



# **Введение в математическую логику и теорию алгоритмов**

Лекция 13

**Алексей Львович Семенов**

# План

- Сложность. Подход теории алгоритмов
- Сложность объекта
- Теорема Колмогорова
- Что такое случайность?
- Теория алгоритмов (напоминание)
- Время вычисления
- Реально решаемые задачи
- Задачи, решаемые перебором
- Универсальная переборная задача
- Естественные переборные задачи, их универсальность
- Проблема перебора

# **Сложность. Подход теории алгоритмов**

# Сложность объекта

- да, да, да,... да (1 млн. раз)
- На экране нет миллиона «да».
- Есть описание объекта.
- Сложность объекта – минимальная длина его описания.

# Сложность объекта

- Что такое описание?
- Аргумент для
  - машины,
  - алгоритма,
  - вычислимой функции,дающих объект.
- Вычислимая функция (дающая объект по его описанию) – способ описания.
- Ее значения – это объекты, имеющие описание.
- Один объект может иметь много описаний, могут быть объекты без описаний.

Будем считать, что объекты и описания – двоичные слова.  $|y|$  – длина слова  $y$ ;

- *Сложность при данном способе описания  $f$*

$$K_f(x) = \min \{ |y| \mid f(y) = x \},$$

если такого  $y$  нет, то  $\min = \infty$ .

# Сложность объекта

- Способы описания бывают разные.
- Есть ли способ описания, дающий самые короткие описания?
- Нет (почти очевидно): возьмем любой способ,  $x$  - «сложный» при выбранном способе описании объект.

**Задача.** Существует  $f$  для которого  $f(0) = x$  и  $K_f(x) \leq 1$ .

- Можно ли получать описания, самые короткие «с точностью до» дополнительного слагаемого?
- Да. Точная формулировка:
- **Теорема Колмогорова.** Существует способ описания  $u$ , такой, что для любого способа описания  $f$  найдется такое число  $C$ , что для всякого объекта  $x$  выполнено:

$$K_u(x) \leq K_f(x) + C.$$

# Теорема Колмогорова

$$K_u(x) \leq K_f(x) + C$$

Д. Фиксируем некоторый вариант задания вычислимых функций алгоритмами, например, алгоритмами Маркова.

- Как мы видели, существует универсальная функция  $u$ : для всякой вычислимой функции  $f$ , если  $p$  – задание (алгоритм для)  $f$ , в алфавите  $\mathbf{B}$ , то для всех  $y$ :

$$f(y) = u(\langle p, y \rangle).$$

- Возьмем произвольное  $x$  и такое  $y$ , что  $f(y) = x$ . Тогда:

$$u(\langle p, y \rangle) = x$$

$K_u(x) \leq |\langle p, y \rangle|$ ,  $y$  можно взять самым коротким.

- Осталось доказать, что  $|\langle p, y \rangle| \leq |y| + C$ , где  $C$  может зависеть от  $p$ , но не от  $y$ .

**Задача.** Каким нужно взять кодирование пар?

**Задача.** Может ли код быть короче, чем  $|p| + |y|$ ?

# Андрей Николаевич Колмогоров

25.04.1903 – 20.10.1987





# Применение сложности объектов (колмогоровской сложности)

- Случайность
- Бросание монеты
- 0110100101011100100101...
- 0101010101010101010101...
- Вторая последовательность неслучайна?
- Вероятность  
одинакова.
- Сложность  
разная.
- Последовательность случайна, если ее  
(колмогоровская) сложность – максимальна.
- Информация в одном объекте о другом...

# **Теория алгоритмов (напоминание)**

# Переход к следующему в ансамбле

Свойство всякого ансамбля:

- Существует объект  $I$  (начальный, например – пустое слово) и
- действие  $S$  (следование), для которых  $\{I, S(I), SS(I)\dots\}$  – это весь ансамбль

• **Задачи** построения функции следования.

- Слова в унарном алфавите
- Двоичный алфавит
- Произвольный алфавит
- Цепочки слов
- Списки

# Кодирование

- Кодирование элементов ансамбля словами в двоичном алфавите.
- **Задача.** Действие, задающее взаимно-однозначное соответствие (обратное – тоже действие) для ансамблей:
  - Слов в любом алфавите
  - Цепочек слов
  - Списков.
- **Тезис Поста.** Тезис Черча – для кодов множеств в произвольных ансамблях

# Перечислимость. Эквивалентные определения

- Перечислимые множества
  - множества значений вычислимых функций
- **Задача.** = множества значений всюду определенных вычислимых функций
- **Задача.** = области определения вычислимых функций
- Идеи:
  - Множество значений  $\Rightarrow$  область определения  
Для всякого исходного данного ждем, пока оно появится как значение, последовательно перебирая все аргументы для перечисления (действие следования). Если дождемся...
  - Область определения  $\Rightarrow$  множество значений всюду определенной функции  
Исходное данное рассматривать как пару: <Исходное данное, Число шагов> (если не пара – ...). Если работа завершилась за это число шагов...

# Перечислимость и разрешимость

**Теорема.** Множество разрешимо тогда и тогда, когда оно и его дополнение перечислимы.

Д. Дано разрешимое множество

Перечислимость – множество значений

- Начинаем работу с проверки, лежит ли элемент в множестве.

Если лежит, то выдаем его в качестве результата, если нет...

Пусть множество и его дополнение перечислимы

- Идея: запустить *параллельно* функции перечисляющие множества и его дополнение.

- Уточнение. Делать в цикле *последовательно* по одному шагу одного и другого алгоритма. Когда один из них закончит работу, ответ есть.

**Задача.** Довершить рассуждения. В каком ансамбле?

**Задача.** Привести пример перечислимого неразрешимого множества (была функция, которую нельзя доопределить до всюду определенной вычислимой)

# Сведение вычислимости к перечислимости

**Задача.** Функция вычислима  $\Leftrightarrow$  ее график  
перечислим

# Перечислимость $\Leftrightarrow$ породимость

Перечислимость  $\Rightarrow$  породимость

- Проверка создания получается из действия преобразования

Породимость  $\Rightarrow$  перечислимость

- Всякое исходное данное рассматривается как вывод.
- Результат

Функция вычислима  $\Leftrightarrow$  ее график породим



# Перечислимость и логика отношений

- Множество общезначимых формул – область определения вычислимой функции
- Породность – исчисление логики отношений

# **Сложность. Подход теории алгоритмов**

# Сложность вычислений

- *О. Сложность (временная) вычисления* – это число шагов вычисляющего алгоритма.
- Какими могут быть отдельные шаги? Какова «модель вычислений»?
- Например, алгоритмы Маркова или интуитивно представляемый компьютер.
- **Задача.** Есть ли алгоритм, быстро решающий данную задачу для заданного конечного множества исходных данных ?
- Обычно говорят об асимптотическом поведении сложности вычисления, и это почти всегда оказывается осмысленным.
- Бывает, что алгоритм очень просто описывается, но требует для своего выполнения много времени. «Много» может означать экспоненту от размера исходного данного или еще больше.

# *Реально решаемая задача*

- Сложность вычисления ограничена полиномом от размера исходного данного (длины двоичного слова). Класс **P**.
- Обычно, если задача реально решается, то коэффициенты и степени полиномов оказываются «небольшими».

# Задачи, решаемые перебором

- Типичная ситуация (считаем, что объекты – двоичные слова и их размер – это длина):

Задача о рюкзаке. Дано  $n+1$  натуральное число:

$a_1, \dots, a_n, b$ , можно ли составить из  $a_i$  сумму, равную  $b$ ?  
(Каждое  $a_i$  разрешается брать не более одного раза.)

только пары  $a_i \Rightarrow$  полиномиальное время – количество пар квадратичное, умножить на время проверки, и т.д.

- Берутся не пары, а подмножества – время перебора экспоненциально.
- **23, 11, 44, 29, 18, 32, 19, 35, 67** – из этих чисел требуется составить сумму, равную **100**.
- ?
- **Давайте попробуем**

**11, 18, 19, 23, 29**

# Алгоритмы для функций и отношений

- Задача здесь состояла в построении быстрого алгоритма для поиска подмножества индексов.
- Можно, однако, спрашивать о быстром алгоритме выяснения, есть ли такое подмножество.
- **Задача.** Имея быстрый (полиномиальный) алгоритм выяснения, есть ли такое подмножество, построить быстрый алгоритм который будет выдавать множество индексов.
- Мы будем рассматривать алгоритмы для отношений (алгоритмы распознавания).

# Задачи, решаемые перебором

Выполнимость. Дана формула логики высказываний.  
Выполнима ли она?

- Можно выяснить, перебирая все возможные наборы значений логических имен.
- Разных имен в формуле может быть, по порядку, например, корень квадратный от размера формулы.
- Экспонента может быть не от размера, а от корня квадратного от размера, но это не очень помогает.

**Задача** (аналогичная предыдущей). Умея проверять существование, строить объект.



# Задачи, решаемые перебором

## Общая формулировка

- Дано двуместное отношение  $R(x,y)$ , где  $x$  – исходное данное,  $y$  – перебираемое (подсказка, подтверждение).
- **Задача:** выяснить по данному  $x$ , существует ли  $y$ , для которого  $R(x,y)$ :

$$\exists y R(x,y),$$

причем:

- размер  $y$  ограничен заданным полиномом от размера  $x$ ,
- сложность вычисления  $R(x,y)$  ограничена полиномом от размера аргументов.

Отношение  $R$  и ограничивающие полиномы фиксированы для данной задачи.

- Задача о рюкзаке ( $y$  – цепочка номеров) и Выполнимость ( $y$  – цепочка значений) – таковы.
- **NP** – Класс всех задач, решаемых перебором.
- **N** Недетерминированность

# Универсальная переборная задача

- **Универсальный алгоритм**
- получает на вход  $x$  и  $y$ , понимает  $x$ , как описание конкретного алгоритма и применяет его к  $y$ .
- Время работы  $P_0$  не сильно отличается от времени работы  $x$ .
- Модифицированный универсальный алгоритм  $P_0$  задает вычислимую функцию (отношение)  $U_0$  :
- Если  $p$  – алгоритм, то  $U_0(\langle x, p \rangle, y)$  есть результат применения  $p$  к  $\langle x, y \rangle$ .
- Если первый аргумент не оказывается парой такого вида, то отношение  $U_0$  ложно.
- Отношение  $U_0$  может и не вычисляться за полиномиальное время: для разных  $p$  полиномы, ограничивающие время работы и длину  $y$ , могут иметь разную степень, как функции длины  $x$ .

# Универсальная переборная задача

- Еще модификация: отношение  $U$ , задаваемое, как  $U(\langle x, p, \ell \rangle, y)$ , где алгоритм, вычисляющий  $U$ , работает так же, как алгоритм  $p$ , при этом ограничивает время работы этого алгоритма  $p$  длиной слова – третьего элемента тройки.  
(Если первый аргумент  $U$  – не такая тройка, то  $U = L$ , если  $p$  не завершает работу за данное время, тоже  $L$ .)
- $U$  имеет полиномиальную сложность.
- $\exists y U(z, y)$  – универсальная переборная задача.

# Универсальность

- Предположим, что мы нашли алгоритм, позволяющий вычислять отношение  $\exists y U(z, y)$  за полиномиальное время.
- Этот алгоритм:
- НЕ перебирает  $y$  и
- НЕ применяет  $p_0$ ,
- а делает что-то совсем другое.
- Тогда мы сможем и любую переборную задачу, заданную алгоритмом  $p$  и соответствующими полиномиальными ограничениями, решать за полиномиальное время, сооружая каждый раз тройку  $\langle x, p, \ell \rangle$ , (и при этом не имитируя работу алгоритма с описанием  $p$  и не перебирая  $y$ ).

# Естественные переборные задачи

- Предположим, что мы можем решать задачу о рюкзаке (в формулировке «найдется ли...») или “Выполнимость” за полиномиальное время.
- Поможет ли это в решении других переборных задач?
- Да.

# Сведение к выполнимости

- Пусть имеется **Задача:  $\exists y R(x, y)$** , решаемая перебором, и отношение  $R$  задается алгоритмом  $r$ .
- Будем считать, что наш алгоритм  $r$  – это алгоритм Маркова.
- Построим исходное данное для задачи выполнимости, то есть формулу логики высказываний  $\Phi$ .

# Моделирование работы алгоритма

с помощью выполнимости формул логики высказываний.

Эту  $R(x, y)$ , отношение отношение  $R$  задается (полиномиальным) алгоритмом  $p$

Мы уже знаем алгоритм  $p$  и  $x$ , но, конечно, не знаем  $y$

Общая схема:

По  $p$  и  $x$  строим формулу логики высказываний

существование  $y$  и допускающего (проверяющего) вычисления  $p \Leftrightarrow$

существование выполняющего формулу набора значений логических имен

## Вычисление

Граница для числа шагов – полином от длины  $x$

Максимальная длина промежуточного результата ограничена полиномом от длины  $x$ . Дополним промежуточный результат «пробелами» до максимальной длины.

Будем кодировать результаты двоичными словами, все символы будут занимать отрезки равной длины.

Вычисление представлено матрицей, будем считать – квадратной.

При необходимости дополним строки и столбцы кодами пробелов.

Размер матрицы  $m$  на  $m$ .

# Моделирование работы алгоритма

Каждому элементу таблицы дадим имя (высказывания, состоящего в том, что этот элемент = 1)

В самой первой строке (соответствующей началу вычисления) соответствует паре  $\langle x, y \rangle$ , сначала код  $x$  потом код  $y$ , потом пробелы.

Как написать: «существует  $x, y$  и вычисление»?

- Существование  $y$  – выполнимость формулы
- Вычисление однозначно определяется  $x, y$

Как записать, что матрица представляет допускающее вычисление? Конъюнкция:

- Элементы первой строки, соответствующие символам  $x$ , равны кодам этих символов.
- Последняя непустая строка начинается с кода символа 1.
- Переход от каждой строки к следующей соответствует шагу работы алгоритма



# Моделирование работы алгоритма

Как записать, что две последовательных строки матрицы (назовем их верхняя и нижняя) соответствуют шагу работы алгоритма?

Применилась данная замена в данном месте

Левых частей предыдущих замен в верхней строке нет

Левая часть данной подстановки не встречается левее данного места в первой строке.

Вторая строка – результат применения замены.

Записываем

Дизъюнкция по выбору замены и места.

Число членов дизъюнкции – количество замен в алгоритме на  $m$

Один член дизъюнкции:

Конъюнкция утверждений, относящихся к значению символов:

Идущих до выбранного места

Начинающихся с выбранного места и относящихся к выбранной замене

Идущих после

# Моделирование работы алгоритма

**Задача.** Ввести логические имена и записать формулы подробнее

**Задача.** Оценить размер формулы в зависимости от размера  $u$ . (Моделируемый алгоритм уже фиксирован.)

**Задача.** Как использовать Выполнимость для решения Задачи о рюкзаке или иной переборной задачи?

# Моделирование работы алгоритма

с помощью выполнимости формул логики высказываний.

## Пример построения и оценки

• Фиксируем натуральное число  $n$  и слово

$\alpha = \alpha_1, \dots, \alpha_n$  в алфавите  $\mathbf{B}$ ;

Строим формулу  $\Phi$ , куда входят имена  $A = A_1, \dots, A_n$ ,

•  $(0, A_i) = \neg A_i$

•  $(1, A_i) = A_i$

$\Phi = (\alpha, A) = ((\alpha_1, A_1) \wedge \dots \wedge (\alpha_n, A_n)) \leftrightarrow$

«цепочка  $A$  значений имен есть слово  $\alpha$ »

**Длина  $\Phi$  меньше, чем  $(10 |\alpha|) \cdot \log(|\alpha|)$  (индексы  $A_i$ )**

# Проблема перебора

## Доказать, что $P \neq NP$

Проблема равенства классов  $P$  и  $NP$  входит [первой] в семь задач тысячелетия, за решение которой Математический институт Клэя назначил премию в миллион долларов США.

– Википедия

А. Н. Колмогоров, Л. Левин,

Л. Хачиян,

А. Разборов