Введение в математическую логику и теорию алгоритмов

Лекция 15

Алексей Львович Семенов

План

- **Сложность. Подход теории алгоритмов**
- Время вычисления
- Реально решаемые задачи
- Задачи, решаемые перебором
- Универсальная переборная задача
- Естественные переборные задачи, их универсальность
- Проблема перебора

Сложность объекта

- Было
- Сложность объекта – минимальная длина его описания.

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 с разными исходными данными. При конкретном исходном данном (или их конечном количестве) можно «запомнить ответ» и мгновенно его «выдать».

Реально решаемая задача

– Сложность вычисления ограничена полиномом от размера исходного данного (длины двоичного слова). Класс P.

• В реальных задачах коэффициенты и степени полиномов оказываются «небольшими».

• Бывает, что алгоритм очень просто описывается, но требует для своего выполнения много времени. Часто «много» означает экспоненту от размера исходного данного или еще больше.

Задачи, решаемые перебором

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

Задача о рюкзаке. Дано n+1 натуральное число: $a_0, a_1, \ldots, a_{n-1}, b$, можно ли составить из a_i сумму, равную b? (Каждое a_i разрешается брать не более одного раза.)

- Если бы нужно было перебирать только пары a_i, то мы бы нашли нужную комбинацию за полиномиальное время – число 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

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.

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 x.

Отношение R и ограничивающие полиномы фиксированы для данной задачи.

• Задача о рюкзаке (номера) и Выполнимость (цепочка) – таковы.

• NP – Класс всех задач, решаемых перебором.

• Недетерминированность

Универсальная переборная задача

- **Универсальный алгоритм** A_0
- получает на вход x, второй аргумент отношения – y, описание конкретного алгоритма A и применяет A к x и y.
- Время работы A_0 не сильно отличается от времени работы A.
- Универсальный алгоритм A_0 задает отношение $U_0(z,y)$ такое, что если p – описание алгоритма A, то $U_0(<x,p>,y)$ есть результат применения A к <x, y>. 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.

Универсальная переборная задача

- Модификация: отношение U, задаваемое, как

  U(<x, p, ℓ >, y), где алгоритм, вычисляющий U, работает так же, как A_0,
- но при этом его время работы ограничено длиной слова – третьего элемента тройки.
  (Если первый аргумент U – не такая тройка, то U – Л, если он не завершает работу за данное время, тоже Л.)
- U имеет полиномиальную сложность.
- $\exists$yU(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 A_0,
- а делает что-то совсем другое.
- Тогда мы сможем и любую переборную задачу решать за полиномиальное время, сооружая каждый раз тройку $< x, p, \ell >$, (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 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 А.
- Будем считать, что наш алгоритм – это алгоритм Маркова.
- Построим исходное данное для задачи выполнимости, то есть формулу логики высказываний Ф.

Моделирование работы алгоритма

с помощью выполнимости формул логики высказываний.

• Все слова кодируем в двоичном алфавите (может оказаться удобным иметь коды равной длины для всех букв).

• Фиксируем натуральное число *n*.

Слово $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ в алфавите **B**;

цепочка имен $A = A_0, A_1, \ldots, A_{n-1}$,

• $(0, A_i) = \neg A_i$

• $(1, A_i) = A_i$

$\Phi = (\alpha, A) = ((\alpha_0, A_0) \wedge (\alpha_1, A_1) \wedge \ldots \wedge (\alpha_{n-1}, A_{n-1})) – И \leftrightarrow$

«Значения из A составляют слово α»

Длина Ф меньше, чем (10 длин α)·log (дл α) (индексы)

Моделирование работы алгоритма

с помощью выполнимости формул логики высказываний

Слово $\beta = \beta_0, \beta_1, \ldots, \beta_{k-1}$ в алфавите **B**;

«Значения $A = A_0, A_1, \ldots, A_{n-1}$, составляют слово, в которое начиная с i–го места входит β»

$\Phi_i = ((\beta_0, A_i) \wedge (\beta_1, A_{i+1}) \wedge \ldots \wedge (\beta_{k-1}, A_{i+k-1}))$ – И

Длина Ф меньше, чем (10 длин β) ▪ log (дл α).

«Слово β входит в A: $\vee \Phi_i$, i= 0… n-I»

Длина формулы умножается на длину α

«Слово β входит в A»

«Первое вождение слова β в A начинается с i–го места»

Моделирование работы алгоритма

«Слова A и B совпадают до i–го места»

«Слово B получается из слова A заменой в нем вождения слова β, начиная с i–го места, на слово γ»

«Слово B получается из слова A заменой в нем первого вождения слова β, начиная с i–го места, на слово γ»

«Слово B получается из слова A за один не последний шаг применения фиксированного алгоритма»

Построение формулы

- Пусть n – длина исходного данного.
- Длина вычисления – ограничена полиномом от n (= полиному).
- Длина промежуточных результатов – ограничена полиномом от n (= полиному).
- Сопоставим каждому моменту вычисления и каждому номеру символа в слове (промежуточном результате) логическое имя.
- (Можно представлять себе матрицу.)
- Число логических имен – произведение полиномов.
- Формула строится для заданного алгоритма и исходного данного и означает:
- «Для заданного алгоритма и исходного данного значения имен описывают вычисление, заканчивающееся кодом 1»
- Выполнимость формулы означает существование подсказки и вычисления на ней.
- Как строится формула?
- Размер формулы?

Построение формулы Ф

Конъюнкция условий на:

- Первое слово

  – первым элементом пары является конкретное слово x: Конъюнкция утверждений $A_i \equiv 0$ или $A_i \equiv 1$, фиксирующих значения x. (На часть, соответствующую y, никаких ограничений.) Длина – не более квадрата (число членов конъюнкции – длина x, индексы, скобки…)

- Каждые два последовательных слова: P и Q.

  «Слово B получается из слова A за один не последний шаг применения фиксированного алгоритма»

- Последнее слово – это код 1 и нельзя применить ни одно из правил или на предпоследнем шаге применялось заключительное правило.

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