# Алгоритмы и их сложность

Алексей Владыкин

СПбГУ ИТМО

6 октября 2010

1. Определения алгоритма
2. Сложность алгоритмов
3. Примеры

- «Алгоритм — это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату» (А. Марков)
- «Алгоритм — это последовательность действий, либо приводящая к решению задачи, либо поясняющая почему это решение получить нельзя»
- «Алгоритм — это понятные и точные предписания исполнителю совершить конечное число шагов, направленных на решение поставленной задачи»

# Исполнитель алгоритма

- Человек
- Компьютер

- Конечный автомат
- Машина Тьюринга
- Машина Поста
- Регистровая машина

# Конечный автомат

- $A = (Q, \Sigma, \delta, q_0, F)$
- $Q$ — конечное множество состояний
- $\Sigma$ — конечное множество входных символов
- $\delta : Q \times \Sigma \to Q$ — функция переходов
- $q_0 \in Q$ — начальное состояние
- $F \subset Q$ — множество допускающих состояний

# Машина Тьюринга

- $M = (Q, \Sigma, \delta, q_0, B, F)$
- $Q$ — конечное множество состояний
- $\Sigma$ — конечное множество ленточных символов
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ — функция переходов
- $q_0 \in Q$ — начальное состояние
- $B \in \Sigma$ — пустой символ
- $F \subset Q$ — множество допускающих состояний

- Временна́я сложность:
  количество элементарных операций, совершаемых исполнителем

- Пространственная сложность:
  количество ячеек памяти, требуемых программе

# Асимптотические обозначения

- $f(x) = O(g(x))$:

$$\exists x_0, c: \ \forall x \geqslant x_0: \ 0 \leqslant f(x) \leqslant c \cdot g(x)$$

- $f(x) = \Omega(g(x))$:

$$\exists x_0, c: \ \forall x \geqslant x_0: \ 0 \leqslant c \cdot g(x) \leqslant f(x)$$

- $f(x) = \Theta(g(x))$:

$$\exists x_0, c_1, c_2: \ \forall x \geqslant x_0: \ c_1 \cdot g(x) \leqslant f(x) \leqslant c_2 \cdot g(x)$$

# Асимптотические обозначения

- $f(x) = o(g(x))$:

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 0$$

- $f(x) = \omega(g(x))$:

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \infty$$

# Градации сложности

- $O(1)$ — константная
- $O(\log n)$ — логарифмическая
- $O(n)$ — линейная
- $O(n^c)$ — полиномиальная (квадратичная, кубическая. . . )
- $O(c^n)$ — экспоненциальная
- $O(n!)$ — факториальная

# Практические соображения

- сложность по времени и по памяти;
- требования к решению;
- сложность реализации и отладки алгоритма;
- константа, скрытая в $O$-обозначении;
- оценки в среднем и в худшем случае.

# Задача 1

Вычислить сумму

$$1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

# Задача 1

Вычислить сумму

$$1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

Алгоритм 1.1:

- считаем каждый член суммы независимо, затем суммируем;
- время $O(n^2)$.

# Задача 1

Вычислить сумму

$$1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

Алгоритм 1.1:

- считаем каждый член суммы независимо, затем суммируем;
- время $O(n^2)$.

Алгоритм 1.2:

- $k$-й член суммы получаем из $(k-1)$-го делением на $k$;
- время $O(n)$.

# Задача 2

Вычислить $n$-е число Фибоначчи:

$$f_1 = 1, \quad f_2 = 1, \quad f_k = f_{k-2} + f_{k-1}$$

# Задача 2

Вычислить $n$-е число Фибоначчи:

$$f_1 = 1, \quad f_2 = 1, \quad f_k = f_{k-2} + f_{k-1}$$

Алгоритм 2.1:

- считаем рекурсивно «с конца» $(f_n, f_{n-1}, \dots)$;
- время $O(2^n)$.

# Задача 2

Вычислить $n$-е число Фибоначчи:

$$f_1 = 1, \quad f_2 = 1, \quad f_k = f_{k-2} + f_{k-1}$$

Алгоритм 2.1:

- считаем рекурсивно «с конца» $(f_n, f_{n-1}, \dots)$;
- время $O(2^n)$.

Алгоритм 2.2:

- считаем в цикле «с начала» $(f_1, f_2, \dots)$;
- время $O(n)$.

# Задача 3

Вычислить $a^n$.

# Задача 3

Вычислить $a^n$.

Алгоритм 3.1:

- считаем в цикле $a$, $a^2$, $a^3$, $\ldots$;
- время $O(n)$.

# Задача 3

Вычислить $a^n$.

Алгоритм 3.1:

- считаем в цикле $a$, $a^2$, $a^3$, $\ldots$;
- время $O(n)$.

Алгоритм 3.2:

- считаем, используя свойство $a^{2k} = (a^2)^k$;
- время $O(\log_2 n)$.

# Быстрое возведение в степень

Важные понятия:

- дихотомия — деление пополам;
- инвариант — условие, сохраняющее истинность в процессе работы алгоритма.

Алгоритм:

$$a^n = p \cdot b^q: \quad p = 1, \quad b = a, \quad q = n;$$

$$p \cdot b^q = \begin{cases} (p \cdot b) \cdot (b^2)^{(q-1)/2}, & \text{при нечетных } q, \\ p \cdot (b^2)^{q/2}, & \text{при четных } q. \end{cases}$$

Выход при $q = 0$.

# Задача 4

Вычислить значение полинома:

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

# Задача 4

Вычислить значение полинома:

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

Алгоритм 4.1:

- считаем каждый член независимо — время $O(n^2)$.

# Задача 4

Вычислить значение полинома:

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

Алгоритм 4.1:

- считаем каждый член независимо — время $O(n^2)$.

Алгоритм 4.2:

- считаем каждый член быстрым возведением в степень — время $O(n \log_2 n)$.

# Задача 4

Вычислить значение полинома:

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

Алгоритм 4.1:

- считаем каждый член независимо — время $O(n^2)$.

Алгоритм 4.2:

- считаем каждый член быстрым возведением в степень — время $O(n \log_2 n)$.

Алгоритм 4.3:

- считаем следующий член на основе предыдущего — время $O(n)$.

# Задача 5

Найти в массиве чисел элемент, встречающийся нечетное число раз.

# Задача 5

Найти в массиве чисел элемент, встречающийся нечетное число раз.

Алгоритм 5.1:

- брать каждый элемент и считать количество его вхождений — время $O(n^2)$.

# Задача 5

Найти в массиве чисел элемент, встречающийся нечетное число раз.

Алгоритм 5.1:

- брать каждый элемент и считать количество его вхождений — время $O(n^2)$.

Алгоритм 5.2:

- за один проход составить таблицу частот, по ней найти ответ — время $O(n)$.

# Задача 5

Найти в массиве чисел элемент, встречающийся нечетное число раз.

Алгоритм 5.1:

- брать каждый элемент и считать количество его вхождений — время $O(n^2)$.

Алгоритм 5.2:

- за один проход составить таблицу частот, по ней найти ответ — время $O(n)$.

Алгоритм 5.3:

- Прохорить все элементы массива — время $O(n)$.

# Задача 6

Проверить наличие числа $a$ в отсортированном массиве чисел.

# Задача 6

Проверить наличие числа $a$ в отсортированном массиве чисел.

Алгоритм 6.1:

- последовательно просмотреть все элементы массива;
- время $O(n)$.

# Задача 6

Проверить наличие числа $a$ в отсортированном массиве чисел.

Алгоритм 6.1:

- последовательно просмотреть все элементы массива;
- время $O(n)$.

Алгоритм 6.2:

- двоичный поиск (дихотомия);
- время $O(\log_2 n)$.

1. Даны натуральные $a$ и $d$. Вычислить частное $q$ и остаток $r$ при делении $a$ на $d$, не используя операции / и %.
2. Вычислить квадраты всех натуральных чисел от 1 до $n$, используя из арифметических операций только сложение и вычитание.
3. Разложить натуральное $n$ на простые множители.
4. Дано натуральное $n$. Подсчитать количество решений неравенства $x^2 + y^2 < n$ в натуральных числах.
5. Дана функция $f : \{1 \dots N\} \to \{1 \dots N\}$. Найти период последовательности $1, f(1), f(f(1)), \dots$
6. Дан отсортированный массив целых чисел длины $N$. Найти количество различных чисел среди элементов этого массива.
7. То же, если массив не отсортирован.
8. То же, если элементы массива — числа от 1 до $k$.
9. Даны два возрастающих массива. Найти количество общих элементов в этих массивах.

# Рекомендуемая литература



Ахо А., Хопкрофт Дж., Ульман Дж.
Структуры данных и алгоритмы. : Пер. с англ. : Уч. пос.
М.: Вильямс, 2000.— 384 с.: ил. // Глава 1


Кормен Т., Лейзерсон Ч., Ривест Р.
Алгоритмы: построение и анализ.
М.: МЦНМО, 1999. — 960 с., 263 ил. // Главы 1–4


Столяр С. Е., Владыкин А. А.
Информатика: Представление данных и алгоритмы.
СПб.: Невский Диалект; М.: БИНОМ. Лаборатория знаний, 2007. — 382 с.: ил.
// Глава А


Хопкрофт Д., Мотвани Р., Ульман Д.
Введение в теорию автоматов, языков и вычислений, 2-е изд.: Пер. с англ.
М.: Вильямс, 2002. — 528 с.: ил.


Шень А.
Программирование: теоремы и задачи.
М.: МЦНМО, 2004. — 296 с.: ил.