Лекция 1.8. Эффективность алгоритмов

Добрый день. В предыдущих лекциях мы рассмотрели принципиально разные способы реализации таких типов данных, как стек, очередь и приоритетная очередь. При этом используются разные структуры данных, либо массивы, либо сцепленные структуры: линейные списки, деревья. Соответственно, различаются и алгоритмы выполнения операций. Как сделать выбор, какую реализацию лучше использовать при решении конкретной задачи? Очевидно, надо применить ту реализацию, которая в данном случае более эффективна. А как определить эту эффективность?

Как мы знаем, в программировании выделяют два основных показателя эффективности: эффективность по памяти (какой объем памяти расходуется для хранения данных и для выполнения операций) и эффективность по времени (то есть скорость выполнения операций). Как правило, решающим фактором является время выполнения, поэтому мы будем в основном говорить об этом показателе.

Время выполнения единичной операции на современных компьютерах настолько мало, что его смешно принимать во внимание. Однако если эти операции надо выполнять много раз и над большим массивом данных, картина резко меняется. И тут очень важно понять, насколько быстро увеличивается время выполнения операций при увеличении объема данных.

Одно дело, если, скажем, при увеличении размера массива в 10 раз время его обработки тоже увеличивается в 10 раз. С этим можно смириться, если это проблема, то надо найти более мощный компьютер. На графике показано, как в этом случае изменяется время работы T, когда размер массива n увеличивается от 1 до 10.

Если в 100 раз – это похуже. Чтобы показать изменение T, пришлось изменить масштаб по вертикальной оси. Но время работы при этом остается в пределах достижимого.

Но если время работы, например, удваивается при увеличении размера массива не «в разы», а просто на единицу, то такой алгоритм вообще непригоден для применения на практике. Еще раз изменив масштаб графика, мы можем видеть, насколько этот случай хуже двух предыдущих.

Еще убедительнее становится различие, если проследить, какое время потребуется при значениях размера до 50. На графике видно, что третья кривая быстро становится похожа на вертикальную линию, стремящуюся вверх, к бесконечности.

Чтобы выразить скорость роста времени в зависимости от размера данных, используется математическое понятие «О-большое». Вот формальное определение: (*см. слайд*). Давайте поясним, как это понимать.

Посмотрим на пример. Пусть нам известно, что время работы T(n) выражается функцией: (*см. слайд*).

Возьмем простую функцию $g(n)=n^3$.

Если мы раскроем скобки в выражении для T(n) и рассмотрим отношение T(n) к g(n), то нетрудно доказать, что при больших n это

отношение стремится к константе. Из всех слагаемых числителя останется только то, которое содержит n^3, остальные стремятся к 0 при увеличении n.

Но это значит, что T(n) есть O-большое от g(n).

То есть, поскольку $g(n) = n^3$, T(n) есть O-большое от n^3.

Теперь вместо громоздкой формулы мы имеем очень простую и понятную оценку времени работы.

Можно сформулировать два полезных правила упрощения формул для O-большого:

- во-первых, если формула состоит из нескольких слагаемых, то можно и нужно оставить только то слагаемое, которое содержит n в самой большой степени. Когда мы разделим все слагаемые на эту степень n, то все они, кроме старшего члена суммы, будут стремиться к 0 при больших n;

- во-вторых, если полученная формула содержит постоянный множитель, не равный 0, то его можно отбросить. В определении O-большого требуется только, чтобы отношение функций было ограничено ненулевой константой, а конкретное значение константы роли не играет.

Насколько оценки «до O-большого» полезны для практики?

Строго говоря, эти оценки асимптотические, то есть они верны только в пределе, когда n стремится к бесконечности. На самом деле, почти всегда оценки O-большого хорошо описывают поведение функций при конечных, но достаточно больших значениях n.

Надо понимать, что эти оценки характеризуют не точное время работы, а скорость роста времени при увеличении размера данных. Однако этого достаточно, чтобы понять:

- чего стоит данный алгоритм,
- насколько он лучше или хуже других алгоритмов,
- и можно ли его вообще успешно применять в реальных задачах.

Итак, что мы узнали на этой лекции?

Что эффективность алгоритма оценивается количеством требуемой памяти и временем работы.

Что время работы алгоритма принято оценивать как порядок роста времени «с точностью до O-большого».

Что такие оценки позволяют сравнить алгоритмы по скорости работы и понять, насколько алгоритм пригоден для практического применения.

На этом лекция закончена. Благодарю за внимание.