Лекция 1.9. Еще об эффективности

№ слайда	Текст
1	Добрый день. На прошлой лекции мы не успели сказать всё важное, что надо знать об оценках эффективности алгоритмов. Давайте продолжим.
2	Вот довольно неожиданный вопрос. В каких единицах следует измерять время, оценивая эффективность? Казалось бы, какой вопрос? Единицы времени – это секунды, миллисекунды и так далее. Однако реальное время работы программы зависит, кроме используемого алгоритма, еще от слишком многих факторов. Это мощность процессора, тип и объем памяти, используемый язык и система программирования, операционная система, мастерство программиста. Можно найти и еще немало факторов. Поэтому в качестве условной меры «как-бы-времени» оценивают количество характерных для алгоритма операций. Это могут быть, например, присваивания, сравнения, вызовы функций и другие операции, часто используемые в исследуемом алгоритме. Такой подход допустим, поскольку на самом деле оценка «О-большое» выражает скорость роста, которая не зависит от единиц измерения.

№ слайда	Текст
3	С какими же значениями оценок «до О-большого» приходится иметь дело чаще всего? Перечислим сначала более или менее «хорошие», приемлемые оценки, в порядке возрастания. O(1) – константная оценка. Это значит, что время работы не зависит от размера данных. Например, такую оценку имеют все операции со стеком или очередью, за исключением только случая, когда переполняется массив, используемый для представления этих структур. Для представления стека или очереди сцепленным списком время операций константно всегда. O(log(n)) – логарифмическая оценка. При этом неважно, по какому основанию берется логарифм, потому что логарифмы по разным основаниям отличаются друг от друга константным множителем, а его, как мы знаем, можно отбрасывать. Логарифмическую оценку имеют, например, операции вставки и извлечения из приоритетной очереди, реализованной как пирамида. O(n) – линейная оценка. Она обычно получается в тех случаях, когда алгоритм требует выполнить один или несколько раз проход по одномерному массиву. Но только если число проходов не зависит от n. Например, такую оценку имеют процедуры отыскания суммы элементов массива, максимального или минимального элемента, вставки или удаления элемента в начале массива и многие другие. O(n·log(n)). Эта формула может показаться слишком специальной, но на самом деле есть большое количество важных алгоритмов, которые имеют такую оценку. Логарифм – это функция, которая, хотя и стремится к бесконечности при больших n, но стремится медленнее, чем любая положительная степень n, так что эта оценка лишь ненамного хуже линейной. $O(n^2)$ – квадратичная оценка. Один их случаев такой оценки – процедуры, требующие рассмотреть по отдельности каждую пару элементов массива. Или другие алгоритмы, использующие двойной цикл по n значениям параметра. $O(n^k)$ при $k \geq 1$ – общий случай полиномиальной оценки. Линейная и квадратичная оценки – частный случай полиномиальной. Понятно, что чем больше k, тем медленнее алгоритм. Но, впрочем, алгоритмы с большими значениями k встречаются редко.

№ слайда	Текст
4	Теперь о плохом. Есть немало задач, для решения которых известны только экспоненциальные алгоритмы с оценкой $O(a^n)$ при $a > 1$. Это тот случай, когда увеличение размера данных n ***на*** одну или несколько единиц вызывают увеличение времени решения ***в*** несколько раз. Такие алгоритмы обычно требуют для обработки значительных объемов данных несусветно огромного времени, которое тратится на перебор всех возможных комбинаций параметров. Еще хуже оценка в виде факториала O(n!). Такую оценку имеют алгоритмы, которые перебирают все возможные перестановки из n значений. На практике, алгоритмы с такими оценками обычно неприменимы. В подобных случаях нередко приходится отказываться от точного решения и использовать приближенные методы.
5	Еще один существенный момент. Говоря об оценках времени, как правило, следует уточнять, среднее или максимальное время работы имеется в виду? Дело в том, что время работы алгоритма зависит не только от размера данных, но и от их конкретного набора. При этом у разных алгоритмов могут быть свои «любимые» и «нелюбимые» данные. Например, существует много алгоритмов сортировки массива, то есть расположения элементов массива в порядке возрастания (или убывания). Для одних алгоритмов самый лучший случай – если элементы массива уже были отсортированы, а самый худший – если они были расположены в обратном порядке. Но для других алгоритмов самый худший случай – это не обратный порядок, а полный беспорядок, случайное расположение элементов. А есть и такие алгоритмы, для которых абсолютно безразлично исходное расположение элементов, время работы будет одно и то же в любом случае. Максимальное время работы Tмакс(n) – это время работы при самом неудачном для данного алгоритма наборе данных размера n. Среднее время Tср(n) – это время работы, усредненное по всем возможным наборам данных размера n. Какая из этих характеристик алгоритма важнее? Они обе важны. Оценка максимального времени дает гарантию, что алгоритм никогда не будет работать слишком долго. Это важно, например, при работе в режиме реального времени, где задержка сверх определенного предела может привести к неприятным последствиям. Оценка среднего времени показывает среднюю эффективность алгоритма при его многократном применении. Она указывает, насколько удачен этот алгоритм для частого применения.

№ слайда	Текст
6	Подведем итоги. Мы узнали, что эффективность по времени не измеряется в единицах времени (слишком много других факторов влияют на программную реализацию алгоритма). Вместо этого при расчете порядка роста времени с точностью до О-большого используют такие условные единицы, как количество характерных операторов. Что практическую ценность имеют алгоритмы с оценкой, не более, чем полиномиальной. Алгоритмы с экспоненциальной или более того оценкой работают, как правило, только для маленьких, «игрушечных» примеров. Исключения есть, но они очень редки. Что важную роль играют оценки как среднего времени, так и максимального. Они позволяют раскрыть разные стороны эффективности алгоритмов.