Задание 4

Сложность вычислений: классы P, NP и co-NP

Литература:

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.
   Алгоритмы. Построение и анализ.
   2-е изд. М.: Вильямс, 2005.

2. Sipser M.
   Introduction to the theory of computation

3. Arora S., Barak B.
   Computational Complexity: A Modern Approach

4. Hopcroft J., Ullman J.
   Introduction to Automata Theory, Languages, and Computation.
   1-st edition, 1979.

На всякий случай прежде всего обращу здесь внимание на то, что ближайший семинар пройдёт в понедельник 13 марта!

Перед основным вспомогательным материалом, разделы которого вынесены в заглавие названия, я хочу напомнить базовые вещи из теории вычислений, раздел которой вы изучали в рамках курса «теория формальных систем и алгоритмов». Этому посвящены первые два раздела данного задания.

Обращаю внимание, что в этих разделах приведены базовые определения, на обобщении которых мы определяем класс NP и полиномиальную m-сводимость. Задачи из этих разделов помечены символом † – это означает, что они являются дополнительными и их можно не сдавать, однако их стоит прорешать, если на это хватает времени и сил.

1 Машины Тьюринга и формальные языки

Под формальным языком L мы понимаем некоторое подмножество множества всех слов над алфавитом Σ. В этой теме слова «язык» и «задача» являются синонимами, поскольку речь идёт только о задачах распознавания, то есть задач проверки принадлежности слова языку. То

есть задача перемножить два числа не является задачей в наших терминах в рамках этого раздела.

Каждому языку мы ставим в соответствие машину Тьюринга, которая его распознаёт.

Определение 1. *Детерминированная Машина Тьюринга* M имеет l лент, k головок, множество состояний Q, множество принимающих состояний $Acc \subset Q$, входной алфавит Σ, функцию перехода $\delta : \Sigma^k \times Q \to \{0, +1, -1\}^k \times \Sigma^k \times Q$ – за такт работы машина меняет состояние, символы под каждой головкой и двигает каждую головку влево, вправо или оставляет её неподвижной. На машину могут быть наложены ограничения, например, полиномиальное время работы или полиномиальная память по входу. Среди всех лент выделена входная лента, на которой в начале работы машины написано входное слово x. Если машина переходит в принимающее состояние, то она останавливается и принимает входное слово.

Будем говорить, что машина Тьюринга M вычисляет функцию $M(x)$, которая равна 1, в случае если машина M на входе x останавливается в принимающем состоянии, если машина M на входе x останавливается в не принимающем состоянии, то $M(x) = 0$, а если машина M не останавливается на входе x, то функция $M(x)$ не определена.

$$M(x) = \begin{cases} 1, & M \text{ остановилась в принимающем состоянии на } x; \\ 0, & M \text{ остановилась в не принимающем состоянии на } x; \\ \uparrow, & M \text{ не остановилась, функция не определена на } x. \end{cases}$$

Машина M *принимает* язык L, если она принимает каждое слово из языка L, если машина M принимает язык L и к тому же все слова, принимаемые машиной M, лежат в языке L, то будем говорить, что машина M *распознаёт* L. Язык, распознаваемый машиной M будем обозначать $L(M)$.

Замечание 1. *Определение языка, распознаваемого машиной Тьюринга, при беглом взгляде кажется естественным, но на самом деле оно весьма коварно. На слова не из L ограничений не накладывается, поэтому на них машина M может и не останавливаться вовсе.*

Языки, распознаваемые машинами Тьюринга, называются *(рекурсивно-)перечислимыми* языками или *распознаваемыми* языками.

Будем говорить, что машина Тьюринга M разрешает язык L, если она останавливается на всех входах и распознаёт язык L.

Языки разрешимые машинами Тьюринга образуют класс *рекурсивных* языков, также называемыми *разрешимыми*.

Если не оговорено противного, мы будем считать, что машина Тьюринга M имеет одну ленту и одну головку.

Поскольку машины Тьюринга имеют конечное описание, то их описание можно закодировать. Будем обозначать M_α машину Тьюринга, описание которой закодирована строкой α. Мы можем просто занумеровать все возможные описания машин Тьюринга и считать, что α – натуральное число. Если же это неудобно, мы можем считать, что α есть просто строка, содержащая описание машины Тьюринга, и если описание некорректно, то будем считать, что M_α – машина Тьюринга, распознающая пустой язык.

Из возможности кодирования описаний машин Тьюринга следует, что существует машина Тьюринга, которая получает на вход описание α и вход x и проделывает работу машины M_α на входе x. Такую машину Тьюринга будем называть *универсальной* и обозначать UM.

2 Разрешимые и неразрешимые задачи

Прежде чем говорить о сложности задачи надо доказать, что она разрешима. Это далеко не всегда так даже в очень естественных случаях.

Пример 1. Пусть язык L лежит в классе CFL. Проверка условия $L \stackrel{?}{=} \Sigma^*$ – неразрешимая задача.

Одна из самых распространённых неразрешимых задач – проблема останова. Под проблемой останова понимается язык HALT состоящий из описаний всех машин Тьюринга, останавливающихся на пустом входе.

Пример 2.

$$\text{HALT} = \{\alpha \mid M_\alpha(\varepsilon) = 1\}$$

Упражнение 1. Докажите что язык HALT является неразрешимым. Если доказать самостоятельно не получается, обратитесь к литературе.

Язык L называется *перечислимым*, если существует такая МТ M, которая выводит все слова из языка. МТ M может вообще говоря никогда не остановится, но если слово w лежит в L, то машина M должна вывести w через некоторое, быть может очень большое, число тактов.

Упражнение 2. Докажите, что данное здесь определение перечислимого языка согласовано с данным выше определением.

Упражнение 3. Докажите, что язык HALT является перечислимым.

Упражнение 4. Докажите, что язык L является разрешимым тогда и только тогда, когда языки L и $\bar{L}$ перечислимы.

Задача 1†. Докажите, что язык, состоящий из описаний МТ, не останавливающихся на некотором входе через более чем 2017 шагов неразрешим.

Задача 2†. Является ли разрешимым язык, состоящий из описаний МТ, которые делают меньше 2015 шагов при обработке каждого входного слова?

Для того, чтобы доказать, что задача неразрешима, достаточно свести к ней неразрешимую задачу (например, HALT). Напомним определение сводимости.

Определение 2. Пусть для языка $A \subseteq \Sigma_1^*$ существует такая всюду-определённая вычислимая функция $f : \Sigma_1^* \to \Sigma_2^*$, что слово x принадлежит A тогда и только тогда, когда слово $f(x)$ принадлежит языку B. Будем говорить, что язык A сводится к языку B m-сводимостью и обозначать это как $A \leqslant_m B$. В формулах это выглядит как

$$x \in A \iff f(x) \in B.$$

Задача 3. Верно ли, что разрешимый язык нельзя свести к неразрешимому?

3 Класс P

Нас будет интересовать классификация языков. Под классом языков понимается некоторое множество языков. Как правило, классы языков задают ограничениями на модели вычислений, которые распознают языки из данного класса.

Если детерминированная машина Тьюринга M распознаёт L, причём для каждого слова x из L она делает не более чем $O(n^c)$ тактов, то язык L лежит в классе $\mathsf{DTIME}(O(n^c))$. Такую машину Тьюринга мы будем называть детерминированной полиномиальной или просто полиномиальной.

Класс P состоит из объединения всех языков, лежащих в $\mathsf{DTIME}(O(n^c))$, то есть

$$\mathsf{P} = \bigcup_{c \geqslant 0} \mathsf{DTIME}(O(n^c))$$

Задача 4†. Покажите, что в определении класса P неважно сколько лент и головок у машины Тьюринга M. То есть, если язык L распознаётся за полиномиальное время машиной Тьюринга M с k лентами и l головками, то он распознаётся и некоторой машиной M' с одной лентой и одной головкой.

Оказывается, что разумные модели вычислений полиномиально-эквивалентны. Поэтому, когда нужно проверить, что язык принадлежит классу P, достаточно найти алгоритм на вашем любимом языке программирования, который разрешает язык за полиномиальное время.

4 Недетерминированные машины Тьюринга

Определение недетерминированной машины Тьюринга получается из определения детерминированной подобно тому как из определения ДКА получается определение НКА: достаточно заменить функцию переходов на отношение переходов и добавить квантор существования в условие приёма слова. То есть, недетеринированная машина Тьюринга M принимает слово x, если существует последовательность переходов, которая приводит машину M на слове x в принимающее состояние.

Упражнение 5. Покажите, что детереминированные и недетерминированные машины Тьюринга распознают один и тот же класс языков.

Если недетерминированная машина Тьюринга M распознаёт язык L, причём для каждого слова x из L она делает не более чем $O(n^c)$ тактов,

то язык L лежит в классе $\mathsf{NTIME}(O(n^c))$. Такую машину Тьюринга мы будем называть недетерминированной полиномиальной.

Таким образом, аналогично определению класса P, получим определение класса NP:

$$\mathsf{NP} = \bigcup_{c \geqslant 0} \mathsf{NTIME}(O(n^c))$$

5 Класс NP

Приведём более наглядное эквивалентное определение класса NP. Под записью $M(x, y)$ мы понимаем, что на вход машине M подали строки x и y, записанные через разделитель, например $M(x, y) = M(x\#y)$.

Напомним, что в случае когда мы рассматриваем соответствующую машине Тьюринга M вычислимую функцию $M(\cdot)$ одного аргумента, запись $M(x) = 1$ означает, что машина M принимает слово x или что то же самое, что слово x принадлежит языку, распознаваемому машиной M: $x \in L(M)$.

Определение 3. Язык L лежит в классе NP, если существуют полином $p(n) : \mathbb{N} \to \mathbb{N}$ и (детерминированная) полиномиальная МТ M, такие что

$$x \in L \Leftrightarrow \exists y \in \Sigma^{p(|x|)}\ M(x, y) = 1.$$

Слово y мы будем называть *сертификатом* для слова x.

Упражнение 6. Покажите, что два определения класса NP эквивалентны.

Задача 5. Пусть мы не накладываем полиномиального ограничения на сертификат, но при этом машина M является полиномиальной по длине входного слова x.

То есть, для языка L есть машина $M(x, y)$ полиномиальная по длине входа $|x|$ и

$$x \in L \Leftrightarrow \exists y \in \Sigma^*\ M(x, y) = 1.$$

Верно ли, что $L \in \mathsf{NP}$? Если да, то как найти полиномиальный по x сертификат y?

6 Сводимости

Довольно часто в сложности вычислений мы сталкиваемся со следующей ситуацией: оказывается, что мы умеем решать задачу A, если мы уже умеем решать задачу B. Или наоборот, задача B является сложной (или даже неразрешимой), если задача A является сложной (неразрешимой).

Для описания таких отношений между языками мы пользуемся сводимостью. В этом задании мы будем говорить об m-сводимости или сводимости по Карпу.

Определение 4. Пусть для языка $A \subseteq \Sigma_1^*$ существует такая всюду-определённая полиномиально-вычислимая функция $f : \Sigma_1^* \to \Sigma_2^*$, что слово x принадлежит A тогда и только тогда, когда слово $f(x)$ принадлежит языку B. Будем говорить, что язык A сводится к языку B полиномиальной m-сводимостью (сводимостью по Карпу) и обозначать это как $A \leqslant_m^p B$.

Для краткости, мы будем говорить вместо «сводится m-сводимостью» и «сводится полиномиальной m-сводимостью» «сводится» и «полиномиально сводится».

Упражнение 7. Покажите, что если задача HALT сводится к задаче A, то задача A является неразрешимой.

Упражнение 8. Покажите, что если задача A сводится полиномиально к задаче B и задача B лежит в классе P, то и задача A лежит в классе P.

Задача A является NP-полной, если задача A лежит в NP и любая задача $B \in \mathsf{NP}$ полиномиально сводится к A Класс NP-полных задач мы будем обозначать NP-с. Формально

$$L \in \mathsf{NP}\text{-с} \Leftrightarrow L \in \mathsf{NP},\ \forall A \in \mathsf{NP} : A \leqslant_m^p L.$$

Факт существования $\mathsf{NP}-$полных задач установили независимо друг от друга Левин и Кук. В ближайшее время изучение $\mathsf{NP}-$полных задач будет нашим основным полем деятельности.

Определение NP-полного языка состоит из двух частей. Первую из них, о том, что любой язык из класса NP сводится к языку A выделяют в отдельное определение – такие языки называются NP-трудными. Таким

образом, язык A является NP-полным, если он является NP-трудным и принадлежит классу NP.

Помимо классов NP и NP-с нас также будет интересовать класс co-NP, состоящий из языков, являющихся дополнением к языкам из NP. То есть, если язык L лежит в классе NP, то язык $\bar{L}$ лежит в классе co-NP

7 NP-полные задачи

Приведём пример NP-полной задачи.

Пример 3. Язык SAT состоит из всех выполнимых булевых формул ϕ, заданных в конъюнктивной нормальной форме.

$$\text{SAT} = \{\phi \,|\, \exists y_1, \ldots, y_n : \phi(y_1, \ldots, y_n) = 1\}$$

Теорема (Кук, Левин)**.** *Язык* SAT *является* NP*-полным.*

Упражнение 9. Изучить доказательство теоремы Кука-Левина. Ознакомиться с ним я рекомендую в книжке Сипсера.

На семинаре мы говорили о языке 3-SAT, который состоит из выполнимых булевых формул, каждый дизъюнкт которых содержит ровно три переменных. Пример такой формулы:

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_4 \vee x_5 \vee x_1).$$

Задача 6. Показать, что из теоремы Кука-Левина следует, что 3-SAT $\in$ NP-с

Задача 7. Показать, что 2-SAT $\in$ P.

Также нам интересен класс co-NP, состоящий из языков, дополнение которых лежит в NP.

Определим язык UNSAT как язык состоящий из невыполнимых булевых формул, заданных КНФ. То есть

$$\text{UNSAT} = \{\phi \,|\, \forall y_1, \ldots, y_n : \phi(y_1, \ldots, y_n) = 0\}$$

Упражнение 10. Показать, что язык UNSAT лежит в классе co-NP.

Полнота языка в классе относительно сводимостей осмыслена не только в случае NP-полных языков. В учебных целях мы будем пользоваться полиномиальной m-сводимостью также в классе P и co-NP. Разумеется, полнота относительно полиномиальной сводимости в классе P выглядит нелепо – в реальной жизни класс P исследуют относительно более осмысленной сводимости – сводимости на логарифмической памяти. Относительно этой сводимости, например, полна задача о проверки пустоты языка, заданного КС-грамматикой. То есть, любая задача из класса P сводится к задаче о пустоте языка, заданного КС-грамматикой на логарифмической памяти. Подробнее об этом можно почитать, например, в книжке Хопкрофта-Ульмана «Introduction to Automata Theory, Languages, and Computation» 1979-го года – второе здание этой книги, выпущенное совместно с Мотвани многие из вас изучали в курсе ТРЯП, но второе издание сильно упрощено и сжато.

Задача 8. Верно ли, что любой язык из класса P является полным относительно полиномиальной m-сводимости?

Задача 9†. Докажите, что язык UNSAT является полным в классе co-NP относительно полиномиальной m-сводимости.

8 О сложности вычислений

Соотношение между классами P, NP и NP-с co-NP представляет собой центральный вопрос сложности вычислений. Задача $\mathsf{P} \stackrel{?}{\neq} \mathsf{NP}$ является основной в сложности вычислений и, по-видимому, безнадёжно трудной. Эта задача стала своего рода наследником теоремы Ферма, в том плане, что регулярно (хотя и не столь часто как в случае теоремы Ферма), находятся люди, которые «доказывают», что $\mathsf{P} = \mathsf{NP}$ или обратное. Интерес к проблеме подогревается институтом Клэя, обещавшим за решение этой задачи \$1 000 000. Тем не менее, недавно была предпринята первая серьёзная попытка действительно доказать, что $\mathsf{P} \neq \mathsf{NP}$. Так что возможно мы всё же доживём до решения этого вопроса.

Сообщество верит, что $\mathsf{P} \neq \mathsf{NP}$ и довольно многие результаты доказываются по модулю этой гипотезы. Более того, сложность этого вопроса используется на практике, в частности в криптографии. В этом курсе мы непосредственно с этим столкнёмся при изучении RSA-алгоритма шифрования. Тем не менее, то что задача лежит в NP и даже в NP-с ещё

вовсе не означает что её частный случай, *экземпляр* трудно решить. Это означает лишь, что задача трудна только для некоторых входов.

9 Домашнее задание

На всякий случай прежде всего обращу здесь внимание на то, что ближайший семинар пройдёт в понедельник 13 марта!

Поскольку у нас пропал семинар 23-го февраля, блок из задач получился большим. Я разбил его на две части — вторая скорее нужна для подготовки к контрольной. В первой части много базовых задач: блок задач 20-24 посвящён различным сводимостям, которые стоит освоить, но из-за нехватки времени их можно решать и после семинара в четверг на следующей неделе. Но не стоит оставлять их все на потом.

Часть I:

Сдать до 15-го марта. Часть задач из блока 20-24 можно прислать до 22 марта.

Базовые задачи: 3, 6, 7 из этого текста; КДЗ 2017: 18, 20-24.

Дополнительные задачи: 5 из этого текста; КДЗ 2017: 15, 16, 17.

Часть II:

Бонусная часть, которую стоит прорешать перед контрольной, и можно прислать по-желанию до 22 марта.

Базовые задачи: КДЗ 2017: 28-29.

Дополнительные задачи: КДЗ 2017: 25.