Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 6

КЛАССИФИКАЦИЯ АЛГОРИТМОВ ПО ТИПУ ТРУДОЕМКОСТИ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, **8 916 589 94 04**

Математическое введение

Во второй половине XIX века немецкий математик Поль (Пауль) Давид Густав Дюбуа-Раймон (фр. Paul David Gustave du Bois-Reymond) ввел специальный символ для сравнения функций по скорости их роста — «птичку», как аналог простого сравнения чисел $a < b$.

Если функции $f(x)$ и $g(x)$ такие что, $\lim_{x\to\infty} f(x) = \infty$ и $\lim_{x\to\infty} g(x) = \infty$, и при этом

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0, \text{ то: } f(x) \prec g(x), \lim_{x\to\infty} \frac{f(x)}{g(x)} = const \neq 0, \text{ то: } f(x) \sim g(x).$$

Это обозначение вводит асимптотическую иерархию функций:

$$\ln x \prec \sqrt{x} \prec x \prec x^2 \prec e^x \prec x^x$$

Вопрос — где в этой иерархии находится функция степенного логарифма?

$$f(x) = ln\, x^{ln\, x}$$

Введение

На ресурсные характеристики алгоритма, — трудоемкость и ёмкостную эффективность оказывают влияние различные характеристики множества исходных данных — совокупности допустимых входов алгоритма. Это как длина входа n, так и некоторые параметры, такие как конкретные значения некоторых элементов входа, их порядок или некоторые другие параметры на множестве D.

Поскольку значение функции трудоемкости $f_A(D)$ может определяться не только длиной входа n, но и рядом параметров множества D, то проведем декомпозицию трудоемкости — выделим в функции $f_A(D)$ количественную и параметрическую составляющую, обозначив их через $f_n(n)$ и $g_p(D)$ тогда

$$f_A(D) = f_A\left(f_n(n),\ g_p(D)\right).$$

Декомпозиция трудоемкости

Для подавляющего большинства алгоритмов функция $f_A(D)$ может быть представлена как композиция функций $f_n(n)$ и $g_p(D)$ либо в мультипликативной, либо в аддитивной форме

$$f_A(n,\ D) = f_n(n) \cdot\ g_p^*(D), \text{ или } f_A(n,\ D) = f_n(n) +\ g_p^+(D).$$

Мультипликативная форма характерна для ряда алгоритмов, когда сильно параметрически зависимый внешний цикл, определяющий перебор вариантов решения задачи, содержит внутренний цикл, имеющий количественно-зависимую трудоемкость. В силу конечности множества D_n существует худший случай для функций $g_p^*(D)$, и $g_p^+(D)$. Обозначим соответствующие функции через $g\hat{}^*(n)$ и $g\hat{}^+(n)$:

$$g\hat{}^*(n) = \max_{D\in D_n}\{\ g_p^*(D)\ \},\ \ g\hat{}^+(n) = g\hat{}(n) = \max_{D\in D_n}\{\ g_p^+(D)\ \}.$$

Типизация задач и решающих их алгоритмов по длине входа

Конкретные входы, порожденные различными задачами, обладают различными характеристическими особенностями. Однако мы можем выделить группу задач, порождающих входы фиксированной длины. Например, задача извлечения квадратно корня с заданной точностью всегда порождает вход, состоящий из двух чисел. В отличие от этой группы, другие задачи порождают входы переменной длины — задачи сортировки или умножения матриц являются характерными примерами задач этой группы. Таким образом, алгоритмы, решающие задачи из разных групп, имеют на входе или множество с заранее фиксированным числом элементов, или множество с переменным числом элементов, что приводит к следующей типизации задач, и, соответственно, решающих их алгоритмов.

A. Тип L_c — задачи с постоянной (фиксированной) длиной входа. Для этого типа задач и соответствующих им алгоритмов

$$\exists\ n = const: \forall D \in D_A\ |D| = n.$$

Примерами для этого типа могут служить алгоритмы вычисления значений стандартных функций, вычисления наибольшего общего делителя двух чисел, возведения в степень и т.д.

B. Тип L_n — задачи с переменной длиной входа. Для этого типа задач и соответствующих им алгоритмов

$$\exists D_1,\ D_2 \in D_A:\ |D_1| \neq |D_2|.$$

Примерами являются алгоритмы выполнения операций с матрицами и векторами, алгоритмы обработки строк символов, алгоритмы сортировки и т.д.

II. Классификация алгоритмов по трудоемкости в типе L_c

Пусть на функцию трудоемкости алгоритма не оказывают влияние никакие другие характеристические особенности входа, за исключением его длины. Но поскольку длина входа постоянна, то тогда эти алгоритмы должны иметь фиксированную трудоемкость. Таким образом, мы вводим в типе L_c класс алгоритмов с постоянной трудоемкостью.

Класс C — алгоритмы с постоянной трудоемкостью в типе L_c:

$$\exists n = const,\ \exists\ k = const: \forall D \in D_A\ |D| = n\ , f_A(D) = k.$$

Тривиальный пример — алгоритм сложения двух чисел. Более реальный пример — вычисление прогиба балки по ее известным характеристикам и значению нагрузки и аналогичные вычисления по известным формулам.

В противном случае трудоемкость зависит от некоторых дополнительных особенностей входа, как правило, от значений элементов D,

Класс PR — алгоритмы с трудоемкостью, параметрически зависящей от значений некоторых элементов множества D в типе L_c

$$\exists n = const,\ \exists\ k = const: \forall D \in D_A\ |D| = n\ f_n(n) = const,\ \Rightarrow f_A(D) = c \cdot g(D).$$

Примерами алгоритмов с параметрически зависимой трудоемкостью при фиксированной длине входа являются алгоритмы вычисления стандартных функций с заданной точностью путем вычисления соответствующих степенных рядов. Очевидно, что такие алгоритмы, имея на входе два числовых значения — аргумент функции и точность, задают существенно зависящее от этих значений количество базовых операций. Например, для алгоритма вычисления x^k последовательным умножением — $f_A(D) = f_A(k)$, а для алгоритма вычисления экспоненты по формуле $e^x = \sum(x^n/n!)$, с точностью до ε — $f_A(D) = f_A(x,\ \varepsilon)$. Заметим, что в данном случае оба элемента входа являются параметрами трудоемкости.

III. Классификации алгоритмов по трудоемкости в типе L_n

Для задач и соответствующих алгоритмов, относящихся к типу L_n возможно несколько вариантов зависимости трудоемкости алгоритма от особенностей входа. В типе L_n мы рассматриваем совокупность подмножеств входов D_n, обладающих длиной n. Могут быть выделены следующие классы:

А. Класс C — алгоритмы с постоянной (фиксированной) трудоемкостью:

$$\exists\ k = const: \forall n: D_n \in D_A\ \forall D \in D_n\ \ f_A(D) = k.$$

Рассмотрим задачу поиска максимального элемента в уже отсортированном массиве, содержащем n элементов. Очевидно, что вне зависимости от значения n результат получается за фиксированное число базовых операций модели вычислений. Еще один пример — поиск по ключу в массиве длиной n с использованием минимальной совершенной функции хеширования, что обеспечивает поиск по любому ключу за $\Theta(1)$ базовых операций.

B. Класс PRn — класс параметрически-зависимых по трудоемкости алгоритмов. Это алгоритмы, трудоемкость которых определяется конкретными значениями всех или некоторых элементов из входного множества D или иными особенностями входа, например порядком расположения элементов.

$$\forall D_n \in D_A \ \forall D \in D_n \ f_n(n) = \text{const} = \text{c}, \ \Rightarrow \ \hat{f_A}(n, \ D) = \text{c} \cdot \hat{g^*}(n).$$

C. Класс N — класс количественно-зависимых по трудоемкости алгоритмов. Это алгоритмы, функция трудоемкости которых зависит только от длины конкретного входа, при этом

$$\hat{g^+}(n) = 0 \Rightarrow \hat{g^*}(n) = 1 \Rightarrow f_A(D) = f_n(n).$$

Примерами алгоритмов с количественно-зависимой функцией трудоемкости могут служить алгоритмы для стандартных операций с массивами и матрицами — умножение матриц, умножение матрицы на вектор и т. д. Анализ таких алгоритмов, как правило, не вызывает затруднений.

Есть ли в классе $\boldsymbol{N}$ алгоритмы сортировки? Какие еще алгоритмы в классе $\boldsymbol{N}$?

D. Класс $\boldsymbol{NPR}$ — класс количественно-параметрических по трудоемкости алгоритмов. Это достаточно широкий класс алгоритмов, т. к. в большинстве практических случаев функция трудоемкости зависит как от длины входе, так и от значений параметров входных данных. В этом случае

$$f_n(n) \neq const, \quad g^{\wedge *}(n) \neq const, g^{\wedge +}(n) \neq const,$$

$$f_A(D) = f_n(n) \cdot g^*(n), \text{ или } f_A(D) = f_n(n) + g^+(n).$$

В качестве одного из общих примеров для мультипликативной декомпозиции можно привести алгоритмы, реализующие ряд численных методов, в которых параметрически зависимый внешний цикл по точности включает в себя количественно-зависимый фрагмент по размерности, порождая мультипликативную форму для $f_A(D)$. Пример — метод простой итерации для решения СЛАУ.

IV. Подклассы класса NPR по степени влияния на трудоемкость параметрического компонента

Выделим в классе $\boldsymbol{NPR}$ подклассы с целью выяснения степени влияния количественной и параметрической составляющей на главный порядок функции $f_A(D) = f_n(n) \cdot g^*(n)$, или $f_A(D) = f_n(n) + g^+(n)$.

Дальнейшая классификация проводится для аддитивной декомпозиции $f_A(D)$.

A. Подкласс $\boldsymbol{NPRL}$ $(\boldsymbol{Low})$ — слабо-параметрический $\boldsymbol{NPR}$:

$$g^{\wedge +}(n) \prec f_n(n),$$

Для алгоритмов этого подкласса параметрическая составляющая не влияет ни на главный порядок функции трудоемкости, ни на мультипликативную константу главного порядка. Параметры входа влияют на асимптотически более слабые компоненты $f_A(D)$. Это алгоритмы, трудоемкость которых слабо зависит от параметрической составляющей. Приведите пожалуйста примеры?

B. Подкласс $NPRE(Equivalent)$ — средне-параметрический подкласс класса NPR. Это подкласс алгоритмов, у которых в функции трудоемкости составляющая $g\hat{}^{+}(n)$ имеет равный с $f_n(n)$ порядок роста:

$$g\hat{}^{+}(n) \sim f_n(n),$$

Для алгоритмов этого подкласса параметрическая составляющая влияет не более чем на коэффициент главного порядка функции трудоемкости. Обе компоненты — и количественная и параметрическая имеют одинаковую асимптотику роста. К этому подклассу относится, например, алгоритм поиска максимума в массиве, т. к. в худшем случае функции $g\hat{}^{+}(n) = \mathrm{a} * \mathrm{n}$ и $f_n(n) = \mathrm{b} * \mathrm{n}$ имеют равные асимптотические порядки. Очевидно, что трудоемкость в худшем случае будет иметь вид $f_A(n) = (a + b) * n$.

Какие еще алгоритмы можно отнести к этому классу?

C. Подкласс $NPRH$ $(High)$ — сильно-параметрической подкласс класса NPR. Это подкласс алгоритмов, в трудоемкости которых составляющая $g^{\wedge +}(n)$ имеет асимптотический порядок роста выше $f_n(n)$:

$$f_n(n) \prec g^{\wedge +}(n) \ ,$$

Алгоритмы этого подкласса отличаются тем, что именно параметрический компонент определяет главный порядок функции трудоемкости. Для таких алгоритмов характерна, как правило, мультипликативная форма функции трудоемкости, и при этом $f_n(n) \prec g^{\wedge *}(n)$. Эта форма особенно ярко выражена в алгоритмах большинства итерационных вычислительных методов, в которых внешний цикл по точности порождает параметрическую составляющую, а трудоемкость тела цикла имеет количественную оценку.

Пример — метод ветвей и границ для задачи коммивояжера.