Лекция 2. Анализ алгоритмов. Рост функции. Асимптотические обозначения

Казахский национальный университет имени аль-Фараби

Кафедра компьютерных наук

Алгоритмы и структуры данных

Лектор: старший преподаватель, PhD, Кенжебек Ержан Ғалымжанұлы

Цель и план лекции

Цель лекции

Освоить модель вычислений RAM и научиться оценивать эффективность алгоритмов независимо от реализации с помощью асимптотических обозначений O, Ω и Θ.

План лекции

1. Модель вычислений RAM
2. Анализ наилучшего, наихудшего и среднего случая
3. Рост функций и порядок роста времени работы
4. Асимптотические обозначения O, Ω, Θ
5. Графическая интерпретация обозначений

Анализ алгоритмов (введение)

- Алгоритмы являются принципиально важным компонентом информатики, т. к. их изучение не требует использования языка программирования или компьютера. Это означает необходимость в методах, позволяющих сравнивать эффективность алгоритмов, не прибегая к их реализации. Самыми значимыми из этих инструментов являются модель вычислений RAM и асимптотический анализ сложности наихудших случаев.
- Для оценки производительности алгоритмов применяется асимптотическая нотация.

Модель вычислений RAM

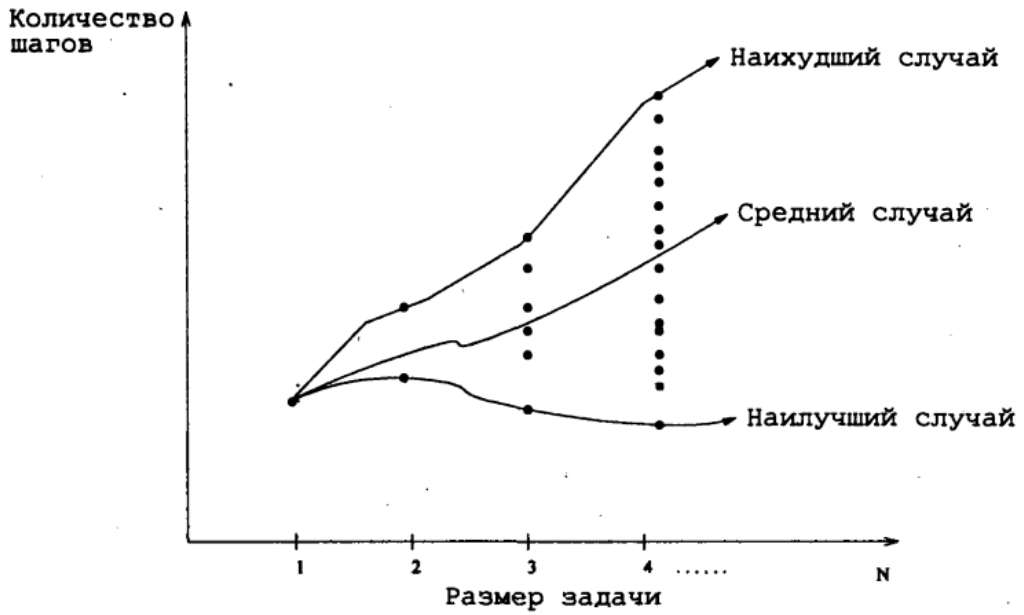
- **Разработка машинно-независимых алгоритмов** основывается на гипотетическом компьютере, называющемся машиной с произвольным доступом к памяти (Random Access Machine) или RAM-машиной. В рассматриваемую RAM модель входят те команды, которые обычно можно найти в реальных компьютерах. Согласно этой модели наш компьютер работает таким образом:
  - для исполнения любой простой операции (+, *, -, =, if, call) требуется ровно один временной шаг;
  - циклы и подпрограммы не считаются простыми операциями, а состоят из нескольких простых операций. Нет смысла считать подпрограмму сортировки одношаговой операцией, т. к. для сортировки 1 000 000 элементов потребуется определенно намного больше времени, чем для сортировки десяти элементов. Время исполнения цикла или подпрограммы зависит от количества итераций или специфического характера подпрограммы;
  - каждое обращение к памяти занимает один временной шаг. Кроме этого, наш компьютер обладает неограниченным объемом оперативной памяти. Кэш и диск в модели RAM не применяются

Модель RAM

- Время исполнения алгоритма в RAM-модели вычисляется по общему количеству шагов, требуемых алгоритму для решения данного экземпляра задачи. Допуская, что наша RAM-машина исполняет определенное количество **шагов/операций** за секунду, количество шагов легко перевести в единицы времени
- Может показаться, что RAM-модель является слишком упрощенным представлением работы компьютеров., В конце концов, на большинстве процессоров умножение двух чисел занимает больше времени, чем сложение, что не вписывается в первое предположение модели. Второе предположение может быть нарушено удачной оптимизацией цикла компилятором или гиперпотоковыми возможностями процессора. Наконец, время обращения к данным может значительно разниться в зависимости от расположения данных: в кэше, в оперативной памяти или на диске. Таким образом, по сравнению с настоящим компьютером, все три основные допущения для RAM-машины неверны
- Тем не менее, несмотря на эти несоответствия настоящему компьютеру, RAM-модель является **превосходной** моделью для понимания того, как алгоритм будет работать на настоящем компьютере. (поведение как у реального компьютера и простота исп.)

Анализ сложности наилучшего, наихудшего и среднего случая

- С помощью RAM-модели можно подсчитать количество шагов, требуемых алгоритму для исполнения любого экземпляра задачи. Но чтобы получить общее представление о том, насколько хорошим или плохим является алгоритм, нам нужно знать, как он работает со всеми экземплярами задачи. Чтобы понять, что означает наилучший, наихудший и средний случай сложности алгоритма (т. е. время его исполнения в соответствующем случае), нужно рассмотреть исполнение алгоритма на всех возможных экземплярах входных данных.
- Сложности в наихудшем, наилучшем и среднем случае (**зависит от количества шагов для обработки любого входного экземпляра размером n**)



Введение в асимптотические обозначения

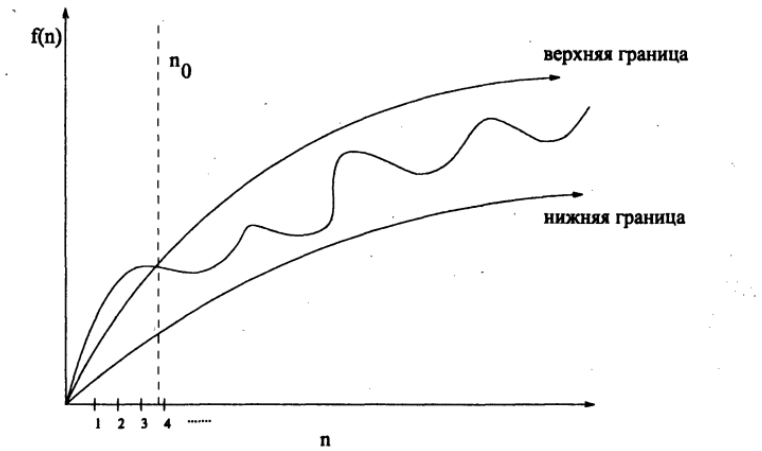
- Рассматривая входные данные достаточно больших размеров для оценки только такой величины, как **порядок роста** времени работы алгоритма, мы тем самым **изучаем асимптотическую эффективность алгоритмов.** Это означает, что нас интересует только то, как время работы алгоритма растет с увеличением размера входных данных. Обычно алгоритм, более эффективный в ассимптотическом смысле, будет более производительным для всех входных данных, за исключением некоторых маленьких.

Асимптотические обозначения (для чего нужен?)

- Временную сложность наилучшего, наихудшего и среднего случая для любого алгоритма можно представить как числовую функцию от размеров возможных экземпляров задачи. Но работать с этими функциями очень трудно, т. к. они обладают такими свойствами:
  - **являются слишком волнистыми**. Точная функция временной сложности любого алгоритма вполне может иметь неровный график с небольшими выпуклостями и впадинами. Границы действительны при n> n0
  - **требуют слишком много информации для точного определения**. Чтобы сосчитать точное количество инструкций RAM-машины, исполняемых в худшем случае, нужно, чтобы алгоритм был расписан в подробностях полной компьютерной программы. Более того, точность ответа зависит от маловажных деталей кодировки (например, был ли употреблен оператор case вместо вложенных операторов if). Точный анализ наихудшего случая, например, такого:

$$T(n) = 13252n^2 + 4353n + 1312\lg n + 25220$$ (***информация: увеличивается квадратический***)

- Оказывается, намного легче работать с верхней и нижней границами функций временной сложности, используя для этого асимптотические обозначения (О-болышое и Омега-большое соответственно). **Асимптотические обозначения позволяют упростить анализ, поскольку игнорируют детали, которые не влияют на сравнение эффективности алгоритмов.**



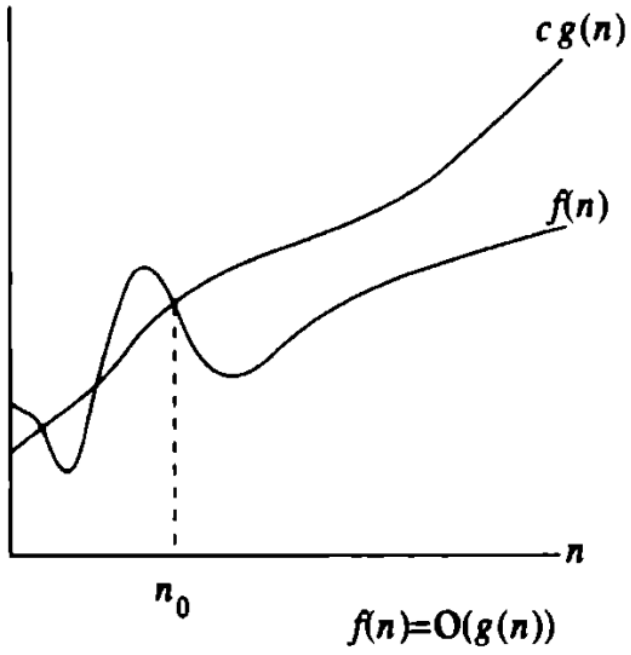
Асимптотические обозначения (O, Ω, θ)

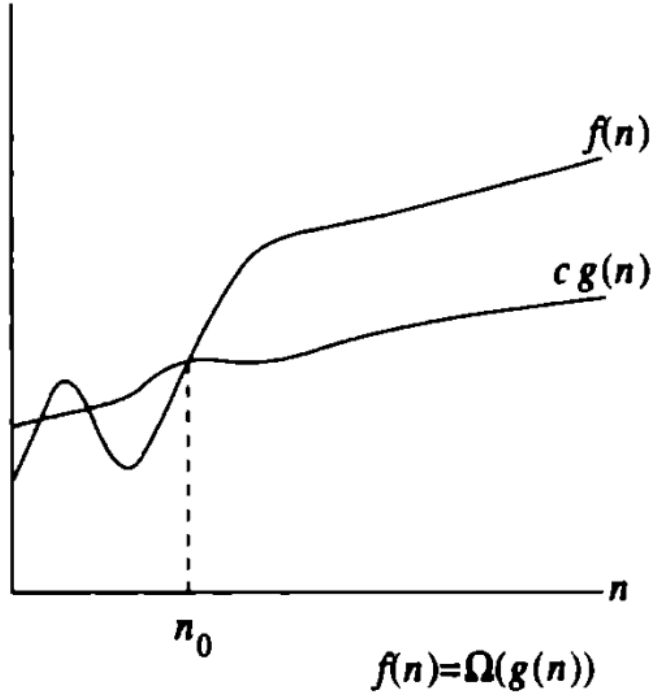
- В частности, в асимптотических обозначениях игнорируется разница между постоянными множителями. Например, в анализе с применением асимптотического обозначения функции:
  - *f(n) = 2(n) и g(n) = n* **являются идентичными.**
- Логично. При сравнении алгоритмов такие постоянные коэффициенты не принимаются во внимание. (**время алгоритма на разных языках; не дает инф. об алгоритме**).

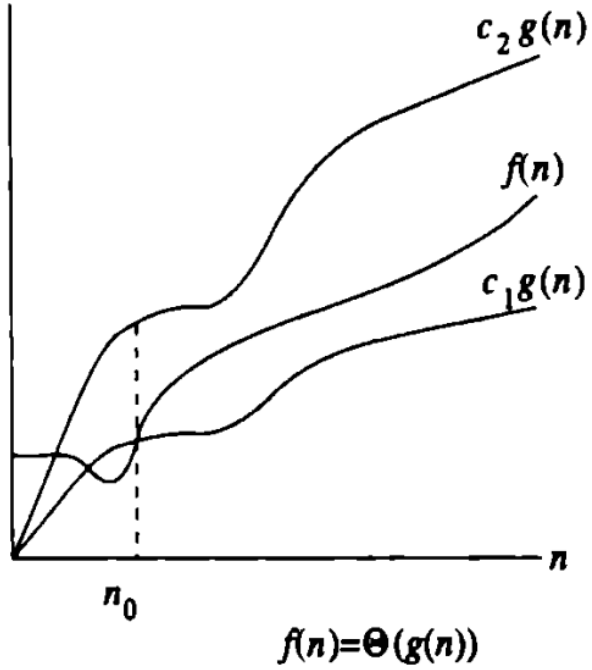
- Формальные определения, связанные с асимптотическими обозначениями, выглядят таким образом:
  - $f(n) = O(g(n))$ означает что функция *f(n)* **ограничена сверху** функцией *c * g(n)*. Существует константа ***c*** $f(n) \leq c * g(n)$
  - $f(n) = \Omega(g(n))$ означает что функция *f(n)* **ограничена снизу** функцией *c * g(n)*. Существует константа ***c*** $f(n) \geq c * g(n)$
  - $f(n) = \theta(g(n))$ означает что функция *f(n)* **ограничена сверху** функцией *c1 * g(n) и* **ограничена снизу** функцией *c2 * g(n)*. Существуют константы ***c1 и c2*** $f(n) \leq c1 * g(n)$ и $f(n) \leq c2 * g(n)$

Графическая иллюстрация асимптотических обозначений

- *O-большое (a),* Ω*−большое* (b), θ*−большое (c)*







Подведение итогов

- В каждом из этих определений фигурирует константа n_0, после которой эти определения всегда верны, нас не интересуют небольшие значения n, т. е. значения слева от n_0. **В конце концов, нам безразлично, что один алгоритм может отсортировать, скажем, шесть или восемь элементов быстрее, чем другой**. Мы ищем алгоритм для быстрой сортировки 10 000 или 1 000 000 элементов. В этом отношении асимптотические обозначения позволяют нам игнорировать несущественные детали и концентрироваться на общей картине.
- Анализ наихудшего случая и асимптотические обозначения являются инструментами, которые существенно упрощают задачу сравнения эффективности алгоритмов.

Задание

- Проверить неравенства и равенства функции с асимптотическими обозначениями О-большое, Ω – большое, θ – большое для n, n^2, n^3,
  - Функция: $3n^2 - 50n + 8$
  - Можно дать любые другие значения к константам ***c, n***

- **Понимание:** Асимптотические обозначения позволяют получить приблизительное представление о равенстве функций при их сравнении. Выражение типа $n^2 = O(n^3)$ может выглядеть странно, но его значение всегда можно уточнить, пересмотрев его определение в терминах верхней и нижней границ. Возможно, это обозначение будет более понятным, если в данном случае рассматривать **символ равенства (=)** как означающий "одна из функций, принадлежащих к множеству функций". Очевидно, что n^2 является одной из функций, принадлежащих множеству функций n^3.

Выводы

1. Модель RAM позволяет сравнивать эффективность алгоритмов, не прибегая к их реализации на конкретном языке и компьютере.
2. В модели RAM простая операция выполняется за один шаг, а циклы и подпрограммы состоят из нескольких простых операций.
3. Алгоритм анализируют для наилучшего, наихудшего и среднего случая; наиболее информативен анализ наихудшего случая.
4. Асимптотические обозначения описывают порядок роста времени работы при увеличении размера входа.

Контрольные вопросы

1. Опишите модель вычислений RAM. Какие допущения она содержит?
2. В чём различие между наилучшим, наихудшим и средним случаем? Приведите пример.
3. Почему на практике чаще анализируют наихудший случай?
4. Дайте определения обозначений O, Ω и Θ.
5. Почему при асимптотическом анализе отбрасывают константы и слагаемые низшего порядка?
6. Расположите по возрастанию скорости роста: n^2, $n \log n$, 2^n, n, $\log n$.

Список литературы

Основная литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е изд. — М.: Вильямс, 2013.
2. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд. — СПб.: БХВ-Петербург, 2011.
3. Левитин А. Алгоритмы: введение в разработку и анализ. — М.: Вильямс, 2006.

Дополнительная литература

4. Дасгупта С., Пападимитриу Х., Вазирани У. Алгоритмы. — М.: МЦНМО, 2014.
5. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. — Addison-Wesley, 1983.
6. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms. 4th ed. — MIT Press, 2022.

Электронные ресурсы

7. MIT OpenCourseWare, курс 6.006 «Introduction to Algorithms»: ocw.mit.edu