі сумою нескінченної спадаючої геометричної прогресії:

$$T(n) = \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) < \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) =$$

$$= \frac{1}{1-(3/16)} cn^2 + \Theta\left(n^{\log_4 3}\right) = \frac{16}{3} cn^2 + \Theta\left(n^{\log_4 3}\right) = \mathrm{O}\left(n^2\right)$$

Отже, для початкового рекурентного співвідношення $T(n) = 3T(n/4) + cn^2$ отримане припущення щодо розв'язку у вигляді $T(n) = \mathrm{O}(n^2)$. Коефіцієнти при cn^2 утворюють геометричну прогресію, а їх сума, обмежена згори константою 16/13. Оскільки вклад кореня в повний час роботи дерева становить cn^2, то час роботи кореня є деякою сталою частиною від загального часу роботи дерева в цілому. Іншими словами, повний час роботи дерева здебільшого визначається часом роботи його кореня.

Насправді, оцінка $\mathrm{O}(n^2)$ повинна бути також асимптотично точною. Адже перший рекурсивний виклик дає внесок в загальний час роботи алгоритму, який виражається як $\Theta(n^2)$, тому нижня границя часу роботи дерева також буде $\Omega(n^2)$.

Тепер за допомогою методу підстановок перевіримо коректність нашого припущення. Необхідно показати, що для деякої константи $d > 0$ виконується нерівність $T(n) \le dn^2$. Використовуючи сталу $c > 0$, отримуємо:

$$T(n) \le 3T\left(\lfloor n/4 \rfloor\right) + cn^2 \le 3d\lfloor n/4 \rfloor^2 + cn^2 \le 3d\left(n/4\right)^2 + cn^2 = \frac{3}{16} dn^2 + cn^2 \le dn^2,$$

де остання нерівність виконується при $d \ge (16/13)c$.

7.3. Основний метод

Основний метод є свого роду „збіркою рецептів", за якими будується розв'язок рекурентних співвідношень вигляду

$$T(n) = aT(n/b) + f(n), \tag{7.4}$$

де $a \ge 1$ та $b > 1$ – константи, а $f(n)$ – асимптотично додатна функція. Метод

складається лише з трьох випадків, які легко запам'ятати та застосовувати на практиці для аналізу рекурентних співвідношень.

Рекурентне співвідношення (7.4) описує час роботи алгоритму, в якому задача розміру n розбивається на a допоміжних підзадач, розміром n/b кожна. Отримані в результаті розбиття a підзадач розв'язуються рекурсивним методом, причому час їх розв'язку становить $T(n/b)$. Час, необхідний на розбиття задач та об'єднання результатів цих підзадач, описується функцією $f(n)$. Наприклад, для рекурентного рівняння, яке виникає при аналізі процедури `MergeSort`, $a = 2$, $b = 2$, а $f(n) = \Theta(n)$.

Строго кажучи, при визначенні наведеного вище рекурентного співвідношення допущена неточність, оскільки число n/b може не бути цілим. Однак заміна кожного з a доданків $T(n/b)$ виразом $T\left(\lfloor n/b \rfloor\right)$ або $T\left(\lceil n/b \rceil\right)$ не впливає на асимптотичну поведінку рішення.

Основний метод ґрунтується на наступній теоремі.

Теорема 7.1 (Основна теорема). Нехай $a \geq 1$ та $b > 1$ – константи, $f(n)$ – довільна функція, а $T(n)$ – функція, визначена на множні невід'ємних цілих чисел за допомогою рекурентного співвідношення $T(n) = aT(n/b) + f(n)$, де вираз n/b інтерпретується або як $\lfloor n/b \rfloor$, або як $\lceil n/b \rceil$. Тоді асимптотичну поведінку функції $T(n)$ можна виразити наступним чином.

1. Якщо $f(n) = \mathrm{O}\left(n^{\log_b a - \varepsilon}\right)$ для деякої сталої $\varepsilon > 0$, то $T(n) = \Theta\left(n^{\log_b a}\right)$.
2. Якщо $f(n) = \Theta\left(n^{\log_b a}\right)$, то $T(n) = \Theta\left(n^{\log_b a} \lg n\right)$.
3. Якщо $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ для деякої сталої $\varepsilon > 0$, і якщо $af(n/b) \leq cf(n)$ для деякої сталої $c < 1$ і всіх достатньо великих n, то $T(n) = \Theta\left(f(n)\right)$.

Ідея основної теореми полягає в наступному. В кожному з трьох означених випадків функція $f(n)$ порівнюється з функцією $n^{\log_b a}$. Інтуїтивно зрозуміло, що асимптотична поведінка розв'язку рекурентного співвідношення визначається більшою з двох функцій. Якщо більшою є функція $n^{\log_b a}$ (випадок 1), то розв'язок –

$T(n) = \Theta\left(n^{\log_b a}\right)$. Якщо швидше зростає функція $f(n)$ (випадок 3), то розв'язок – $T(n) = \Theta(f(n))$. Якщо ж обидві функції зрівнянні (випадок 2), то відбувається множення на логарифмічний додаток і розв'язок – $T(n) = \Theta\left(n^{\log_b a} \lg n\right)$.

В першому випадку функція $f(n)$ повинна бути не просто менше функції $n^{\log_b a}$, а поліноміально менше. Це означає, що функція $f(n)$ повинна бути асимптотично менше функції $n^{\log_b a}$ в n^{ε} разів. Аналогічно, в третьому випадку функція $f(n)$ повинна бути поліноміально більшою, а окрім того задовольняти умові регулярності: $af(n/b) \le cf(n)$. Ця умова виконується для більшості поліноміально обмежених функцій.

Важливо розуміти, що цими трьома випадками поведінка функції $f(n)$ не обмежується. Між випадками 1 та 2 є проміжок, в якому функція $f(n)$ менше за $n^{\log_b a}$, але не поліноміально менше. Аналогічний проміжок є між випадками 2 та 3, коли функція $f(n)$ більше $n^{\log_b a}$, але не поліноміально більше. Якщо функція $f(n)$ потрапляє в один з цих проміжків, або якщо для неї не виконуються умови регулярності випадку 3, основний метод не можна застосовувати для рекурентних співвідношень.

Приклад 1. Для прикладу розглянемо наступне рекурентне співвідношення:

$$T(n) = 9T(n/3) + n.$$

В цьому випадку $a = 9$, $b = 3$, а $f(n) = n$, так що $n^{\log_b a} = n^{\log_3 9} = \Theta\left(n^2\right)$. Оскільки $f(n) = \mathrm{O}\left(n^{\log_3 9 - \varepsilon}\right)$, де $\varepsilon = 1$, можна застосувати *випадок 1* основної теореми і зробити висновок, що $T(n) = \Theta\left(n^2\right)$.

Приклад 2. Тепер розглянемо наступне співвідношення:

$$T(n) = T(2n/3) + 1,$$

в якому $a = 1$, $b = 3/2$, $f(n) = 1$, $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$. Тут виконується *випадок 2*, адже $f(n) = \Theta\left(n^{\log_b a}\right) = \Theta(1)$, тому $T(n) = \Theta(\lg n)$.

Приклад 3. Для рекурентного співвідношення

$$T(n) = 3T(n/4) + n\lg n$$

виконуються умови $a = 3$, $b = 4$, $f(n) = n\lg n$, $n^{\log_b a} = n^{\log_4 3} = \mathrm{O}\left(n^{0,793}\right)$. Оскільки $f(n) = \Omega\left(n^{\log_4 3+\varepsilon}\right)$, де $\varepsilon \approx 0,2$, застосовується *випадок 3* (якщо вдасться показати виконання умов регулярності для функції $f(n)$). При достатньо великих n умова $af(n/b) = 3(n/4)\lg(n/4) \le (3/4)n\lg n = cf(n)$ виконується при $c = 3/4$. Відповідно умові 3, розв'язком цього співвідношення буде $T(n) = \Theta(n\lg n)$.

Приклад 4. До рекурентного співвідношення

$$T(n) = 2T(n/2) + n\lg n$$

основний метод не можна застосовувати, хоч воно й має потрібний вигляд: $a = 2$, $b = 2$, $f(n) = n\lg n$, $n^{\log_b a} = n^{\log_2 2} = n$. Може здатись, що до нього застосовується *випадок 3*, адже функція $f(n) = n\lg n$ асимптотично більша за $n^{\log_b a} = n$. Однак, проблема полягає в тому, що дана функція *не поліноміально* більша. Відношення $f(n)/n^{\log_b a} = (n\lg n)/n = \lg n$ асимптотично менше функції n^{ε} для будь-якого $\varepsilon > 0$. Відповідно, дане рекурентне співвідношення знаходиться в „проміжному стані" між випадками 2 та 3.

Приклад 5. Тепер застосуємо основний метод до визначення часу роботи алгоритмів, які розглядались у темі 3 в рамках парадигми „розділяй та володарюй". Для методу сортування злиттям рекурентне співвідношення має вигляд

$$T(n) = 2T(n/2) + n,$$

де $a = 2$, $b = 2$, $f(n) = n$, $n^{\log_b a} = n^{\log_2 2} = n$. Тут виконується *випадок 2*, адже $f(n) = \Theta\left(n^{\log_b a}\right) = \Theta(n)$, тому $T(n) = \Theta(n\lg n)$ (як ми й пересвідчились в цьому раніше за допомогою дерев рекурсії).

Приклад 6. Для методу Штрассена добутку матриць рекурентне співвідношення має вигляд

$$T(n) = 7T(n/2) + \Theta\left(n^2\right),$$

де $a = 7$, $b = 2$, $f(n) = n^2$, $n^{\log_b a} = n^{\log_2 7}$. Тут виконується *випадок 1*, оскільки

$f(n) = O\left(n^{\log_2 7-\varepsilon}\right)$, для $\varepsilon < 0{,}8$. Тому $T(n) = \Theta\left(n^{\log_2 7}\right)$.

7.4. Доведення основної теореми

Нижче наводиться доведення основної теореми для аналізу рекурентного співвідношення (7.4) з припущенням, що функція $T(n)$ визначена тільки для точних степенів числа $b > 1$, тобто $n = 1, b, b^2, \ldots$ Для точного доведення у випадку дробових ступенів b відсилаємо до джерел літератури, наведених вкінці теми.

Аналіз відбувається шляхом доведення трьох допоміжних теорем. В першій з них розв'язок рекурентного співвідношення зводиться до обрахунку виразу, який містить додавання. У другій теоремі визначаються межі цієї суми. В третій за допомогою перших двох доводиться версія основної теореми для випадку, коли n – точна степінь b.

Теорема 7.2. Нехай $a \geq 1$ та $b > 1$ – константи, $f(n)$ – невід'ємна функція, визначена для точних степенів числа b. Визначимо функцію $T(n)$ на множині точних степенів числа b за допомогою наступного рекурентного співвідношення:

$$T(n) = \begin{cases} \Theta(1), & \text{якщо } n = 1; \\ aT(n/b) + f(n), & \text{якщо } n = b^i. \end{cases},$$

де i – додатне ціле число. Тоді отримуємо:

$$T(n) = \Theta\left(n^{\log_b a}\right) + \sum_{j=0}^{\log_b n - 1} a^j f\left(n / b^j\right). \qquad (7.5)$$

Доведення. Скористаємось деревом рекурсії, яке представлено на рис. 7.2. Час виконання, який відповідає кореню дерева, дорівнює $f(n)$. Корінь має a дочірніх гілок, час виконання кожної з яких дорівнює $f(n/b)$. Кожна з цих дочірніх гілок, в свою чергу, також має a дочірніх гілок, час виконання яких дорівнює $f(n/b^2)$. Таким чином, отримуємо, що на другому рівні дерева є a^2 вузлів. Узагальнюючи ці міркування, можна сказати, що на j-му рівні знаходиться a^j вузлів, а час виконання кожного з них дорівнює $f(n/b^j)$. Час виконання кожного листка дорівнює $T(1) = \Theta(1)$ і всі листки знаходяться на глибині $\log_b n$ рівня, оскільки $n / b^{\log_b n} = 1$. Всього ж в дереві $a^{\log_b n} = n^{\log_b a}$ листків.

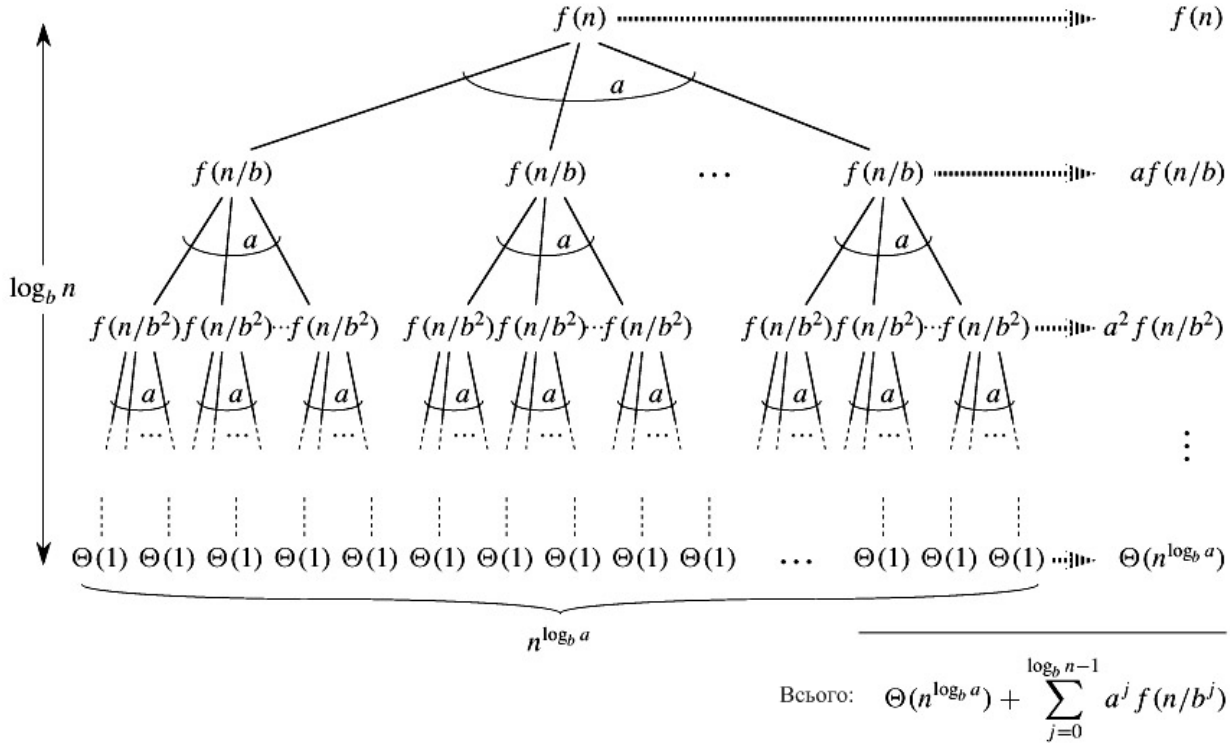

Рис. 7.2. Дерево рекурсії співвідношення $T(n) = aT(n/b) + f(n)$.

Рівняння 7.5 можна отримати, додавши час виконання всіх листків дерева, як показано на рисунку. Час виконання внутрішніх вузлів рівня j дорівнює $a^j f\left(n / b^j\right)$, так що повний час роботи всіх внутрішніх вузлів можна записати як:

$$\sum_{j=0}^{\log_b n-1} a^j f\left(n / b^j\right).$$

В алгоритмі розбиття, який лежить в основі рекурентного співвідношення, що аналізується, ця сума відповідає повному часу, який витрачається на розбиття поставленої задачі на допоміжні підзадачі, та подальшому об'єднанню їх розв'язків. Сумарний час виконання всіх листків, тобто час розв'язку всіх $n^{\log_b a}$ допоміжних підзадач з розміром 1, дорівнює $\Theta\left(n^{\log_b a}\right)$. ■

Три часткових випадки основної теореми, які виражені в термінах дерева рекурсії, відповідають ситуаціям, коли повний час виконання дерева 1) здебільшого визначається часом виконання його листків, 2) рівномірно розподілений по всім рівням дерева або 3) здебільшого визначається часом виконання кореня. Сума в рівнянні (7.5) описує час, який витрачається на етапи розбиття та об'єднання в алгоритмі розбиття, що лежить в основі рекурентного

співвідношення. В наступній теоремі обґрунтовуються асимптотичні оцінки порядку зростання цієї суми.

Теорема 7.3. Нехай $a \geq 1$ та $b > 1$ – константи, $f(n)$ – невід'ємна функція, визначена для точних степенів числа b. Тоді функція $g(n)$, яка визначена для точних степенів числа b за допомогою співвідношення

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f\left(n / b^j\right), \tag{7.6}$$

можна асимптотично оцінити наступним чином.

1. Якщо для деякої константи $\varepsilon > 0$ виконується умова $f(n) = \mathrm{O}\left(n^{\log_b a - \varepsilon}\right)$, то $g(n) = \mathrm{O}\left(n^{\log_b a}\right)$.
2. Якщо $f(n) = \Theta\left(n^{\log_b a}\right)$, то $g(n) = \Theta\left(n^{\log_b a} \lg n\right)$.
3. Якщо для деякої константи $c < 1$ та всіх $n \geq b$ виконується співвідношення $af(n/b) \leq cf(n)$, то $g(n) = \Theta\left(f(n)\right)$.

Доведення. В першому випадку виконується співвідношення $f(n) = \mathrm{O}\left(n^{\log_b a - \varepsilon}\right)$, з якого слідує, що $f\left(n / b^j\right) = \mathrm{O}\left(\left(n / b^j\right)^{\log_b a - \varepsilon}\right)$. Підставивши цю рівність у рівняння (7.6), отримуємо наступне співвідношення:

$$g(n) = \mathrm{O}\left(\sum_{j=0}^{\log_b n - 1} a^j (n / b^j)^{\log_b a - \varepsilon}\right), \tag{7.7}$$

Оцінимо суму в О-позначенні. Для цього винесемо з-під знаку додавання сталий множник та виконуємо деякі спрощення, в результаті чого сума перетвориться на геометричну прогресію, що зростає:

$$\sum_{j=0}^{\log_b n - 1} a^j (n / b^j)^{\log_b a - \varepsilon} = n^{\log_b a - \varepsilon} \sum_{j=0}^{\log_b n - 1} \left(\frac{ab^\varepsilon}{b^{\log_b a}}\right)^j = n^{\log_b a - \varepsilon} \sum_{j=0}^{\log_b n - 1} (b^\varepsilon)^j =$$

$$= n^{\log_b a - \varepsilon} \left(\frac{b^{\varepsilon \log_b n} - 1}{b^\varepsilon - 1}\right) = n^{\log_b a - \varepsilon} \left(\frac{n^\varepsilon - 1}{b^\varepsilon - 1}\right)$$

Оскільки b та ε – константи, останній вираз можна переписати у вигляді

$n^{\log_b a-\varepsilon}\mathrm{O}\left(n^{\varepsilon}\right)=\mathrm{O}\left(n^{\log_b a}\right)$. Підставивши цей вираз під знак суми в рівнянні (7.7), отримуємо, що $g\left(n\right)=\mathrm{O}\left(n^{\log_b a}\right)$, що відповідає випадку 1.

У випадку 2 робиться припущення, що $f\left(n\right)=\Theta\left(n^{\log_b a}\right)$, тому $f\left(n/b^j\right)=\Theta\left(\left(n/b^j\right)^{\log_b a}\right)$. Підставляючи це співвідношення у рівняння (7.6), отримуємо:

$$g(n)=\Theta\left(\sum_{j=0}^{\log_b n-1}a^j\left(n/b^j\right)^{\log_b a}\right). \tag{7.8}$$

Як і у випадку 1, оцінимо суму, що стоїть під знаком Θ, але на цей раз геометрична прогресія не отримується:

$$\sum_{j=0}^{\log_b n-1}a^j(n/b^j)^{\log_b a}=n^{\log_b a}\sum_{j=0}^{\log_b n-1}\left(\frac{a}{b^{\log_b a}}\right)^j=n^{\log_b a}\sum_{j=0}^{\log_b n-1}1=n^{\log_b a}\log_b n.$$

Підставивши цей вираз для суми у рівняння (7.8), отримуємо співвідношення:

$$g\left(n\right)=\Theta\left(n^{\log_b a}\log_b n\right)=\Theta\left(n^{\log_b a}\lg n\right),$$

що доводить випадок 2.

Випадок 3 доводиться аналогічно. Оскільки функція *f*(*n*) фігурує в означенні (7.6) функції *g*(*n*), і всі доданки функції *g*(*n*) невід'ємні, можна зробити висновок, що $g\left(n\right)=\Omega\left(f\left(n\right)\right)$ для точних степенів числа *b*. Згідно припущення, для деякої константи $c<1$ співвідношення $af\left(n/b\right)\le cf\left(n\right)$ виконується для всіх $n\ge b$, звідки $f\left(n/b\right)\le\left(c/a\right)f\left(n\right)$. Ітеруючи *j* разів, отримуємо, що $f\left(n/b^j\right)\le\left(c/a\right)^j f\left(n\right)$, або, що теж саме $a^j f\left(n/b^j\right)\le c^j f\left(n\right)$. Після підстановки у рівняння (7.6) та деяких спрощень отримуємо геометричну прогресію, що спадає:

$$g\left(n\right)=\sum_{j=0}^{\log_b n-1}a^j f\left(n/b^j\right)\le\sum_{j=0}^{\log_b n-1}c^j f\left(n\right)\le f\left(n\right)\sum_{j=0}^{\infty}c^j=f\left(n\right)\left(\frac{1}{1-c}\right)=\mathrm{O}\left(f\left(n\right)\right).$$

Таким чином, можна прийти до висновку, що $g\left(n\right)=\Theta\left(f\left(n\right)\right)$. ■

Тепер можна довести основну теорему в частковому випадку, коли n – точна степінь b.

Теорема 7.4. Нехай $a \geq 1$ та $b > 1$ – константи, $f(n)$ – невід'ємна функція, визначена для точних степенів числа b. Визначимо функцію $T(n)$ на множині точних степенів числа b за допомогою наступного рекурентного співвідношення:

$$T(n) = \begin{cases} \Theta(1), & \text{якщо } n = 1; \\ aT(n/b) + f(n), & \text{якщо } n = b^i, \end{cases}$$

де i – додатне ціле число. Тоді для цієї функції можна дати наступну асимптотичну оцінку (яка справедлива для точних степенів числа b).

1. Якщо для деякої сталої $\varepsilon > 0$ виконується умова $f(n) = \mathrm{O}\left(n^{\log_b a - \varepsilon}\right)$, то $T(n) = \Theta\left(n^{\log_b a}\right)$.
2. Якщо $f(n) = \Theta\left(n^{\log_b a}\right)$, то $T(n) = \Theta\left(n^{\log_b a} \lg n\right)$.
3. Якщо для деякої сталої $\varepsilon > 0$ виконується умова $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$, і якщо для деякої сталої $c < 1$ і всіх достатньо великих n, справедливе співвідношення $af(n/b) \leq cf(n)$, то $T(n) = \Theta\left(f(n)\right)$.

Доведення. За допомогою асимптотичних границь, які отримані в теоремі 7.3, оцінимо суму (7.5) з теореми 7.2. У випадку 1 маємо:

$$T(n) = \Theta\left(n^{\log_b a}\right) + \mathrm{O}\left(n^{\log_b a}\right) = \Theta\left(n^{\log_b a}\right),$$

у випадку 2 –

$$T(n) = \Theta\left(n^{\log_b a}\right) + \Theta\left(n^{\log_b a} \lg n\right) = \Theta\left(n^{\log_b a} \lg n\right),$$

а у випадку 3, оскільки $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$$T(n) = \Theta\left(n^{\log_b a}\right) + \Theta\left(f(n)\right) = \Theta\left(f(n)\right). \blacksquare$$

Література

1. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ, 2-е издание. : Пер. с англ. – М. : Изд. дом "Вильямс", 2011. – 1296 с. (Глава 4)