Алгоритмы и анализ сложности, 3 семестр ПИ, Ответы на билеты

Собрано 21 декабря 2022 г. в 20:14

Содержание

Раздел #1: ? TODO: проблемы и доказательства

1.1. Машины Тьюринга и тезис Чёрча

Определение 1 (Машина Тьюринга). Да что за «Машина Тьюринга»?

- Абстрактная вычислительная машина.
- Формализация понятия алгоритма.
- Расширение конечного автомата.
- Лента (бесконечная).
- Головка записи-чтения (управляющее устройство), способная находиться в одном из множества состояний, которое конечно и точно задано.
- Это управляющее может перемещаться влево и вправо по ленте, читать и записывать в ячейки символы некоторого конечного алфавита.
- Существует ϵ-символ, который заполняет все пустые клетки ленты.
- Управляющее устройство работает согласно правилам перехода, которые представляют алгоритм, реализуемый данной МТ. Каждое правило перехода предписывает машине, в зависимости от текущего состояния и наблюдаемого в текущей ячейке символа, записать в эту клетку новый символ, перейти в новое состояние и переместиться на одну клетку влево или вправо (существует некий «синтаксический сахар» — остаться на месте). Некоторые состояния могут быть помечены как терминальные, и переход в любое из них означает конец работы, остановку алгоритма.

Это все очень интересно, но как насчет формализма?
Формально: $M = \{Q, G, \epsilon, \Sigma, \delta, q_0, F\}$ — запомните этот набор из семи элементов!

- Q — конечное, не являющееся пустым, множество состояний.
- G — конечный, не являющийся пустым, набор символов ленточного алфавита.
- ϵ — единственный пустой символ.
- $\Sigma = G \backslash \{\epsilon\}$ — набор входных символов.
- $\delta : (Q \backslash F) \times G \nrightarrow Q \times G \times \{L, R\}$ — частичная функция, называемая функцией перехода, где L — сдвиг влево, а R — сдвиг вправо. Если δ не определена для текущего состояния и символа ленты, то машина останавливается.
- q_0 — это начальное состояние.
- $F \subset Q$ — набор конечных состояний.

Замечание. А что если мы хотим такое состояние, которое будет являться term при одном символе на ленте и nonterm при другом символе?
Если немного подумать, то здесь все в порядке, поскольку мы можем сделать это состояние nonterm, перейти в другое term состояние, а при определенном символе сдвинуться, например, вправо, записав на ленту такой же символ, что был на ней.

Замечание. Хоть любой конечный алфавит и не ограничен одними цифрами 0 и 1, очевидно, что мы всегда можем его представить в виде двоичных чисел, введя на нем порядок.

Определение 2 (Частичная функция). частичная функция f из множества X в множество Y — это функция из подмножества S из X (возможно, самого X) в Y (обозначение: $\nrightarrow$).

Определение 3 (Детерминированная и недетерминированная машины Тьюринга). Машина Тьюринга называется детерминированнй, если каждой паре состояни и ленточного символа соответствует не более одного правила. В ином случае, машина является недетерминированной.

Определение 4 (Тезис Черча/Тьюринга/Черча-Тьюринга). Есть ли отличие?
На самом деле, все они говорят об одном, просто Черч в свое время придумал λ-исчисления, Тьюринг придумал Машину Тьюринга, а позже было показано, что эти формализмы эквивалентны.
Сам тезис сформулируем следующим образом: Класс алгоритмически вычислимых частичных функций совпадает с классом всех функций, вычислимых на машине Тьюринга.

Замечание. Стоит понимать, что это именно тезис, а не теорема, ведь понятие «алгоритмически вычислимая частичная функция» неформально.

Определение 5 (Вычислимая функция). Вычислимые функции — это множество функций вида, $f\colon N \to N$, которые могут быть реализованы на машине Тьюринга.
В качестве множества N обычно рассматривается множество B^* — множество слов в двоичном алфавите $B = \{0, 1\}$, с оговоркой, что результатом вычисления может быть не только слово, но и специальное значение «неопределённость», соответствующее случаю, когда алгоритм «зависает». Таким образом, можно дать следующее определение N:
$N = B^* \cup \{\text{undef}\}$, где $B = \{0, 1\}$, а undef — специальный элемент, означающий неопределённость.
Роль множества N может играть множество натуральных чисел, к которому добавлен элемент undef, и тогда вычислимые функции — это некоторое подмножество натуральнозначных функций натурального аргумента. Удобно считать, что в качестве N могут выступать различные счётные множества — множество натуральных чисел, множество рациональных чисел, множество слов в каком-либо конечном алфавите и др.

1.2. Классы RE, R и $co-RE$, доказательство $R = RE \cap co-RE$

Определение 6 (Классы RE и $co-RE$). Класс RE (recursively enumerable) — класс decision problems (проблемы принятия решения), для которых ответ «да» может быть проверен машиной Тьюринга за конечное время.

- Если на проблему ответ «да», то существует некоторая процедура, которая требует конечного времени для определения этого.
- Ложь здесь отсутствует.
- Если на проблему ответ «нет», то машина Тьюринга может остановиться, а может и не остановиться.

Класс $co-RE$ является дополнением к классу RE: ответ «нет» можно получить за конечное время абсолютно всегда, получение противоположного ответа может занять вечность.

Определение 7 (Формальный язык). Формальный язык (или просто язык) — это множество конечных слов над конечным алфавитом.

Определение 8 (Класс R). R — класс decision problems, решаемых на MT (набор всех рекурсивных языков).

Определение 9 (Рекурсивный язык). Формальный язык является рекурсивным, если существует полная машина Тьюринга (машина Тьюринга, которая останавливается для каждого заданного ввода), которая, когда на вход подается конечная последовательность символов, принимает ее, если она принадлежит языку, и отвергает ее в противном случае.

Теорема 1. $R = RE \cap co-RE$.

Доказательство. Обозначим за X класс decision problems, содержащихся в классе RE, ответы «да» и «нет» на которые можно получить за конечное время. Очевидно, что $X \subset co-RE$ по определению, а также никакая другая задача, содержащаяся в RE не содержится в $co-RE$. То есть, $X = RE \cap co-RE$. Однако $X = R$, так как это в точности класс decision problems, решаемых на машине Тьюринга, то есть: $R = X = RE \cap co-Re$. □

1.3. Проблемы Acceptence, Halting, Emptiness

Определение 10 (Halting Problem). Проблема останова машины Тьюринга...

Раздел #2: ? TODO: полностью

Раздел #3: ?

3.1. Построение универсальной машины Тьюринга

Определение 11 (Универсальная машиина Тьюринга). Универсальная машина Тьюринга — такая машина, которая может заменить собой любую машину Тьюринга. Получив на вход программу и входные данные, она вычисляет ответ, который вычислила бы по входным данным машина Тьюринга, чья программа была дана на вход.

Определение 12 (Построение УМТ).

Раздел #4: 4?

Раздел #5: 5?

5.1. Определение классов сложности P, NP, $co-NP$

Определение 13 (Класс сложности P). Классом P называются все проблемы принятия решений, которые могут быть решены детерминированной машиной Тьюринга с использованием полиномиального количества времени вычислений или полиномиального времени.

Пример (Задача из класса сложности P). Пусть у нас есть массив натуральных чисел, состоящий из n элементов. Вопрос: содержится ли число 5 в этом массиве?
Решение: Простой перебор элементов массива (если массив упорядочен, то можно воспользоваться бинарным поиском).

Определение 14 (Класс сложности NP). Классом NP называют множество задач принятия решения, решение с ответом «да» каждой из которых можно проверить на детерминированной машине Тьюринга за время, не превосходящее какой-либо полином.

Пример (Задача из класса сложности NP). Дано число n. Вопрос: раскладывается ли данное число на три простых?
Решение: заметим, что нельзя точно утверждать, содержится ли данная задача в классе P, поскольку единственное решение, которое пока что придумано — это простой перебор, занимающий более чем полиномиальное время работы. Однако если у нас на руках существует решение данной задачи с ответом «да», то мы легко можем проверить данное решение, перемножив три числа, содержащихся в решении. Если перемножение дает верный ответ, то решение верно, и наоборот. Доказательство того, что проверка занимает не более чем экспоненциальное время оставим в качестве упражнения читателям.

Замечание. На самом деле, решение прошлой задачи с ответом «нет» тоже можно проверить за экспоненциальное время. Само решение будет состоять в том, чтобы показать, что число раскладывается не на три простых, а на какое-либо другое количество. Проверка решения аналогична: перемножить и убедиться, либо же опровергнуть корректность решения. Это значит, что описанная нами задача также принадлежит и классу $co-NP$, о котором сказано ниже.

Замечание. Очевидно: $P \subset NP$ (если мы можем решить задачу за полиномиальное время, то мы можем проверить решение задачи, просто решив ее).

Определение 15 (Класс сложности $co-NP$). Классом co-NP называют множество задач принятия решения, дополнение к которому лежит в классе NP. Это означает, что каждая задача, решение которой с ответом «нет» можно проверить на детерминированной машине

Тьюринга за время, не превосходящее какой-либо полином, лежит в классе $co\text{-}NP$.

5.2. Определение полиномиальной сводимости и класса NP-полных языков

Определение 16 (Полиномиальная сводимость). Любой язык L_1 называется сводимым по Карпу к языку L_2, если существует функция $F: \Sigma^* \mapsto \Sigma^*$, вычисляемая за полиномиальное время, где F(x) принадлежит L_2 в том случае, если x принадлежит L_1.

Определение 17 (Класс сложности $NP\text{-}complete$). Класс $NP\text{-}complete$ — множество задач принятия решения из класса NP, к каждой из которых можно свести **любую** другую задачу из этого класса за полиномиальное время.

Замечание. Найдя алгоритм для решения любой задачи из класса $NP\text{-}complete$ за полиномиальное время, возможно решать каждую задачу NP за полиномиальное время, а это решает проблему $P = NP$.

5.3. Взаимосвязи этих классов

Замечание. Взаимосвязи:
$P \subset NP$;
$P \subset co - NP$;
$NP \cap co - NP \neq \varnothing$.

5.4. Доказательство того, что если NP-полный язык лежит в $co - NP$, то $NP = co - NP$

Теорема 2. Если NP-полный язык лежит в $co\text{-}NP$, то $NP = co\text{-}NP$.

Доказательство. Пусть L — это язык из класса $co - NP$. Заметим, что дополнение к этому языку лежит в классе NP. Сведем это дополнение к NP-полному языку (за полиномиальное время), который лежит в $co - NP$. Получается, что дополнение к языку L лежит в классе $co - NP$. Данное рассуждение мы можем проделать для любого $co - NP$ языка. Получается, что дополнения к каждой задаче лежат в $co - NP$, но эти дополнения по определению лежат в NP, а значит $NP = co - NP$. □

Раздел #6: x

Раздел #7: x

7.1. Теорема Кука-Левина об NP-полноте задачи CNF-SAT

Определение 18 (Булева формула). Булева формула — формула логики высказываний.

Замечание. Формула называется тождественно истинной (ложной), если она истинна (ложна) при любых значениях переменных. Две булевы формулы называются эквивалентными тогда и только тогда, когда они истинны на одном и том же подмножестве множества значений аргументов.

Определение 19 (Задача SAT). Задача SAT — это задача выполнимости булевых формул. Экземпляром задачи является булева формула, состоящая только из имён переменных, скобок и операций $\wedge$ (И), $\vee$ (ИЛИ) и $\neg$ (НЕ). Задача заключается в следующем: можно ли назначить всем переменным, встречающимся в формуле, значения ложь и истина так, чтобы формула стала истинной.

Определение 20 (Задача CNF-SAT). Определение почти аналогично, однако на булеву формулу накладывается ограничение: она должна быть записана в конъюнктивной нормальной форме (должна иметь вид конъюнкции дизъюнкций литералов).

Замечание. Любая булева формула может быть приведена к КНФ.

Пример. Формулы в КНФ:
$\neg A \wedge (B \vee C)$;
$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$;
$A \wedge B$.
Формулы не в КНФ:
$\neg(B \vee C)$;
$(A \wedge B) \vee C$;
$A \wedge (B \vee (D \wedge E))$.

Определение 21 (Задача 3-SAT). Задача 3-SAT — частный случай задачи SAT, где булева формула записана в 3-конъюнктивной нормальной форме.

Определение 22 (k-КНФ). k-конъюнктивной нормальной формой называют конъюнктивную нормальную форму, в которой каждая дизъюнкция содержит ровно k литералов.

Пример. Следующая формула записана в 2-КНФ: $(A \vee B) \wedge (\neg B \vee C) \wedge (B \vee \neg C)$.

Определение 23 (Задача 1-in-3-SAT). Задача 1-in-3-SAT — частный случай задачи 3-SAT, где каждая дизъюнкция содержит три литерала, только один из которых может быть правдив (TRUE).

Теорема 3 (Теорема Кука-Левина). Задача CNF-SAT выполнимости является NP-полной. То есть она находится в NP, и любая задача в NP может быть сведена за полиномиальное время детерминированной машиной Тьюринга к булевой задаче выполнимости.

7.2. co-NP-полнота задачи TAUTOLOGY

Определение 24 (Тавтология). Тавтологией называется формула или утверждение, которое верно во всех возможных интерпретациях.

Пример. $x \neq y \vee x = y$.

Определение 25 (co-NP-complete). Язык L называется co-NP-complete, если любой co-NP язык можно свести к этому языку за время, не превосходящее какой-либо полином.