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

Тамарин Вячеслав

9 ноября 2020 г.

Оглавление

Глава 1

Введение в теорию сложности вычислений

Лекция 1: †

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)**.**

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

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

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

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

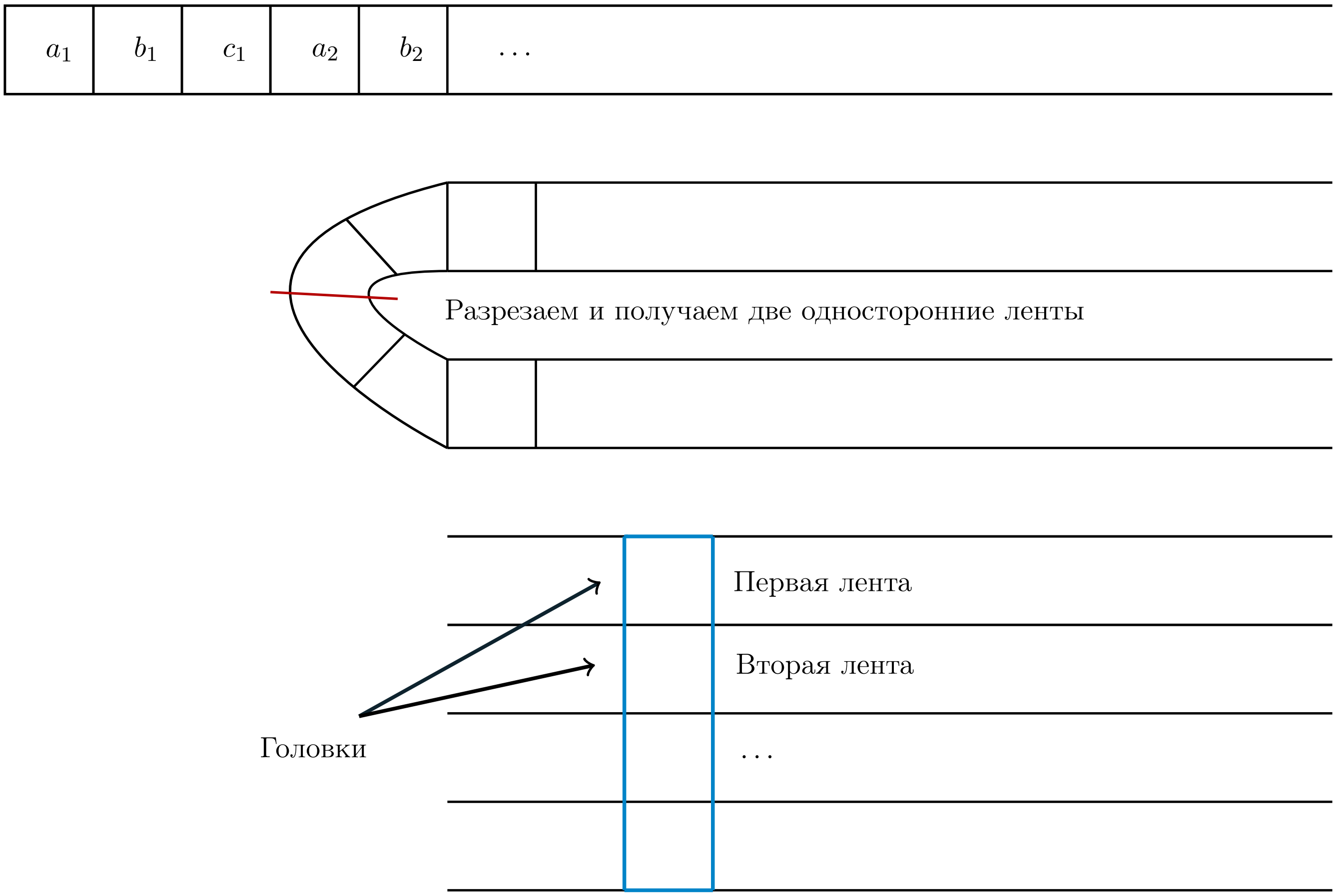

Рис. 1.1: Построение новой МТ

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.

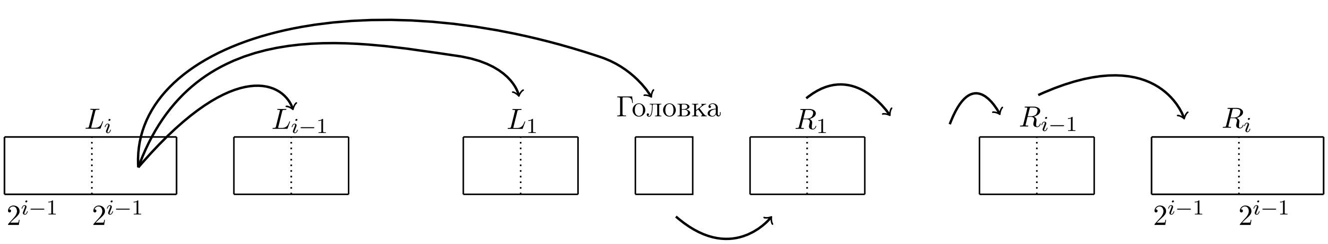

Рис. 1.2: Структура блоков

- 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 $\mathtt{DTIME}[t(n)]$, 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 —

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

Будем обозначать задачи, заданные отношениями волной.

Определение 8

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

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

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 $\widetilde{\mathtt{NP}}$

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

Определение 10: Класс $\widetilde{\mathtt{P}}$

$\widetilde{\mathtt{P}}$ — класс задач поиска из $\widetilde{\mathtt{NP}}$, 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 $(x, w) \in R$.

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

Определение 11: Класс NP

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

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

Замечание. $L \in \mathtt{NP}$, 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.

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

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

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

[1] полиномиально ограниченная полиномиально проверяемая