Конспект по теории сложности
III семестр
Современное программирование, факультет математики и компьютерных наук, СПбГУ
(лекции Гирша Эдуарда Алексеевича)

Тамарин Вячеслав

16 ноября 2020 г.

Оглавление

Глава 1

Введение в теорию сложности вычислений

Лекция 1: †

5 nov

1.0.1 Напоминания

Обсудим, что мы решаем.

Обозначение.

- Алфавит будет бинарный $\{0, 1\}$;
- Множество всех слов длины n : $\{0, 1\}^n$;
- Множество всех слов конечной длины $\{0, 1\}^*$;
- Длина слова x: $|x|$.

Определение 1

Язык (задача распознавания, decision problem) — $L \subseteq \{0, 1\}^*$.
Индивидуальная задача — пара, первым элементом которой является условие, а второй – решение; принадлежит $\{0, 1\}^* \times \{0, 1\}^*$.
Массовая задача — некоторое множество индивидуальных задач, то есть бинарное отношение на $\{0, 1\}^*$.

Определение 2

Будем говорить, что алгоритм **решает задачу поиска** для массовой задачи R, если для условия x он находит решение w, удовлетворяющее $(x, w) \in R$.
Можем сопоставить массовой задаче, заданной отношением R, язык

$$L(R) - \{x \mid \exists w \colon (x, w) \in R\}.$$

Пример 1.0.1 (Массовая задача и соответствующий язык)**.**

$$\widehat{\mathrm{FACTOR}} = \{(n, d) \mid d|n,\ 1 < d < n\}.$$

Здесь условием задачи является натуральное число n, а решением некоторый (не 1, и не n) числа n.
Данной задаче соответствует язык

$$L(\widehat{\mathrm{FACTOR}}) = \text{множество всех составных чисел}.$$

1.0.2 Детерминированная машина Тьюринга

Определение 3: Детерминированная машина Тьюринга

Детерминированная машина Тьюринга —

- конечный алфавит (с началом ленты и пробелом): $\Sigma = \{0, 1, \triangleright, \_\}$;
- несколько лент, бесконечных в одну сторону;
- читающие/пишущие головки, по одной на каждую ленту;
- конечное множество состояний, в том числе начальное (q_S/q_0), принимающее (q_Y/q_{acc}) и отвергающее (q_N/q_{rej});
- управляющее устройство (программа), содержащее для каждых $q, c_1, \dots c_k$ одну инструкцию вида
$$(q, c_1, \dots c_k) \mapsto (q', c_1', \dots c_k', d_1, \dots d_k),$$
где $q, q' \in Q$ — состояния, $c_i, c_i' \in \Sigma$ — символы, обозреваемые головками, $d_i \in \{\to, \leftarrow, \cdot\}$ — направления движения головок.

ДМТ **принимает** входное слово, если заканчивает работу в q_{acc}, и **отвергает**, если заканчивает в q_{rej}.

ДМТ M **распознает язык** A, если принимает все $x \in A$ и отвергает все $x \notin A$.
$$A = L(M).$$

Замечание. Обычно есть отдельная строка только для чтения, куда записаны входные данные, и строка только для вывода, куда нужно поместить ответ. Остальные строки будут рабочими.

Определение 4

Время работы машины M на входе x — количество шагов (применений инструкций) до достижения q_{acc} или q_{rej}.
Используемая память — суммарное крайнее правое положение всех головой на *рабочих* лентах.

Теорема 1.0.1. *Для любого $k \in \mathbb{N}$, работу ДМТ M с k рабочими лентами, работающую t шагов, можно промоделировать на ДМТ с двумя рабочими лентами за время $\mathcal{O}(t \log t)$, где константа $O(\dots)$ зависит только от размеров записи машины M).*

□

- Перестроим исходную МТ:
  - Запишем все ленты в одну строку по символу из всех лент по очереди.
  - Будем бегать «лентой по головке»: выровняем все ленты, чтобы головки стояли друг над другом и далее будем сдвигать нужную ленту.
  - Заметим, что двустороннюю ленту можно смоделировать на односторонней с увеличением количества операций в константу раз: разрезаем двустороннюю пополам и записываем элементы через один.
- Теперь поймем, как экономично сдвигать ленты в однострочной записи.

  Разобьем строку на блоки начиная от позиции головки в две стороны: справа блоки R_i, слева L_i. При этом $|L_i| = |R_i| = 2^i$. Раздвинем символы, заполняя пустоту специальными символами пустоты, так, чтобы в каждом блоке ровно половина элементов были пустыми.

  Далее будем поддерживать такое условие:

  1. В блоке либо информация, либо пусто, либо наполовину пусто
  2. L_i пустой, согда R_i полный
  3. L_i наполовину пустой, согда R_i наполовину полный
  4. L_i полный, согда R_i пустой

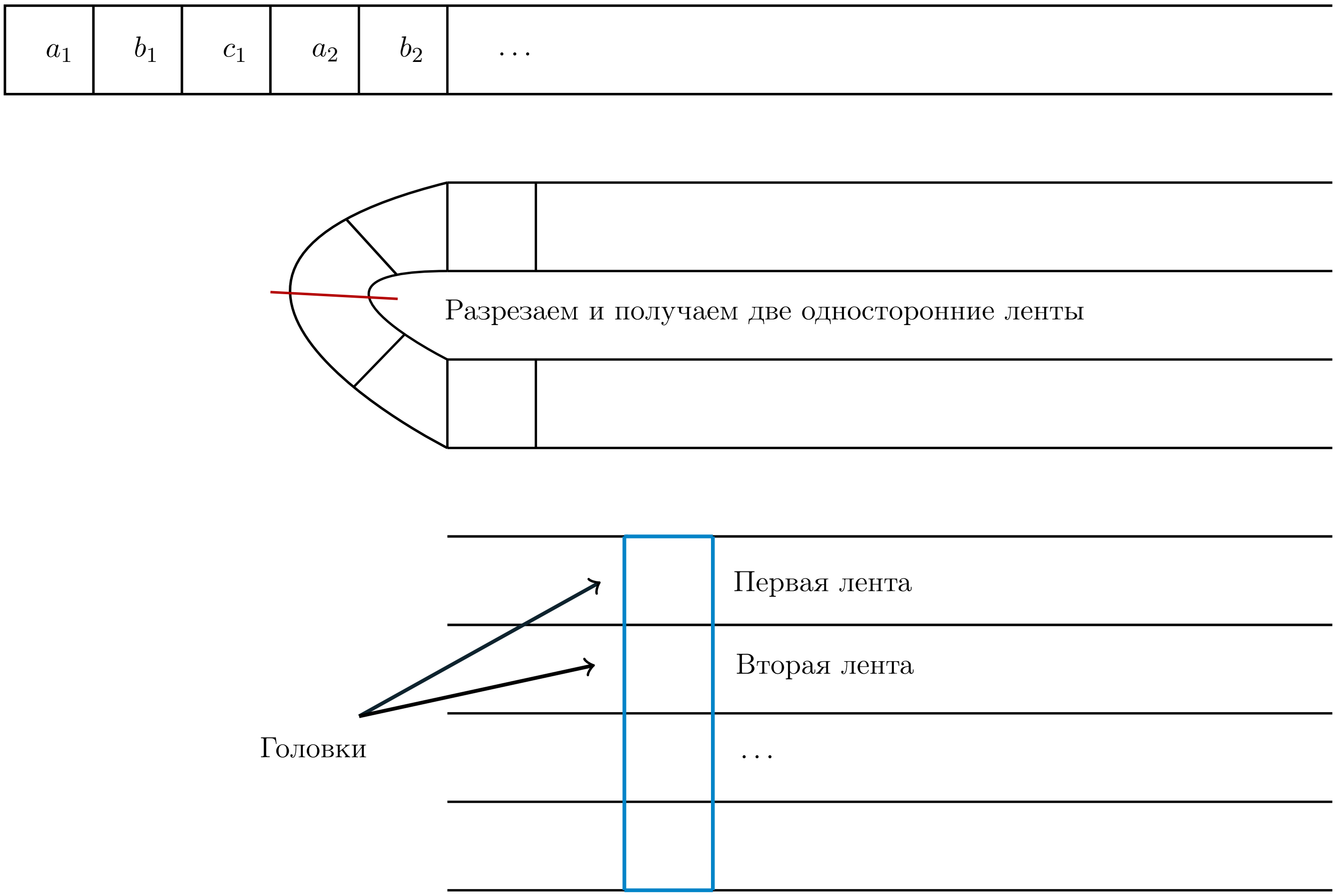

Рис. 1.1: Построение новой МТ

Пусть нужно подвинуть головку влево. Найдем слева первый не пустой блок L_i. Возьмем из него правую половину и разложим по пустым $L_{<i}$ так, чтобы порядок сохранился и каждый из $L_{<i}$ стал полупустым, а первый символ попал под головку.

Так получится сделать, так как всего перемещаемых символов 2^{i-1}, а в j-й блок будет помещено 2^{j-1} символов, поэтому всего в $L_{<i}$ поместится

$$1 + 2 + 4 + \ldots + 2^{i-2} = 2^{i-1} - 1.$$

И один символ под головку.

Чтобы инвариант сохранился нужно теперь исправить правую часть.

Так как первые $i-1$ левых блоков были пусты, первые $i-1$ правых блоков полны, а R_i пуст. Заполним половину в R_i символами из R_{i-1}. Теперь R_{i-1} пустой, а меньшие полные. Проделаем ту же операцию еще раз для $i-1$, потом для $i-2$ и так далее.

Кода мы дойдем до R_1, положим туда элемент из-под головки.

Итого, инвариант сохранился.

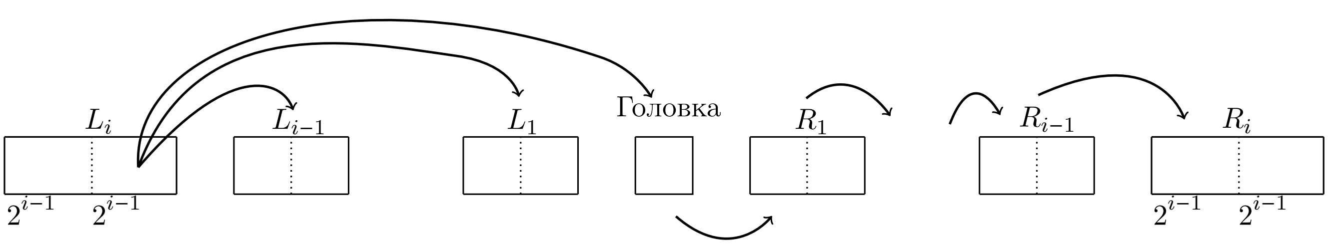

Рис. 1.2: Структура блоков

- Посчитаем количество операций. В алгоритме мы переносим различные отрезки из одного места в другое. Чтобы делать это за линию, сначала скопируем нужный участок на вторую ленту, а затем запишем с нее.

  Тогда при перераспределении происходит $c \cdot 2^i$ операций: каждый символ переносили константное число раз (на вторую ленту, со второй ленты) плюс линейное перемещение от L_i к R_i несколько раз.

  Докажем, что с i-м блоком происходят изменения не чаще 2^{i-1} шагов. Пусть L_i пустой хотя бы наполовину заполнен. Когда мы забрали половину из него, мы заполнили все $L_{<i}$ и $R_{<i}$ наполовину.

  Поэтому, чтобы изменить L_i еще раз, нужно сначала опустошить все $L_{<i}$. При перераспределении в левой части становится на один элемент меньше, а всего там 2^{i-1} заполненное место. Для того, чтобы все они ушли из левой половины, придется совершить 2^{i-1} сдвигов.

  Итого, для t шагов исходной машины будет

$$\sum_i c \cdot 2^i \cdot \frac{t}{2^{i-1}} = \mathcal{O}(t \log t).$$

■

Теорема 1.0.2 (Об универсальной МТ)**.** *Существует ДМТ U, выдающая на входе $(M,\ x)$ тот же результат, что дала бы машина M на входе x, за время $\mathcal{O}(t \log t)$, где t — время работы M на входе x.*

□ Используем прием из прошлой теоремы 1.0.1. ■

1.1 Классы сложности

1.1.1 Классы DTIME и P

Определение 5: Конструируемая по времени функция

Функция $t\colon \mathbb{N} \to \mathbb{N}$ называется **конструируемой по времени**, если

- $t(n) \geqslant n$;
- двоичную запись $t(|x|)$ можно найти по входу x на ДМТ за $t(|x|)$ шагов.

Определение 6: Класс DTIME

Язык L принадлежит классу $\mathsf{DTIME}[t(n)]$, если существует ДМТ M, принимающая L за время $\mathcal{O}(t(n))$, где t конструируема по времени.
Константа может зависеть от языка, но не от длины входа.

Определение 7: Класс P

Класс языков, распознаваемых за полиномиальное время на ДМТ —

$$\mathsf{P} = \bigcup_c \mathsf{DTIME}[n^c].$$

Будем обозначать задачи, заданные отношениями волной.

Определение 8

Массовая задача R **полиномиально ограничена**, если существует полином p, ограничивающий длину кратчайшего решения:

$$\forall x \left(\exists u\colon (x,u) \in R \Longrightarrow \exists w\colon \big((x,w) \in R \wedge |w| \leqslant p(|x|)\big)\right).$$

Массовая задача R **полиномиально проверяема**, если существует полином q, ограничивающий время проверки решения: для любой пары (x, w) можно проверить принадлежность $(x, w) \overset{?}{\in} R$ за время $q(|(x, w)|)$.

Определение 9: Класс $\widetilde{\mathsf{NP}}$

$\widetilde{\mathsf{NP}}$ — класс задач поиска, задаваемых полиномиально ограниченными полиномиально проверяемыми массовыми задачами.

Определение 10: Класс $\tilde{\mathsf{P}}$

$\tilde{\mathsf{P}}$ — класс задач поиска из $\widetilde{\mathsf{NP}}$, разрешимых за полиномиальное время.
То есть класс задач поиска, задаваемых отношениями R, что для всех $x \in \{0,1\}^*$ за полиномиальное время можно найти w, для которого $(x, w) \in R$.

$$\boxed{\text{Ключевой вопрос теории сложности } \tilde{\mathsf{P}} \overset{?}{=} \widetilde{\mathsf{NP}}}$$

Определение 11: Класс NP

NP — класс языков (задач распознавания), задаваемых полиномиально ограниченными полиномиально проверяемыми массовыми задачами:

$$\mathsf{NP} = \{L(R) \mid R \in \widetilde{\mathsf{NP}}\}.$$

Замечание. $L \in \mathsf{NP}$, если существует массовая п.о.п.п.[1] задача, такая, что

$$\forall x \in \{0,1\}^* \colon x \in L \iff \exists w \colon (x, w) \in R.$$

Определение 12: Класс P

P— класс языков (задач распознавания), распознаваемых за полиномиальное время.

$$\mathsf{P} = \{L(R) \mid R \in \mathsf{P}\}.$$

Замечание. Очевидно, $\mathsf{P} \subseteq \mathsf{NP}$.

$$\boxed{\text{Ключевой вопрос теории сложности } \mathsf{P} \overset{?}{=} \mathsf{NP}}$$

Лекция 2: †

12 nov

1.2 Недетерминированная машина Тьюринга

Определение 13: Недетерминированная машина Тьюринга

Недетерминированная машина Тьюринга — машина Тьюринга, допускающая более одной инструкции для данного состояния $q \in Q$ и $c_1, \dots c_k \in \Sigma$, то есть для состояния q и символа c функция δ будет многозначной.

Из такого определения получаем **дерево вычислений** вместо последовательности состояний ДМТ.

Мы говорим, что НМТ[2] **принимает** вход, если существует путь в дереве вычислений, заканчивающийся принимающим состоянием.

[1] полиномиально ограниченная полиномиально проверяемая
[2] недетерминированная машина Тьюринга

Утверждение. *В машины (ДМТ / НМТ) с заведомо ограниченным временем работы можно встроить* **будильник** *и считать время вычислений на входах одной длины всегда* **одинаковым**. *Для этого можем просто записать на дополнительную ленту $t(n)$ единиц и стирать по одной за ход.*

Определение 14: Эквивалентное определние НМТ

Недетерминированная машина Тьюринга — ДМТ, у которой есть дополнительный аргумент (конечная подсказка w на второй ленте).

□ Докажем эквивалентность. Представим дерево вычисления как бинарное дерево и пронумеруем ребра из каждой вершины 0 и 1. Теперь запишем нужную принимающую ветку на ленту-подсказку. По подсказке можем построить дерево, где будет нужный путь. ■

Определение 15

Еще одно определение класса NP — класс языков, принимаемых полиномиальными по времени НМТ.

1.3 Сведения и сводимости

Определение 16: Сведение по Карпу

Язык L_1 **сводится по Карпу** к языку L_2, если существует полиномиально вычислимая функция f такая, что

$$\forall x\colon x \in L_1 \iff f(x) \in L_2.$$

Определение 17: Сведение по Левину

Задача поиска (отношение) R_1 **сводится по Левину** к задаче R_2, если существуют функции f, g, h такие, что для всех x_1, y_1, y_2 верно

- $R_1(x_1, y_1) \implies R_2(f(x_1), g(x_1, y_1))$;
- $R_1(x_1, h(f(x_1), y_2)) \impliedby R_2(f(x_1), y_2)$;
- f, g, h полиномиально вычислимые.

Замечание. Первое условие нужно для того, чтобы образы каждого входа, имеющего решение первой задачи, имели решение и второй задачи.

Теорема 1.3.1. *Классы* $\mathsf{P}, \mathsf{NP}, \tilde{\mathsf{P}}, \widetilde{\mathsf{NP}}$ *замкнуты относительно этих сведений.*

□ Рассмотрим $R_2 \in \tilde{\mathsf{P}}$ и $R_1 \to R_2$. Тогда должно выполняться $R_1 \in \tilde{\mathsf{P}}$. Аналогично для $\widetilde{\mathsf{NP}}$. В обоих случаях R_2 задано п.о.п.п. Если говорим про $\tilde{\mathsf{P}}$, то еще есть алгоритм, работающий за полиномиально время.

Что можно узнать про R_1? Если есть решение для R_1, то функция g дает решение R_2, которое не на много длиннее. Еще есть h, которая позволяет построить из решения R_2 обратно построить решение R_1.

- Пусть есть некоторое решение y_1 для задачи R_1. Для него можно получить некоторое решение R_2 – $g(x_1, y_1)$.

  Так как R_2 полиномиально ограничено, для него есть полиномиальное решение поменьше. Поэтому, когда оно будет возвращено функцией h, исходное решение y_2 тоже окажется коротким.

- Полиномиальная проверяемость проверяется аналогично.

- Про алгоритм: если есть алгоритм для R_2 и x_1, сначала перегоняем $x_1 \to f(x_1)$, далее применяем алгоритм, получаем y_2, а далее, используя h, перегоняем обратно в R_1.

■

Определение 18: Оракульная МТ

Оракульная МТ — МТ с доступом к оракулу, который за один шаг дает ответ на некоторый вопрос.

Обозначение.
При переходе в состояние q_{in} происходит «фантастический переход» в состояние q_{out}, заменяющий содержимое некоторой ленты на ответ оракула.
M^B — оракульная машина M, которой дали оракул B.

Определение 19: Сведение по Тьюрингу

Язык или задача A **сводится по Тьюрингу** к B, если существует оракульная машина полиномиальная по времени $M^{\bullet}$ такая, что M^B решает A.
Например, если A — язык, $A = L(M^B)$.

Пример 1.3.1. Классы P и $\tilde{\mathsf{P}}$ замкнуты относительно сведений по Тьюрингу. А NP и $\widetilde{\mathsf{NP}}$ могут быть незамкнуты: если $A = \mathsf{UNSAT}$, $B = \mathsf{SAT}$, $M^O(x) = \overline{(x \in O)}$, и A сводится по Тьюрингу к B, но $B \in \mathsf{NP}$, а про A не известно.

1.3.1 Трудные и полные задачи

Определение 20: Трудный и полные задачи

Задача A называется **трудной** для класса C, если $\forall C \in \mathsf{C}\colon C \to A$.
Задача A называется **полной** для класса C, если она трудная и принадлежит C.

Теорема 1.3.2. *Если A — NP-трудная и $A \in \mathsf{P}$, то $\mathsf{P} = \mathsf{NP}$.*

Следствие 1. Если A — NP-полная, то

$$A \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

Задача об ограниченной остановке

Определение 21: BH

Определим задачу об ограниченной остановке $\widetilde{\mathsf{BH}}(\langle M, x, 1^t\rangle, w)$[a] так: дана НМТ M и вход x, требуется найти такую подсказку w, чтобы M распознавала x не более чем за t шагов.
Соответствующая задача распознавания — ответить, существует ли такая подсказка.

[a] здесь 1^t — служебные t единиц

Теорема 1.3.3. *Задача об ограниченной остановке $\widetilde{\mathsf{NP}}$-полная, а соответствующий язык NP-полный.*

□

- Принадлежность $\widetilde{\mathsf{NP}}$ и NP следует из существования универсальной ДМТ, которая за $\mathcal{O}(t \log t)$ промоделирует вычисление ДМТ, описание которой дано ей на вход.
- Проверим, что язык NP-трудный. Пусть язык L принадлежит NP, что равносильно существованию для соответствующего отношения R машины Тьюринга $M^*(x, w)$, которая работает за $\mathsf{poly}(|x|)$.

  Сведем L к задаче BH. Рассмотрим тройку $\langle M^*, x, 1^{\mathsf{poly}(|x|)}\rangle$. Пусть функция $f(x)$ будет равна этой тройке.

  Если и только если существует подсказка w для тройки $f(x)$ принадлежит BH, а наш язык L и определяется машиной M^*.

- Аналогично для задач поиска.

■

1.3.2 Булевы схемы

Определение 22: Булева схема

Булева схема — ориентированный граф без циклов,в вершинах которого записаны бинарные, унарные или нульарные операции над битами ($\wedge, \vee, \oplus$), при этом есть специальные вершины-входы и вершины-выходы.

Определение 23: $\mathsf{CIRCUIT\_SAT}$

$$\mathsf{CIRC\widetilde{UIT}\_SAT} = \{(C, w) \mid C \text{ — булева схема}, C(w) = 1\}.$$

Очевидно, что $\mathsf{CIRCUIT\_SAT} \in \mathsf{NP}$. Чтобы доказать NP-трудность, сведем $\mathsf{BH} \to \mathsf{CIRCUIT\_SAT}$: будем рисовать конфигурацию МТ на схемах.

- Пусть каждый этаж системы — конфигурация ДМТ. Всего этажей будет столько же, сколько шагов в МТ, то есть t. Если в последнем этаже q_{acc}, то результат 1.
- Пересчет конфигураций: меняются только гейты рядом с положением головки. Выделим подсхему, которая должна по состоянию и элементу рядом с головкой получить новое состояние, положение головки и поменять элемент, то есть построить новый уровень с измененными элементами.

  Для хранения головки будем после каждого символа с ленты хранить d_i равное единице, если головка на символе c_i перед ним.

  Если $d_i = 0$, то новый $c_i^{'} = c_i$, иначе нужно заменить c_i на $c_i^{'}$ из программы МТ:

$$(q, c_{i-1}, c_i, c_{i+1}, d_{i-1}, d_i, d_{i+1}) \mapsto (q^{'}, c_i^{'}, d_i^{'}).$$

- Так как входная строка x нам дана, запаяем ее в схему. Тогда входом схемы останется подсказка w для НМТ, а выходом — попадание в q_{acc}.

Таким образом, мы по M, x, t построили схему $C(w)$.