Algorithms and data structures

lecture #4. Master Theorem

Mentor: Rustam Khakov

lecture #4. Master Theorem

- Master Theorem
  - Описание
  - Общая форма
  - Применение

Master Theorem (Основная теорема)

Основная теорема о рекуррентных соотношениях (Master theorem) используется в анализе алгоритмов для получения асимптотической оценки рекурсивных при анализе алгоритмов типа «разделяй и властвуй» (divide and conquer).

Теорема была введена и доказана Джоном Бентли, Доротеном Хакеном и Джеймсом Хакеном в 1980 году.

Основная теорема о рекуррентных соотношениях — это формула, предназначенная для решения рекуррентных соотношений следующего вида:

$T(n) = aT(n/b) + f(n)$, где

n = объем входных данных

a = количество подзадач в рекурсии

n/b = размер каждой подзадачи. Предполагается, что все подзадачи имеют одинаковый размер.

$f(n)$ = оценка выполненной работы вне рекурсивных вызовов.

Формулировка

Если a ≥ 1 и b > 1 — константы, а f(n) — асимптотически положительная функция, то временная сложность рекуррентного соотношения задается выражением:

T(n) = aT(n/b) + f(n)

T(n) имеет следующие асимптотические оценки:

$$T(n) = \begin{cases} a\,T\left(\frac{n}{b}\right) + O(n^c), & n > 1 \\ O(1), & n = 1 \end{cases},$$

где $a \in \mathbb{N}, b \in \mathbb{R}, b > 1, c \in R^+$.

Тогда асимптотическое решение имеет вид:

1. Если $c > \log_b a$, то $T(n) = O\left(n^c\right)$
2. Если $c = \log_b a$, то $T(n) = O\left(n^c \log n\right)$
3. Если $c < \log_b a$, то $T(n) = O\left(n^{\log_b a}\right)$

Алгоритм	Рекуррентное соотношение	Время работы	Комментарий
Целочисленный двоичный поиск	$T(n) = T\left(\frac{n}{2}\right) + O(1)$	$O(\log n)$	По мастер-теореме $c = \log_b a$, где $a = 1, b = 2, c = 0$
Обход бинарного дерева	$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$	$O(n)$	По мастер-теореме $c < \log_b a$, где $a = 2, b = 2, c = 0$
Сортировка слиянием	$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$	$O(n \log n)$	По мастер-теореме $c = \log_b a$, где $a = 2, b = 2, c = 1$

Пример использования

T(n) = 3T(n/2) + n^2

Что есть, что:

a = 3

n/b = n/2

f(n) = n^2

log b a = log2 3 ≈ < 2

то есть f(n) < n log b a+ϵ, где ϵ — константа.

То есть, это третья оценка.

T(n) = f(n) = O(n^2)

Двоичный поиск –

T(n) = T(n/2)+O(1) = O(log n) – вторая оценка

Сортировка слиянием –

T(n) = 2T(n/2)+O(n) = O(n log n) – вторая оценка

Когда не работает

- $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$

  a не является константой, для основной теоремы требуется постоянное количество подзадач;

- $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

  между f(n) и $n^{\log_b a}$ существует неполиномиальная зависимость;

- $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

  a<1, но основная теорема требует наличия хотя бы одной подзадачи;

- $T(n) = 64T\left(\frac{n}{8}\right) - n^2 \log n$

  f(n) является отрицательной величиной;