ЛИКБЕЗ
Лекция 2:
Еще немного теории сложности вычислений. Конечные автоматы.

Дмитрий Ицыксон

ПОМИ РАН

30 сентября 2007

План

1. Еще немного теории сложности вычислений
   - Примеры **NP**-полных задач
   - **LOGSPACE** $\subseteq$ **P** $\subseteq$ **NP** $\subseteq$ **PSPACE** $\subseteq$ **EXP**
   - Вероятностные алгоритмы. Классы **RP** и **BPP**
2. Конечные автоматы
   - Недетерминированные конечные автоматы
   - Регулярные выражения, автоматные грамматики
   - Pumping лемма

Литература

1. А. Китаев, А. Шень, М. Вялый. Классические и квантовые вычисления.
2. C.H. Papadimitriou. Computational complexity.
3. Б.К. Мартыненко. Языки и трансляции.
4. А. Шень. Программирование: теоремы и задачи.

Напоминание

- **P** — класс языков, для которых существует полином $p(n)$ и алгоритм, работающий время $O(p(n))$, распознающий этот язык.
- $L \in \mathbf{NP}$, если существует полиномиально ограниченное и полиномиально проверяемое отношение $R \subseteq \Sigma \times \Sigma$, что $L = \{x | \exists y : (x, y) \in R\}$.
- Язык L сводится к L', если по любому алгоритму, который распознает L' можно построить алгоритм, распознающий L, работающий не более, чем в полином раз хуже.
- Язык $L \in \mathbf{NP}$ называется **NP**-полным, если к нему сводится любой другой язык из **NP**.
- Теорема Кука-Левина: SAT — **NP**-полный язык.

3-SAT

3-SAT: язык выполнимых формул в 3-КНФ. Например:
$(x \vee y \vee \neg z) \wedge (\neg x \vee y \vee t) \wedge (\neg x \vee \neg y \vee \neg t)$.

Теорема. Язык 3-SAT **NP**-полный

Доказательство Сводим SAT к 3-SAT.

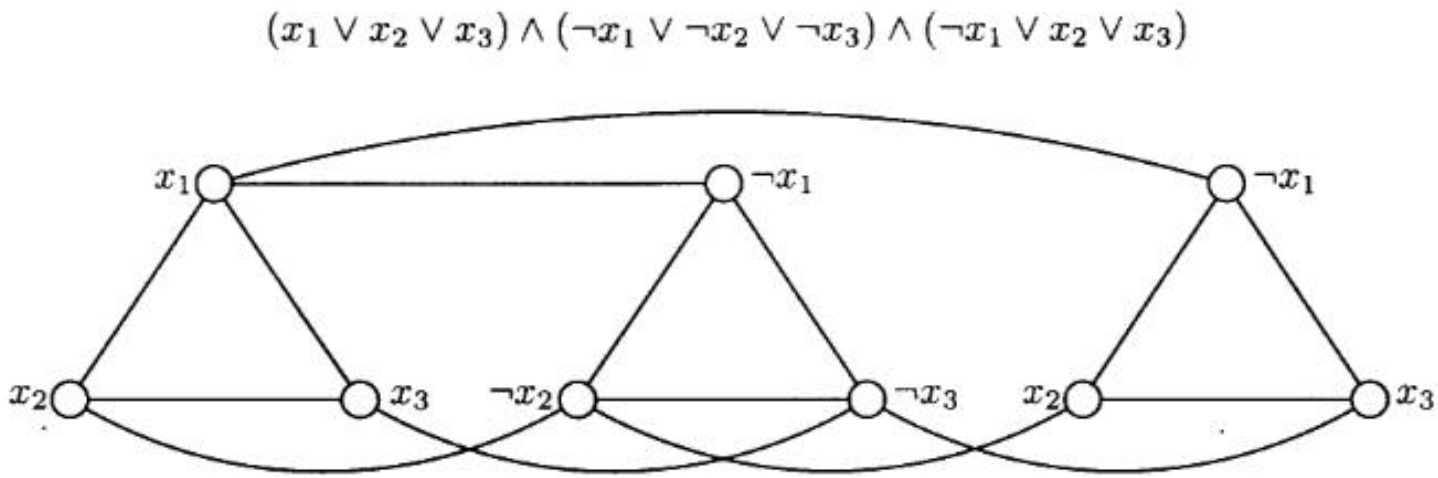
- Переписываем длинную дизъюнкцию $(x_1 \vee x_2 \vee x_3 \vee \ldots x_m)$ в несколько коротких
- Вместо $(x_1 \vee x_2 \vee \underbrace{x_3 \vee \cdots \vee x_m}_{=z_1})$ запишем
  $(x_1 \vee x_2 \vee z_1) \wedge (z_1 = (x_3 \vee \cdots \vee x_m))$

Продолжение доказательства

- Записываем в 3-КНФ равенства: $z = (y_1 \vee y_2 \vee \cdots \vee y_k)$
- Если $k = 2$, то это заведомо можно сделать за $O(1)$.
- Если $k > 2$, то перепишем $(z = y_1 \vee \underbrace{y_2 \vee \cdots \vee y_k}_{=z_2})$ в виде $(z = (y_1 \vee z_2)) \wedge (z_2 = (y_2 \vee \cdots \vee z_k))$.
- Последнее равенство содержит меньше переменных
- Эта процедура займет полиномиальное время

Независимое множество

Задача о независимом множестве: есть ли в данном графе k вершин, попарно несоединенных ребром? Сведем 3-SAT к этой задаче.



У формулы есть выполняющий набор, если в графе есть независимое множество такого же размера, что и количество клозов.

Независимое множество и CLIQUE

Теорема. Задача о независимом множестве **NP**-полна.
Следствие. Задача CLIQUE (по графу требуется определить, есть ли в нем полный подграф размера k) **NP**-полна.
Доказательство. Чтобы свести задачу о независимом множестве к задаче CLIQUE, достаточно инвертировать ребра: нарисовать ребро там, где его не было и стереть ребро там, где оно было.

PSPACE

Определение. $L \in$ **PSPACE**, если существует полином $p(n)$ и алгоритм, распознающий L, сложность которого по памяти $O(p(n))$.

Теорема. $NP \subset PSPACE$

Доказательство.

- Пусть $L \in NP$. Языку L соответствует отношение $R(x, y)$.
- $|y| < |x|^k$. Будем перебирать все y длины не более, чем $|x|^k$
- Принимаем x, если такой y нашелся, отвергаем иначе.
- Если для разных y переиспользовать старую память, то можно ограничится полиномиальной памятью.

EXP

Определение. $L \in \mathbf{EXP}$, если существует полином $p(n)$ и алгоритм, распознающий L, сложность которого по времени $O(2^{p(n)})$.

Теорема. $\mathbf{PSPACE} \subseteq \mathbf{EXP}$

Доказательство. (Аналогично $\mathbf{LOGSPACE} \subseteq \mathbf{P}$)

- Конфигурация MT: содержимое ленты, состояние, положение головки
- Конфигураций не больше, чем $|\Sigma|^{n^k}$ для некоторого k
- Конфигурации не могут повторяться
- Значит, время работы ограничено $|\Sigma|^{n^k} = O(2^{n^{k+1}})$

Что мы уже знаем?

$$\textbf{LOGSPASE} \subseteq \textbf{P} \subseteq \textbf{NP} \subseteq \textbf{PSPACE} \subseteq \textbf{EXP}$$

Известно:

- $\textbf{P} \subsetneq \textbf{EXP}$
- $\textbf{LOGSPACE} \subsetneq \textbf{PSPACE}$

Вероятностные алгоритмы

- Используют случайные биты (подбрасывают монетку)
- Могут иногда выдавать неправильный ответ
- Для каждого входа с какой-то вероятностью дают правильный ответ. Вероятность при этом берется только по случайным числам .

Пример вероятностного алгоритма

Задача. Дан двудольный граф $G(V_1, V_2, E \subset V_1 \times V_2), |V_1| = |V_2| = n, |E| = m$. Требуется определить, есть ли в этом графе совершенное паросочетание или нет. Т.е., можно ли его вершины разбить на пары так, чтобы в каждой паре вершины были соединены ребром.

- Построим символьную матрицу смежности графа G.
$$A_{i,j} = \begin{cases} x_{i,j}, \text{ если } (i,j) \in E \\ 0, \text{ если } (i,j) \notin E \end{cases}$$
- Определитель $\det A$ — это многочлен от m переменных $x_{i,j}$.
- $\det A = \sum_{\pi} \sigma(\pi) \prod_{i=1}^{n} A_{i,\pi[i]}$
- $\det A$ не нулевой многочлен $\iff$ в графе есть совершенное паросочетание.

Как проверить невырожденность символьной матрицы?

- Если бы матрица была невырожденной, ты мы бы посчитали определитель по методу Гаусса, приведя ее к диагональной.
- Лемма. $p(x_1, x_2, \ldots, x_m)$ — ненулевой многочлен, степень каждой переменной не превосходит 1. Тогда среди наборов $(a_1, a_2, \ldots, a_m) \in \{0, 1, 2, \ldots M-1\}^m$, не более, чем mM^{m-1} обнуляет p.

  Доказательство. Индукция по m.
  - $m = 1$. $p(x_1) = ax_1 + b$ имеет не более одного корня.
  - $m \mapsto m+1$.
    $p(x_1, x_2, \ldots, x_{m+1}) = q(x_1, x_2, \ldots, x_m)x_{m+1} + r(x_1, x_2, \ldots, x_m)$.
    Если $q(a_1, a_2, \ldots, a_m) \neq 0$, то найдется не более одного a_{m+1}, что $p(a_1, \ldots, a_{m+1}) = 0$. Наборов $(a_1, a_2, \ldots, a_m)$, обнуляющих q не больше, чем mM^{m-1}
  - Итого наборов $(a_1, a_2, \ldots, a_{m+1})$, обнуляющих p не более, чем $M^m + mM^{m-1} \cdot M = (m+1)M^m$.

Алгоритм

1. Выбираем $M = 2m$;
2. Выбираем $a_1, a_2, \ldots, a_m$ случайно из $\{0, 1, 2, \ldots M-1\}^m$;
3. Считаем по методу Гаусса определитель A, если вместо x_k поставить a_k.
4. Если определитель отличен от нуля, то выдать "паросочетание есть"
5. Если определитель равен нулю, то выдать "возможно нет"

Анализ

- Если паросочетания нет, то ответ будет "возможно нет"
- Если паросочетание есть, то с вероятностью $\geq \frac{1}{2}$, ответ будет: "паросочетание есть"

RP

Определение. Класс **RP** состоит из языков L, для которых существует полиномиальный по времени вероятностный алгоритм A:

- если $x \notin L$, то $A(x) = 0$
- если $x \in L$, то $\Pr\{A(x) = 1\} \geq \frac{1}{2}$

Комментарий

- Константу $\frac{1}{2}$ можно заменить на любую строго между 0 и 1.
- Если повторить алгоритм k раз и выдать 1, если хотя бы один из ответов был 1, то вероятность неверного ответа будет не больше, чем $\frac{1}{2^k}$.

$\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$

Лемма 1. $\mathbf{P} \subseteq \mathbf{RP}$

Доказательство. Алгоритм из $\mathbf{P}$ допускает ошибку с вероятностью 0.

Лемма 2. $\mathbf{RP} \subseteq \mathbf{NP}$

Доказательство.

- Пусть $L \in \mathbf{RP}$, A вероятностный алгоритм для языка L.
- Определим отношение R. $R = \{(x, r) | A_r(x) = 1\}$, где A_r использует строку r вместо случайных битов.
- R — полиномиально ограниченное и проверяемое.

Комментарий

RP-машины можно представлять себе, как недетерминированные, которые принимают вход, если хотя бы половина подсказок приводят в принимающее состояние.

BPP

Определение. Класс **BPP** состоит из языков L, для которых существует полиномиальный по времени вероятностный алгоритм A, такой что

- если $x \in L$, то $\Pr\{A(x) = 1\} \geq \frac{3}{4}$
- если $x \notin L$, то $\Pr\{A(x) = 0\} \geq \frac{3}{4}$

Комментарий

- Константу $\frac{3}{4}$ можно заменить на любую большую, чем $\frac{1}{2}$.
- Можно ли экспоненциально понизить вероятность ошибки повторением?

Понижение вероятности ошибки

Теорема. (Оценки Чернова) Положим $x_1, x_2, \ldots, x_n$ независимые случайные переменные, принимающие значения 1 и 0 с вероятностями p и $1-p$ соответственно. $X = \sum_{i=1}^{n} x_i$. Тогда для всех $0 \leq \theta \leq 1, \Pr\{X \geq (1+\theta)pn\} \leq e^{-\frac{\theta^2}{3}pn}$

- Положим $\theta = 1$.
- Повторим **BPP**-алгоритм n раз $x_i = 1$, если i-й ответ был ошибочный. Вероятность этого события равняется $p \leq \frac{1}{4}$.
- $\Pr\{\sum_{i=1}^{n} x_i \geq \frac{n}{2}\} \leq e^{-\frac{p}{3}n}$
- Итоговым ответом будет ответ, который встречался больше всего раз.

Конечные автоматы

Конечный автомат состоит из:

- Q — конечное множество состояний
- $q_0 \in Q$ — начальное состояние
- $F \subset Q$ — множество конечных состояний
- Σ — конечный входной алфавит
- $\delta : Q \times \Sigma \to Q$ — функция перехода

Автомат начинает работу в состоянии q_0. Он читает символы один за другим и переходит согласно функции перехода в новое состояние. Если, прочитав вход, автомат придет в конечное состояние, то он этот вход принимает, в противном случае вход отвергается.

Пример

Опишем конечный автомат, который принимает все строки, в которых содержится подстрока “ПОМИ”.

- $Q = \{q_0, q_\text{п}, q_\text{о}, q_\text{м}, q_\text{и}, q_f\}$, $F = \{q_f\}$
- $(q_0, \Sigma \setminus \{\text{П}\}) \mapsto q_0$
- $(q_0, \text{П}) \mapsto q_\text{п}$
- $(q_\text{п}, \Sigma \setminus \{\text{П,О}\}) \mapsto q_0$
- $(q_\text{п}, \text{О}) \mapsto q_\text{о}$
- $(q_\text{о}, \Sigma \setminus \{\text{П,М}\}) \mapsto q_0$
- $(q_\text{о}, \text{М}) \mapsto q_\text{м}$
- $(q_\text{м}, \Sigma \setminus \{\text{П,И}\}) \mapsto q_0$
- $(q_\text{м}, \text{И}) \mapsto q_f$
- $(q_{0,\text{п,о,м}}, \text{П}) \mapsto q_\text{П}$
- $(q_f, \Sigma) \mapsto q_f$

Недетерминированный конечный автомат

- Функция перехода неоднозначна: $\delta : Q \times \Sigma \to 2^Q$
- НКА принимает слово, если существует легальная последовательность шагов, приводящих после прочтения слова в конечное состояние.

Теорема. По любому НКА можно построит детерминированный конечный автомат, принимающий тот же язык.

Доказательство. У детерминированного автомата множеством состояний будут подмножества Q. $S = 2^Q, s \subset Q$. Начальное состояние $s_0 = \{q_0\}$. Правило перехода: $(s, a) \mapsto \bigcup_{q \in s} \delta(q)$.

Конечные состояния $F_s = \{s \in S | F \cap S \neq \emptyset\}$.

Регулярные выражения

Определим язык регулярных выражений.
Алфавитом этого языка будет некий алфавит Γ и не входящие в него символы $\Lambda, \epsilon, (,), *, |$.

- Буквы алфавита Γ — регулярные выражения;
- Символы Λ, ϵ — регулярные выражения;
- Если $A_1, A_2, \ldots A_n$ — регулярные выражения, то $(A_1 A_2 \ldots A_n)$ — регулярное выражение;
- Если $A_1, A_2, \ldots A_n$ — регулярные выражения, то $(A_1 | A_2 | \ldots | A_n)$ — регулярное выражение;
- Если A — регулярное выражение, то $A*$ регулярное выражение.

Смысл регулярных выражений

Каждому регулярному выражению мы сопоставим язык:

- Букве из Γ соответствует одноэлементный язык из этой буквы;
- Символу ϵ — пустой язык, Λ — язык из пустого слова;
- Выражению $(A_1 A_2 \dots A_n)$ соответствует конкатенация языков для $A_1, A_2, \dots, A_n$.
- Выражению $(A_1|A_2|\dots|A_n)$ соответствует объединение языков для $A_1, A_2, \dots, A_n$.
- Выражению $A*$ соответствует язык, слова которого можно разрезать на части, принадлежащие A.

Примеры регулярных выражений

- $(a|b)*$ — все слова из букв a и b;
- $(aa)*$ — все слова из четного числа букв a;
- $(\text{Г} * (\text{ПОМИ})\text{Г}*)$ — все слова, содержащие подстроку ПОМИ.

Автомат по выражению

Теорема. По любому регулярному выражению можно построить конечный автомат, который принимает язык, задаваемый этим выражением.

Доказательство. Индукция по построению выражения. Достаточно построить недетерминированный автомат.

- Для ϵ: $Q = F = \{q_o\}$; Для Λ: конечное состояние недостижимо;
- $a \in \Gamma$: $Q = \{q_0, q_1, q_f\}, F = \{q_f\}(q_0, a) \mapsto q_f, (q_0, \Gamma \setminus \{a\}) \mapsto q_1, (q_1, \Gamma) \mapsto q_1$;
- Покажем, как по автоматам A_1 и A_2 построить автомат, который примет язык, конкатенацию языков для A_1 и A_2. Добавим во все конечные состояния автомата A_1 правила, соответствующие начальному состоянию A_2.

Автомат по выражению (продолжение)

- Покажем, как по автоматам A_1 и A_2 построить автомат, который примет язык, объединение языков для A_1 и A_2. Добавим в начальное состояние автомата A_1 правило, соответствующее начальному состоянию автомата A_2.
- Покажем, как по автомату A, принимающему язык L, построить автомат, принимающий язык $L*$. Для этого сделаем новое начальное состояние q'_0 (старое оставим) Добавим все правила, которые были у старого, сделаем q'_0 конечным состоянием, кроме того во все конечные состояния добавим правила, какие есть у q'_0.

Регулярное выражение по автомату

Теорема. По любому конечному автомату можно построить реуглярное выражение, которое задает тот же язык, что принимает автомат.

Доказательство.

- Для НКА можно считать, что у него есть всего 1 конечное состояние. Его можно модифицировать следующим образом: добавить состояние q'_f, из всех состояний, откуда можно попасть в конечное, добавить аналогичный переход в q'_f. Добавить еще одно неконечное состояние q_∞, и все шаги из q'_f направить в q_∞, и из q_∞ все правила ведут в него самого.
- Пусть вершины НКА пронумерованы числами $1, 2, \ldots, k$. Вершина с номером 1 — начальная, вершина с номером k — единственная конечная.
- Обозначим через $D_{i,j,s}$ множество всех слов, которые можно прочитать, на пути из вершины i в j, если по пути можно заходить только в вершины с номерами $1, 2, \ldots, s$.

Регулярное выражение по автомату (продолжение)

- Язык, задаваемый автоматом $D_{1,k,k}$.
- Индукцией по s доказываем регулярность выражения $D_{i,j,s}$.
- База $s = 0$. $D_{i,j,0}$ либо пуст либо состоит из пустого слова, либо из буквы
- Как определить $D_{i,j,s+1}$? Отметим все заходы в состояние $s + 1$.
- $$D_{i,j,s+1} = D_{i,j,s} | (D_{i,s+1,s} D_{s+1,s+1,s} * D_{s+1,j,s})$$

Итого

- Языки задаваемые недетерминированными автоматами и детерминированными совпадают.
- Языки задаваемые автоматами и регулярными выражениями совпадают.
- Такие языки называют: автоматными или регулярными.

Автоматные грамматики

Пусть N — множество нетерминальных символов, а T — множество терминальных символов. $S \in N$ — начальный нетерминальный символ.
P — множество правил вида $A \mapsto qB$ или $A \mapsto q$, где $A, B \in N, q \in T$.
Правила можно применять друг за другом и остановится, если текущее слово не содержит нетерминальных символов.
Пример.
$N = \{S, A\}, T = \{a, b\}, P = \{S \mapsto aA, A \mapsto b, A \mapsto aS\}$. Можно выводить так: $S \mapsto aA \mapsto aaS \mapsto aaaA \mapsto aaab$.
Язык: состоит из нечетного числа букв a, за которой следует b.
Упражнение. Докажите, что автоматные грамматики задают в точности регулярные языки.

Pumping лемма

Лемма с помощью которой можно доказать, что язык не является автоматным.

Лемма. Для любого регулярного языка L существует число n, что любую строку $\alpha \in L, |\alpha| > n$ можно представить в виде $\alpha = uvw$, где $|v| > 1$ и для всех k слово $uv^k w \in L$.

Доказательство.

- Пусть n — число состояний детерминированного КА, принимающего L;
- Если $\alpha \in L, |\alpha| > 3n + 1$, то в процессе принятия слова α какое-то состояние q встретилось не менее 3-х раз. Обозначим строчку до первого вхождения в состояние q за u, строчку между первым и последним состоянием q за v, и оставшуюся часть за w.
- Так как q встретилось не менее 3-х раз, то $|v| > 1$.
- Очевидно, что строчка $uv^k w$ тоже будет приниматься.

Пример использования pumping леммы

Пример. Язык $L = \{ \underbrace{11..1}_{p \text{ штук}} \mid p - \text{ простое}\}$ не является регулярным.

Доказательство. Если бы он был регулярным, то нашлось бы такое n, что для $p > n$ $\underbrace{11..1}_{p \text{ штук}} = uvw, |v| > 1$ и для всех k $uv^k w \in L$. Пусть $|v| = d > 1$, $|u| + |w| = a$. Тогда при всех $k > 0$ число $kd + a$ должно быть простым. Выберем $k = d + a + 1$: $(d + a + 1)d + a = (d + a)(d + 1)$ — не является простым.

Задача. Является ли язык простых чисел, записанных в 10-й системе счисления автоматным?

Задачи

- Является ли язык $\{a^n b^n c^n | n \in \mathbb{N}\}$ регулярным?
- Является ли язык палиндромов регулярным?
- Докажите, что язык десятичных чисел, делящихся на 3 является регулярным.
- Докажите, что $\mathbf{P} \neq \mathbf{EXP}$.
- Докажите, что $\mathbf{NL} \subseteq \mathbf{P}$. Где $\mathbf{NL}$ — язык, принимаемый НМТ, использующей $O(\log n)$ памяти.