Модуль 7.6. Алгоритмы и их сложность

Класс однородных вычислительных задач мы будем называть ***проблемой*** (также используется понятие «***массовая задача***» или «***абстрактная задача***»). Индивидуальные случаи проблемы Q мы будем называть ***частными случаями*** проблемы Q. Мы можем, например, говорить о проблеме умножения матриц. Частные случаи этой проблемы суть пары матриц, которые нужно перемножить.

Более формально мы принимаем следующую абстрактную модель вычислительной задачи. Абстрактная задача есть произвольное бинарное отношение Q между элементами двух множеств — множества ***условий*** (или входных данных) I и множества ***решений*** S. Например, в задаче умножения матриц входными данными являются две конкретные матрицы-сомножители, а матрица-произведение является решением задачи. В задаче поиска кратчайшего пути между двумя заданными вершинами некоторого неориентированного графа[1] $G = (V, E)$ условием (элементом I) является тройка, состоящая из графа и двух вершин, а решением (элементом S) — последовательность вершин, составляющих требуемый путь в графе. При этом один элемент множества I может находиться в отношении Q с несколькими элементами множества S (если кратчайших путей между данными вершинами несколько).

Нам бы хотелось связать с каждым частным случаем проблемы некоторое число, называемое его ***размером***, которое выражало бы меру количества входных данных. Например, размером задачи умножения матриц может быть наибольший размер матриц-сомножителей. Размером задачи о графах может быть число ребер данного графа.

Решение задачи на компьютере можно осуществлять с помощью различных алгоритмов. Прежде чем подавать на вход алгоритма исходные данные (т. е. элемент множества I), надо договориться о том, как они представляются «в понятном для компьютера виде»; мы будем считать, что исходные данные закодированы последовательностью битов. Формально говоря, ***представлением*** элементов некоторого множества M называется отображение e из M во множество битовых строк. Например, натуральные числа 0, 1, 2, 3,… обычно представляют битовыми строками 0, 1, 10, 11, 100,… (при этом, например, $e(17) = 10001$).

Фиксировав представление данных, мы превращаем абстрактную задачу в ***строковую***, для которой входным данным является битовая строка, представляющая исходное данное абстрактной задачи. Естественно считать размером строковой задачи длину строки.

*Будем говорить, что алгоритм A **решает** строковую задачу за **время** $O(T(n))$, если на входном данном битовой строки длины n алгоритм работает время $O(T(n))$. Будем называть оценку $O(T(n))$ **асимптотической временной сложностью** алгоритма A.*

[1] О графах смотрите, например [1].

В качестве временной оценки работы алгоритма вместо общего числа шагов мы можем подсчитывать число шагов некоторого вида, таких как арифметические операции при алгебраических вычислениях, число сравнений при сортировке или число обращений к памяти.

Можно подумать, что колоссальный рост скорости вычислений, вызванный появлением нынешнего поколения компьютеров, уменьшит значение эффективных алгоритмов. Однако происходит в точности противоположное. Так как компьютеры работают все быстрее и мы можем решать все большие задачи, именно сложность алгоритма определяет то увеличение размера задачи, которое можно достичь с увеличением скорости машины.

Следуя [2], рассмотрим это более подробно. Допустим, у нас есть пять алгоритмов $A_1, A_2, \ldots, A_5$ с временными сложностями соответственно: $O(n)$, $O(n\log n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$.

Пусть единицей времени будет 1 мс и мультипликативные константы в точных оценках временной сложности во всех алгоритмах равны 1. Тогда алгоритм A_1 может обработать за 1 с вход размера 1000, в то время как A_5 — вход размера не более 9. В таблице 1 приведены размеры задач, которые можно решить за 1 с, 1 мин и 1 ч каждым из этих пяти алгоритмов.

Таблица 1 – Границы размеров задач, определяемые скоростью роста сложности

Алгоритм	**Асимптотическая временная сложность**	**Максимальный размер задачи**		
		1 с	1 мин	1 ч
A_1	$O(n)$	1000	6×10^4	$3{,}6 \times 10^6$
A_2	$O(n\log n)$	140	4893	$2{,}0 \times 10^5$
A_3	$O(n^2)$	31	244	1897
A_4	$O(n^3)$	10	39	153
A_5	$O(2^n)$	9	15	21

Предположим, что следующее поколение компьютеров будет в 10 раз быстрее нынешнего. В таблице 2 показано, как возрастут размеры задач, которые мы сможем решить благодаря этому увеличению скорости.

Таблица 2 – Эффект десятикратного ускорения

Алгоритм	**Асимптотическая временная сложность**	**Максимальный размер задачи**	
		до ускорения	**после ускорения**
A_1	$O(n)$	S_1	$10S_1$
A_2	$O(n\log n)$	S_2	Примерно $10S_2$ для больших S_2
A_3	$O(n^2)$	S_3	$3{,}16S_3$
A_4	$O(n^3)$	S_4	$2{,}15S_4$
A_5	$O(2^n)$	S_5	$S_5 + 3{,}3$

Заметим, что для алгоритма A_5 десятикратное увеличение скорости увеличивает размер задачи, которую можно решить, только на три, тогда как для алгоритма A_3 размер задачи более чем утраивается.

Вместо эффекта увеличения скорости рассмотрим теперь эффект применения более действенного алгоритма. Вернемся к таблице 1. Если в качестве основы для сравнения взять 1 мин, то, заменяя алгоритм A_4 алгоритмом A_3, можно решить задачу в 6 раз большую, а заменяя A_4 на A_2, можно решить задачу, большую в 125 раз. Эти результаты производят гораздо большее впечатление, чем двукратное улучшение, достигаемое за счет десятикратного увеличения скорости. Если в качестве основы для сравнения взять 1 ч, то различие оказывается еще значительнее. Отсюда мы заключаем, что асимптотическая скорость алгоритма служит важной мерой качественности алгоритма, причем такой мерой, которая обещает стать еще важнее при последующем увеличении скорости вычислений.

Несмотря на то, что основное внимание здесь уделяется порядку роста величин, надо понимать, что больший порядок сложности алгоритма может иметь меньшую мультипликативную постоянную, чем малый порядок роста сложности другого алгоритма. В таком случае алгоритм с быстро растущей сложностью может оказаться предпочтительнее для задач с малым размером — возможно, даже для всех задач, которые нас интересуют.

Список литературы по модулю

[1] Андерсон Д. Дискретная математика и комбинаторика / Д. Андерсон. — М. : Вильямс, 2003. — 960 с.

[2] Ахо А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. — М. : Мир, 1979. — 536 с.