Вычислительная сложность

СиАОД лекция 8

Вычислительная сложность

Понятие в информатике и теории алгоритмов, обозначающее функцию зависимости объёма работы, которая выполняется некоторым алгоритмом, от размера входных данных.

Объём работы в информационном времени и пространстве - вычислительных ресурсах. Время -количество элементарных шагов, необходимых для решения задачи; пространство - объём памяти или место на носителе данных. Задача - ответить на вопрос: «как изменится время исполнения и объём занятой памяти в зависимости от размера входа?». Под размером входа понимается длина описания данных задачи в битах, а под размером выхода — длина описания решения задачи.

Временная и пространственная сложности

Теория сложности вычислений возникла из потребности сравнивать быстродействие алгоритмов, чётко описывать их поведение (время исполнения и объём необходимой памяти) в зависимости от размера входа.

Количество элементарных операций, затраченных алгоритмом для решения конкретного экземпляра задачи, зависит не только от размера входных данных, но и от самих данных. Например, количество операций алгоритма сортировки вставками значительно меньше в случае, если входные данные уже отсортированы. Чтобы избежать подобных трудностей, рассматривают понятие временной сложности алгоритма в худшем случае.

Временная сложность алгоритма

(в худшем случае) — это функция от размера входных данных, равная максимальному количеству элементарных операций, проделываемых алгоритмом для решения экземпляра задачи указанного размера.

Количество элементарных операций, затраченных алгоритмом для решения конкретного экземпляра задачи, зависит не только от размера входных данных, но и от самих данных. Например, количество операций алгоритма сортировки вставками значительно меньше в случае, если входные данные уже отсортированы. Чтобы избежать подобных трудностей, рассматривают понятие временной сложности алгоритма в худшем случае.

Аналогично понятию временной сложности в худшем случае определяется понятие временная сложность алгоритма в **наилучшем случае**. Также рассматривают понятие **среднее время** работы алгоритма, то есть **математическое ожидание** времени работы алгоритма. Иногда говорят просто: «Временная сложность алгоритма» или «Время работы алгоритма», имея в виду временную сложность алгоритма в худшем, наилучшем или среднем случае (в зависимости от контекста).

Пространственная сложность алгоритма

По аналогии с временной сложностью, определяют *пространственную сложность алгоритма*, только здесь говорят не о количестве элементарных операций, а об объёме используемой памяти.

Асимптотическая сложность

Несмотря на то, что функция временной сложности алгоритма в некоторых случаях может быть определена точно, в большинстве случаев искать точное её значение бессмысленно. Дело в том, что во-первых, точное значение временной сложности зависит от определения элементарных операций (например, сложность можно измерять в количестве арифметических операций, битовых операций или операций на машине Тьюринга), а во-вторых, при увеличении размера входных данных вклад постоянных множителей и слагаемых низших порядков, фигурирующих в выражении для точного времени работы, становится крайне незначительным.

Рассмотрение входных данных большого размера и оценка порядка роста времени работы алгоритма приводят к понятию асимптотической сложности алгоритма. При этом алгоритм с меньшей асимптотической сложностью является более эффективным для всех входных данных, за исключением лишь, возможно, данных малого размера. Для записи асимптотической сложности алгоритмов используются асимптотические обозначения:

Обозначение	Интуитивное объяснение
$f(n) \in O(g(n))$	f ограничена сверху функцией g (с точностью до постоянного множителя) асимптотически
$f(n) \in \Omega(g(n))$	f ограничена снизу функцией g (с точностью до постоянного множителя) асимптотически
$f(n) \in \Theta(g(n))$	f ограничена снизу и сверху функцией g асимптотически
$f(n) \in o(g(n))$	g доминирует над f асимптотически
$f(n) \in \omega(g(n))$	f доминирует над g асимптотически
$f(n) \sim g(n)$	f эквивалентна g асимптотически

Примеры:

«почистить ковёр пылесосом» требует время, линейно зависящее от его площади (Θ (A)), то есть на ковёр, площадь которого больше в два раза, уйдет в два раза больше времени. Соответственно, при увеличении площади ковра в сто тысяч раз объём работы увеличивается строго пропорционально в сто тысяч раз, и т. п.

«найти имя в телефонной книге» требует всего лишь времени, логарифмически зависящего от количества записей ($O(\log_2(n))$, так как, открыв книгу примерно в середине, мы уменьшаем размер «оставшейся проблемы» вдвое (за счет сортировки имен по алфавиту). Таким образом, в книге объёмом в 1000 страниц любое имя находится не больше, чем за $\log_2 1000 \approx 10$ раз (открываний книги). При увеличении объёма страниц до ста тысяч проблема все ещё решается за $\log_2 100000 \approx 17$ заходов.

Замечание

Поскольку $\log_a b = \frac{\log_c b}{\log_c a}$ в асимптотической оценке сложности часто пишут «логарифм» без упоминания основания — например:

$$O(n \log n)$$

Необходимо подчеркнуть, что степень роста наихудшего времени выполнения — не единственный или самый важный критерий оценки алгоритмов и программ.

Приведем несколько соображений, позволяющих посмотреть на критерий времени выполнения с других точек зрения:

Если создаваемая программа будет использована только несколько раз, тогда стоимость написания и отладки программы будет доминировать в общей стоимости программы, то есть фактическое время выполнения не окажет существенного влияния на общую стоимость. В этом случае следует предпочесть алгоритм, наиболее простой для реализации.

Если программа будет работать только с «малыми» входными данными, то степень роста времени выполнения будет иметь меньшее значение, чем константа, присутствующая в формуле времени выполнения. Вместе с тем и понятие «малости» входных данных зависит от точного времени выполнения конкурирующих алгоритмов.

Существуют алгоритмы, такие как алгоритм целочисленного умножения, асимптотически самые эффективные, но которые никогда не используют на практике даже для больших задач, так как их константы пропорциональности значительно превосходят подобные константы других, более простых и менее «эффективных» алгоритмов.

Другой пример — фибоначчиевы кучи, несмотря на асимптотическую эффективность, с практической точки зрения программная сложность реализации и большие значения констант в формулах времени работы делают их менее привлекательными, чем обычные бинарные деревья.

Известны примеры, когда эффективные алгоритмы требуют таких больших объёмов машинной памяти (без возможности использования более медленных внешних средств хранения), что этот фактор сводит на нет преимущество «эффективности» алгоритма. Таким образом, часто важна не только «сложность по времени», но и «сложность по памяти» (пространственная сложность).

В численных алгоритмах точность и устойчивость алгоритмов не менее важны, чем их временная эффективность.

Класс P

Класс P вмещает все те проблемы, решение которых считается «быстрым», то есть время решения которых полиномиально зависит от размера входа. Сюда относится сортировка, поиск в массиве, выяснение связности графов и многие другие.

Класс NP

Класс NP содержит задачи, которые недетерминированная машина Тьюринга в состоянии решить за полиномиальное количество шагов от размера входных данных. Их решение может быть проверено детерминированной машиной Тьюринга за полиномиальное количество шагов.

Класс NP

Следует заметить, что недетерминированная машина Тьюринга является лишь абстрактной моделью, в то время как современные компьютеры соответствуют детерминированной машине Тьюринга с ограниченной памятью.

Поскольку детерминированная машина Тьюринга может рассматриваться как специальный случай недетерминированной машины Тьюринга, класс NP включает в себя класс P, а также некоторые проблемы, для решения которых известны лишь алгоритмы,

Класс NP

экспоненциально зависящие от размера входа (то есть неэффективные для больших входов). В класс NP входят многие знаменитые проблемы, такие как задача коммивояжёра, задача выполнимости булевых формул, факторизация и др.

Проблема равенства классов P и NP

Вопрос о равенстве этих двух классов считается одной из самых сложных открытых проблем в области теоретической информатики. Математический институт Клэя включил эту проблему в список проблем тысячелетия, предложив награду размером в один миллион долларов США за её решение.

«O» большое и «o» малое

- математические обозначения для сравнения асимптотического поведения (асимптотики) функций. Используются в различных разделах математики, но активнее всего — в математическом анализе, теории чисел и комбинаторике, а также в информатике и теории алгоритмов. Под асимптотикой понимается характер изменения функции при её стремлении к определённой точке.

Смысл определени

o(f) «о малое от *f*» обозначает «бесконечно малое относительно *f*», пренебрежимо малую величину при рассмотрении *f*. Смысл термина «О большое» зависит от его области применения, но всегда *O(f)* растёт не быстрее, чем *f*.

Примеры использования

фраза «сложность алгоритма есть ***O(f(n))***» означает, что с увеличением параметра ***n***, характеризующего количество входной информации алгоритма, время работы алгоритма будет возрастать не быстрее, чем некоторая константа, умноженная на ***f(n)***;

Примеры использования

фраза «функция ***f(x)*** является „о“ малым от функции ***g(x)*** в окрестности точки p» означает, что с приближением ***x*** к ***pf(x)*** уменьшается быстрее, чем ***g(x)*** (отношение |***f(x)***| / |***g(x)***| стремится к нулю).

Определение

Пусть $f(x)$ и $g(x)$ — две функции, определенные в некоторой проколотой окрестности точки x_0, причем в этой окрестности g не обращается в ноль. Говорят, что:

- f является «О» большим от g при $x \to x_0$, если существует такая константа $C > 0$, что для всех x из некоторой окрестности точки x_0 имеет место неравенство

$$|f(x)| \leqslant C|g(x)|;$$

Определение

- f является «о» малым от g при $x \to x_0$, если для любого $\varepsilon > 0$ найдется такая проколотая окрестность U'_{x_0} точки x_0, что для всех $x \in U'_{x_0}$ имеет место неравенство

$$|f(x)| < \varepsilon |g(x)|.$$

Пояснение

Иначе говоря, в первом случае отношение $\frac{|f|}{|g|} \leqslant C$ в окрестности точки x_0 (т.е ограничено сверху), а во втором оно стремится к нулю при $x \to x_0$.

Обозначение

Обычно выражение «f является O большим (o малым) от g» записывается с помощью равенства $f(x) = O(g(x))$ (соответственно, $f(x) = o(g(x))$).

Это обозначение очень удобно, но требует некоторой осторожности при использовании (а потому в наиболее элементарных учебниках его могут избегать). Дело в том, что это не равенство в обычном смысле, а несимметричное отношение.

В частности, можно писать

$$f(x) = O(g(x)) \text{ (или } f(x) = o(g(x)) \text{)},$$

но выражения

$$O(g(x)) = f(x) \text{ (или } o(g(x)) = f(x) \text{)}$$

бессмысленны.

Другой пример: при $x \to 0$ верно, что

$$O(x^2) = o(x)$$

но неверно, что

$$o(x) = O(x^2).$$

При любом x верно

$$o(x) = O(x),$$

т.е. бесконечно малая величина является ограниченной, но неверно, что ограниченная величина является бесконечно малой:

$$O(x) = o(x).$$

Вместо знака равенства методологически правильнее было бы употреблять знаки принадлежности и включения, понимая $O(\,)$ и $o(\,)$ как обозначения для множеств функций, то есть, используя запись в форме

$$x^3 + x^2 \in O(x^3)$$

или

$$O(x^2) \subset o(x)$$

вместо, соответственно,

$$x^3 + x^2 = O(x^3)$$

и

$$O(x^2) = o(x)$$

Однако на практике такая запись встречается крайне редко, в основном, в простейших случаях.

При использовании данных обозначений должно быть явно оговорено (или очевидно из контекста), о каких окрестностях (одно- или двусторонних; содержащих целые, вещественные или комплексные числа и т. п.) и о каких допустимых множествах функций идет речь (поскольку такие же обозначения употребляются и применительно к функциям многих переменных, к функциям комплексной переменной, к матрицам и др.).

Литература по данной теме

- *Д. Грин, Д. Кнут.* Математические методы анализа алгоритмов. — Пер. с англ. — М.: Мир, 1987. — 120 с.
- *Джон Э. Сэвидж.* Сложность вычислений. — М.: Факториал, 1998. — 368 с.
- *В. Н. Крупский.* Введение в сложность вычислений. — М.: Факториал Пресс, 2006. — 128 с.