Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 4

ТЕРМИНОЛОГИЯ И ОБОЗНАЧЕНИЯ В АНАЛИЗЕ АЛГОРИТМОВ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, 8 916 589 94 04

Универсальные оценки алгоритмов

Первые работы в области оценки алгоритмов связаны с алгоритмическими формализмами теории алгоритмов. Предложенные в этих работах подходы опирались на соответствующие этим формализмам модели вычислений. Переход в область практического анализа алгоритмов без сомнений связан с именем Д.Э. Кнута и его фундаментального труда «Искусство программирования для ЭВМ», в котором моделью вычислений была абстрактная машина MIX с соответствующим языком ассемблера. Отдавая дань уважения Д. Э. Кнуту, отметим, что для анализа некоторого алгоритма необходимо было записать его реализацию в ассемблере машины MIX, что не всегда удобно и рационально.

Работы по оценке алгоритмов появились в конце 50-х - 60-х годах XX века. В это время сформировались следующие два основных подхода к решению задачи оценки алгоритмов:

— оценка сложности записи самого алгоритма;

— оценка сложности вычислительного процесса, задаваемого алгоритмом.

При оценке сложности записи обычно рассматривается некоторый функционал, ставящий в соответствие алгоритму A некоторое число $\rho(A)$, характеризующее сложность записи алгоритма. Такими характеристиками может быть количество команд (в принятом алгоритмическом формализме) в записи алгоритма, длина записи алгоритма, и т. д. Одними из первых работ с применением таких мер сложности были работы А.Н. Колмогорова и А.А. Маркова. В современной постановке задачи сравнительного анализа алгоритмов, такая оценка сложности самого алгоритма может быть соотнесена, например, с объемом программы, реализующей данный алгоритм. Более точно, за оценку алгоритма может быть принят *объем оперативной памяти в области кода*, занимаемый его исполнительным файлом (файл *.exe).

Оценка сложности вычислительного процесса, задаваемого алгоритмом связана с определением меры сложности вычислений, задаваемых алгоритмом для конкретных допустимых задач. В рамках этого подхода рассматривается некоторый функционал, соотносящий алгоритму и индивидуальной (конкретной) задаче определенное число. **Функционал** — функция, заданная на произвольном множестве и имеющая числовую область значений: обычно множество вещественных чисел или комплексных чисел.

Пусть A — алгоритм, α — конкретная задача из класса задач, решаемых алгоритмом A, тогда функционал $\mu(A, \alpha)$ определяет сложность вычислений. Это число определяет затраты алгоритма применительно к конкретным исходным данным до получения соответствующего результата. В качестве функционала $\mu(A, \alpha)$ можно рассматривать количество элементарных шагов алгоритма в принятом алгоритмическом формализме.

Подход к оценкам алгоритмов в теории ресурсной эффективности

Более детальное исследование алгоритмов связано с оценками их *ресурсной эффективности*. Выполнение требования *финитности* еще не говорит о том, насколько долго мы будем ожидать результата. В этом смысле было бы целесообразно попытаться получить зависимость ресурсных оценок алгоритма, как функцию некоторого аргумента. Поскольку интуитивно различные конкретные проблемы, скорее всего, будут решаться данным алгоритмом за большее время при большем количестве элементов на входе, то в качестве такого аргумента можно выбрать *оценку длины входных данных алгоритма*. В рамках практического анализа алгоритмов под **длиной входа** обычно понимается или количество объектов, обрабатываемых алгоритмом (для большинства задач — это количество чисел или символов) или некоторая мера количества этих объектов, мы уточним это понятие ниже.

Терминология и обозначения в теории ресурсной эффективности

Наиболее употребительными оценками ресурсной эффективности алгоритмов в выбранной модели вычислений являются:

— оценка требуемого ресурса механизма реализации — **вычислительная сложность**;

— оценка требуемого ресурса информационного носителя — **емкостная сложность**.

Требуемые алгоритмом ресурсы принятой модели вычислений отражаются в *универсальных* оценках качества алгоритмов и исследуются в теории ресурсной эффективности. В аспекте программной реализации алгоритма можно говорить о требуемом *ресурсе процессора и оперативной памяти* — это характеристики *ресурсной эффективности программной реализации*.

Таким образом, *ресурсная эффективность алгоритма оценивается в ресурсах модели вычислений*, а его *программной реализации — в ресурсах реального компьютера*.

Методы теории сложности позволяют получить асимптотические оценки требуемых алгоритмом ресурсов. Для сравнения алгоритмов в реальном диапазоне, определяемом областью применения программной системы, необходимо более детальные знания о ресурсных требованиях алгоритма.

В обширной литературе по анализу алгоритмов и в публикациях по сложности вычислений часто используется термин «**размерность задачи**», как правило, не определяемый, мы уточняем его, вводя термин «**длина входа**». Укажем также на возникающую у ряда авторов неоднозначность, связанную с определением понятий вычислительной и временной сложности, равно как и временной эффективности.

Вход и длина входа

Введем следующие определения, учитывая, что в настоящее время вместо термина Поста «Общая проблема» общепринятым является термин «**Задача**», который мы и будем использовать далее, с обозначением Z, а вместо постовского термина «Конкретная проблема» мы будем далее использовать термин «**Индивидуальная задача**» и вводимый ниже термин «**Вход алгоритма**».

Определение 1. **Вход алгоритма**.

► Пусть A — алгоритм, решающий задачу Z. Индивидуальная задача Z представляет собой множество исходных данных D. В рамках принятой модели вычислений D есть конечное упорядоченное множество (кортеж), состоящее из m элементов d_i, каждый из которых есть слово фиксированной длины β в алфавите $\{0,1\}$: $D = \left(d_i,\ i = \overline{1,\ m}\right),\ |D| = m$. Множество D, задающее индивидуальную задачу, будем называть *входом алгоритма* A. ◄

Таким образом алгоритм A решает индивидуальную задачу, заданную входом D. Обозначим далее через D_Z множество всех допустимых индивидуальных задач задачи Z, тогда $D \in D_Z$. Тем самым сама задача Z задана множеством D_Z, т.е. множеством индивидуальных задач, что соответствует формализму Поста: «общая проблема задана множеством конкретных проблем».

Конкретизируем понятие «*размерность задачи*». Простой способ такой конкретизации — использование мощности входа D как естественное определение длины входа. Однако стремление выделить основной «информационной носитель» задачи вступает в противоречие с таким естественным определением. Например, передавая массив на обработку, мы передаем не только сами элементы, но значение длины массива. Кроме того, для задачи умножения матриц, исторически сложилось специальное понимание размерности, — в данном случае это линейный размер квадратной матрицы.

Определение 2**. Длина входа**.

► Определим для входов $D \in D_Z$ функционал $\mu_Z(\cdot)$ с областью определения на D_Z и областью значений в множестве $\boldsymbol{N}$, отражающий особенности понимания размерности задачи Z. Значение $\mu_Z(D)$ будем называть *длиной входа* алгоритма, решающего задачу Z на индивидуальной задаче D. Везде далее через n будет обозначаться значение $\mu_Z(D)$.◄

Особо отметим, что введенная функция $\mu_Z(D)$ «*индивидуальна*» по отношению к задаче Z. Разные задачи при одинаковой мощности входа могут иметь различные длины входа, в силу специфики функции $\mu_Z(D)$. Например, для индивидуальной задачи сортировки $|D| = m$, тогда $\mu_Z(D) = m - 1 = n$ (мы считаем, что длина входа равна длине массива, и не учитываем одну ячейку, хранящую эту длину). Для некоторых задач естественным является определение функции $\mu_Z(\cdot)$ с областью значений во множестве $\boldsymbol{N}^k, k \geq 2$.

В качестве примеров, для $k = 2$ и $\mu_Z(D) = (n, m)$ приведем задачу умножения прямоугольных матриц $(n, m) \times (m, n)$ и задачу поиска подстроки в строке, где n — длина строки, а m — длина подстроки поиска.

Функции ресурсной эффективности компьютерных алгоритмов

Используя введенное понятие входа алгоритма, эквивалентное индивидуальной задаче, и понятие длины входа, определим основные функции, отражающие ресурсные затраты алгоритма на решение индивидуальной задачи в принятой модели вычислений.

Определение 3. **Трудоёмкость алгоритма**.

► Под трудоёмкостью алгоритма A на входе D будем понимать количество элементарных операций в принятой модели вычислений, задаваемых алгоритмом A на входе D.◄

Очевидно, что смена модели вычислений приводит к изменению трудоемкости алгоритма. В этом аспекте обязательным является применение одной и той же модели вычислений при сравнительном анализе различных алгоритмов решения интересующей исследователя задачи.

Определение 4. **Функция трудоемкости**.

► Определим функцию $f_A(D)$ с областью определения на D_Z и областью значений во множестве $\boldsymbol{N}$. Значение $f_A(D)$ есть трудоёмкость алгоритма A на индивидуальной задаче, заданной входом D. Функцию $f_A(D)$ будем называть *функцией трудоемкости*. ◄

Заметим, что значение функции трудоемкости для любого допустимого входа D является целым положительным числом в силу того, что алгоритм является финитным 1-процессом по Посту, а элементарные операции модели вычислений неделимы.

При более детальном анализе ряда алгоритмов оказывается, что не всегда трудоемкость алгоритма на одном входе D длины n, где $n = \mu_Z(D)$, совпадает с его трудоемкостью на другом входе такой же длины. Мы наблюдаем различные трудоемкости на входах фиксированной длины

Наличие таких этих колебаний приводит к введению верхних и нижних оценок трудоемкости на основе следующих рассуждений. Рассмотрим допустимые входы длины n для задачи Z — в общем случае существует подмножество (для большинства задач собственное) множества D_Z, включающее все входы, имеющие длину n, — обозначим его через D_n:

$$D_n = \{\ D\ \ |\ \ \mu_Z(D) = n\ \}.$$

Поскольку элементы d_i представляют собой слова фиксированной длины в алфавите $\{0,1\}$, множество D_n является конечным — обозначим его мощность через M_{D_n}, т. е. $M_{D_n} = |D_n|$.

Определение 8.5 **Трудоёмкость в худшем и лучшем случаях**.

► Под худшим случаем будем понимать такой вход D длины n, на котором алгоритм A задает наибольшее количество элементарных операций, при этом максимум берется по всем $D \in D_n$. Трудоемкость алгоритма на этом входе будем называть *трудоемкостью в худшем случае*, и обозначать ее через $f_A^{\wedge}(n)$, тогда

$$f_A^{\wedge}(n) = \max_{D \in D_n} \{ f_A(D) \},$$

по аналогии через $f_A^{\vee}(n)$ будем обозначать *трудоемкость в лучшем случае*, как трудоемкость с наименьшим количеством операций на всех входах длины n:

$$f_A^{\vee}(n) = \min_{D \in D_n} \{ f_A(D) \}. \blacktriangleleft$$

Заметим, что для функций трудоемкости в худшем и лучшем случаях произошло изменение *аргумента* — мы оперируем теперь длиной входа, а не собственно конкретным входом, т.е. индивидуальной задачей.

Определение 8.6 **Трудоёмкость в среднем**.

► Трудоемкость алгоритма в среднем — это среднее количество операций, задаваемых алгоритмом A на входах длины n, где усреднение берется по всем $D \in D_n$. Введем для трудоемкости в среднем обозначение $\overline{f_A}(n)$, тогда

$$\overline{f_A}(n) = \sum_{D \in D_n} p(D) \cdot f_A(D),$$

где $p(D)$ есть частотная встречаемость входа D для анализируемой области применения алгоритма. ◄ При равновозможности всех $D \in D_n$:

$$\overline{f_A}(n) = \frac{1}{M_{D_n}} \sum_{D \in D_n} f_A(D).$$

Функция $\overline{f_A}(n)$ есть *вещественнозначная* функция целочисленного аргумента. Отметим, что задача конкретизации особенностей проблемной области в терминах $p(D)$ является по необходимости трудной.

Оценка памяти — емкостная сложность

Состояние памяти модели вычислений определяется значениями, записанными в ячейках этой памяти. Тогда механизм реализации модели вычислений, выполняя элементарные операции, заданные алгоритмом, переводит исходное состояние памяти — вход алгоритма в конечное состояние — найденное алгоритмом решение задачи. При этом в ходе решения задачи может быть задействовано некоторое *дополнительное* количество ячеек памяти, помимо памяти входа и памяти результата.

Поскольку все алгоритмы, решающие задачу Z, работают с одинаковыми входами и получают одинаковые результаты, т.е. объемы памяти для входа и результата одинаковы для всех сравниваемых алгоритмов, то для их сравнения имеет смысл использовать требуемый алгоритмами *объем дополнительной памяти*. Рассуждая по аналогии с трудоемкостью, имеем:

Определение 8.8 **Функция объема дополнительная памяти**.

► Под объемом дополнительной памяти, требуемым алгоритмом A для входа D, будем понимать максимальное количество ячеек памяти модели вычислений, задействованных в ходе выполнения алгоритма, без учёта памяти входа и памяти результата. Функцию, значением которой является объем дополнительной памяти, требуемый алгоритмом для входа D будем обозначать через $V_A(D)$. Значение функции $V_A(D)$ есть целое положительное число. ◄

Введем для функции объема памяти, по аналогии с трудоемкостью, обозначения для лучшего, худшего и среднего случая на множестве входов длины n:

$$V_A^{\wedge}(n),\ V_A^{\vee}(n),\ \overline{V_A}(n).$$

Литература к лекции 4

[1.1] Кнут Д. Э. Искусство программирования, том 1. Основные алгоритмы, 3-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 720 с.

[1.2] Офман Ю. П. Об алгоритмической сложности дискретных функций // Доклады АН СССР. 1962. Т. 45. Вып. 1. С.48–51.

[1.3] Трахтенброт Б. А. Сложность алгоритмов и вычислений — Новосибирск: НГУ, 1967. —243 с.

[1.4] Колмогоров А. Н. Три подхода к определению понятия количества информации // Проблемы передачи информации. Вып. 1. 1965.

[1.5] Колмогоров А. Н., Успенский В. А. К определению алгоритма // Успехи математических наук. 1958. Т. 13. № 4. С. 3–28.

[1.6] Марков А. А. Теория алгоритмов. // Труды математического института АН СССР им. В. А. Стеклова. — М.,1954. Т. 42.

[1.7] Цейтин Г. С. Оценка числа шагов при применении нормального алгоритма. // Математика в СССР за 40 лет. — М., 1959. Т. 1.

[1.8] Алферова З. В. Теория алгоритмов. — М.: Статистика, 1973. — 163 с.

[1.9]Гашков С. Б., Чубариков В. Н. Арифметика. Алгоритмы. Сложность вычислений: Учеб. пособие для вузов / Под ред. В. А. Садовничего. –2-е изд., перераб. — М.: Высш. шк., 2000. — 320 с.

[1.10] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов: Пер. с англ.: — М.: Мир, 1979. — 546 с.

[1.11] Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. — М.: МЦНМО, 1999. — 960 с.

[1.12] Хопкрофт Дж., Мотовани Р., Ульман Дж. Ведение в теорию автоматов, языков и вычислений, 2-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 528 с.

[1.13] Макконелл Дж. Основы современных алгоритмов. 2-е дополненное издание. — М.: Техносфера, 2004. — 368 с.

[1.14] Ульянов М.В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. — М.: ФИЗМАТЛИТ, 2008. — 304 с.