Листок 3

Сводимость, трудность и полнота.

Сперва определим понятие сводимости: язык L в общем смысле сводится к языку L', если существует функция $f : \Sigma^* \to \Sigma^*$ такая, что $x \in L$ тогда и только тогда, когда $f(x) \in L'$.

Такая общая сводимость совершенно неинтересна, для получения каких-то результатов на функцию f накладывают ограничения. В курсе ТФСиА функция f должна была быть вычислимой. Самая важная для нашего курса сводимость – полиномиальная m-сводимость или сводимость по Карпу.

Сводимость по Карпу

Говорят, что язык L_1 сводится к языку L_2 по Карпу (символ $\leqslant_m^p$ или $\leqslant_p$), если функция сводимости полиномиально вычислима. Точнее $L_1 \leqslant_p L_2$ тогда и только тогда, когда существует $f : \Sigma^* \to \Sigma^*$, полиномиально вычислимая некоторым алгоритмом, что $x \in L_1$ эквивалентно $f(x) \in L_2$.

Формально $L_1 \leqslant_p L_2$:

– существует функция f, вычислимая некоторой МТ за полиномиальное время: т.е. на входе $y \in \{0,1\}^*$ МТ останавливается за $p(|y|)$ в состоянии q_{halt}, на ленте после останова написано $f(y)$;

– если $x \in L_1$, то $f(x) \in L_2$;

– если $x \notin L_1$, то $f(x) \notin L_2$;

Символ m в сводимости отсылает к названию: «many-to-one» reduction, сводимость нескольких к одному. Название подчеркивает: функция сводимости вполне может не быть взаимно однозначной.

Трудность и полнота

Рассмотрим произвольный класс сложности $\mathcal{C}$, некоторую сводимость и некоторый язык L.

Если любой язык $L' \in \mathcal{C}$ сводится к L, то, умея разрешать L, мы умеем разрешать любой язык из $\mathcal{C}$: сперва преобразуем вход с помощью функции f, затем проверим принадлежность $f(x)$ языку L.

В некотором смысле задача распознавания L не проще, чем любая задача распознавания из $\mathcal{C}$: научившись решать L, мы можем разрешить и класс $\mathcal{C}$. В этом случае сводимость становится отношением порядка, а язык L становится «трудным» языком для класса $\mathcal{C}$ (по-русски ближе по смыслу было бы сказать «не лёгким»).

Если любой язык $L' \in \mathcal{C}$ сводится к L, то L называют $\mathcal{C}$-трудным или $\mathcal{C}$-hard относительно заданной сводимости.

Если любой язык $L' \in \mathcal{C}$ сводится к L и $L \in \mathcal{C}$, то L называют $\mathcal{C}$-полным или $\mathcal{C}$-complete относительно заданной сводимости.

(co-)$\mathcal{NP}$-полнота.

Класс $\mathcal{NPC}$ он же класс $\mathcal{NP}$-complete – это класс языков, полных для $\mathcal{NP}$ относительно полиномиальной сводимости.

$L \in \mathcal{NP}$-hard если:

$\forall L' \, (L' \in \mathcal{NP} \Rightarrow L' \leqslant_p L)$

$L \in \mathcal{NPC}$ если:

1) $L \in \mathcal{NP}$,

2) $\forall L' \, (L' \in \mathcal{NP} \Rightarrow L' \leqslant_p L)$

$L \in \mathrm{co}\mathcal{NPC}$ если:

1) $L \in \mathrm{co}\mathcal{NP}$,

2) $\forall L' \, (L' \in \mathrm{co}\mathcal{NP} \Rightarrow L' \leqslant_p L)$

Существование хотя бы одного $\mathcal{NP}$-полного языка неочевидно из определения, тем не менее, они существуют. Теорема Кука-Левина утверждает, что задача о выполнимости булевой функции SAT является $\mathcal{NP}$-полной.

Описание некоторых $\mathcal{NP}$-полных задач.

CNF–SAT, она же КНФ-выполнимость: дана КНФ, есть ли набор переменных, выполняющих формулу? Соответствующий язык – язык все выполнимых КНФ.

n–CNF–SAT: дана КНФ, в которой в каждый дизъюнкт входит (“не более” либо “ровно” в зависимости от источника, будем считать, что ровно) n переменных — есть ли набор переменных, выполняющий КНФ? $n \geqslant 3$. Соответствующий язык – язык все выполнимых КНФ с указанным ограничением на размер дизъюнкта.

Clique, она же Клика: дан неориентированный граф G и натуральное число k, есть ли в G клика (полный подграф) на k вершинах? Соответствующий язык: язык пар (G, k) таких, что в графе G есть клика размера k.

IndSet, она же Независимое множество: дан неориентированный граф G и натуральное число k, есть ли в G независимое множество вершин (никакие две не соединены ребром – ещё его называют антикликой) размера k? Соответствующий язык: язык пар (G, k) таких, что в графе G есть антиклика размера k.

VCover, она же Вершинное покрытие: дан неориентированный граф G и натуральное число k, если ли в G вершинное покрытие (подмножество вершин таких, что каждое ребро графа инцидентно по крайней мере одной вершине множества) из k вершин? Соответствующий язык: язык пар (G, k) таких, что в графе G есть вершинное покрытие размера k.

Hamcycle, она же Гамильтонов граф: дан неориентированный граф G, есть ли в нём гамильтонов цикл? Соответствующий язык: язык графов, содержащих граф с гамильтоновым циклом.

Hampath, она же Гамильтонов путь: дан неориентированный граф G, есть ли в нём гамильтонов путь? Соответствующий язык: язык графов, содержащих граф с гамильтоновым путём.

Color, она же Хроматическое число: даны неориентированный граф G и натуральное число k, можно ли раскрасить вершины G в k цветов так, чтобы смежные вершины были окрашены в разные цвета? Соответствующий язык: язык пар (G, k) таких, что граф G можно правильно

раскрасить в k цветов.

n–Color: дан неориентированный граф G, можно ли раскрасить вершины G в n цветов так, чтобы смежные вершины были окрашены в разные цвета? $n \geqslant 3$. Соответствующий язык: язык графов, которые можно правильно раскрасить в n цветов.

Partition, она же Разбиение: дано множество S, состоящее из n натуральных чисел, существуют ли подмножества A и B такие, что $A \cap B = \varnothing$, $A \cup B = S$ и

$$\sum_{a \in A} a = \sum_{b \in B} b?$$

Соответствующий язык: язык всех множеств S, удовлетворяющих указанным требованиям.

n–Partition: дано натуральное число m и множество S, состоящее из mn натуральных чисел, существует ли разбиение S на m непересекающихся множеств, в каждом из которых будет по n чисел, таких, что суммы чисел в каждом множестве равны? $n \geqslant 3$

Subset sum, она же Рюкзак: дано множество натуральных чисел S и натуральное число t, есть ли в S подмножество, сумма элементов которого равна t? Соответствующий язык: язык всех пар (S, t) таких, что в S существует подмножество, сумма элементов которого равна t

Hitting set, она же Протыкающее множество: дано семейство конечных множеств $\{A_1, \ldots, A_m\}$ и натуральное число k, существует ли множество мощности k, пересекающее каждое A_i?

и много, много других ...

Поиск и распознавание.

Большому количеству $\mathcal{NPC}$ задач и, соответственно, $\mathcal{NPC}$ языков соответствуют задачи поиска. Приведём пример:

Clique = $\{(G, k) \mid G$ содержит клику размера $k\}$ – задача распознавания и одновременно язык. Дана пара (G, k), выясним «да, принадлежит» или «нет, не принадлежит».

ExactClique = $\{(G, k) \mid G$ содержит клику размера k и не содержит клику размера $k + 1\}$ – тоже задача распознавания и одновременно язык. Дана пара (G, k), выясним «да, принадлежит» или «нет, не принадлежит».

MaxSizeOfClique$(G) = k$ – задача поиска, на вход подаётся граф G, на выходе нужно получить размер макисмальной клики в графе G – число k.

MaxClique$(G) = G'$ – задача поиска, на вход подаётся граф G, на выходе нужно получить клику максимального размера, являющуюся подграфом G – назовём её G'.

Заметим, Clique – это язык (либо предикат), MaxClique – это не язык, это функция, которая может быть вычислена некоторым алгоритмом.

Задачи

Задача 3.0: Снова не задача:

Пусть $L_1 \leqslant_p L_2$

1) доказать, что если $L_2 \in \mathcal{P}$, то $L_1 \in \mathcal{P}$,

2) доказать, что если $L_2 \in \mathcal{NP}$, то $L_1 \in \mathcal{NP}$,

Задача 3.1:

1) Верно ли, что если $L \in \mathcal{NPC}$ и $L \in \text{co}\mathcal{NP}$, то $\mathcal{NP} = \text{co}\mathcal{NP}$?

2) Верно ли, что если $L \in \mathcal{NP}$ и $L \in \text{co}\mathcal{NPC}$, то $\mathcal{NP} = \text{co}\mathcal{NP}$?

3) Верно ли, что язык $L \in \mathcal{NPC}$ тогда и только тогда, когда $\overline{L} \in \text{co}\mathcal{NPC}$?

4) Доказать, что если $\mathcal{P} = \mathcal{NP}$, то любой нетривиальный язык $\mathcal{NP}$-трудный. (Что насчет тривиальных языков?)

5) Верно ли, что если $L_1 \leqslant_p L_2$, то $\overline{L}_1 \leqslant_p \overline{L}_2$?

Задача 3.2 (видимо, $\mathcal{P} = \mathcal{NP}$?):

По аналогии с обычной SAT определим задачу DNF–SAT: дана ДНФ, нужно проверить, выполнима ли она (т.е. дан язык всех выполнимых ДНФ, нужно построить распознаватель для него)

1) Придумать полиномиальный алгоритм решения DNF–SAT (это очень просто).

2) Построим сводимость CNF–SAT к DNF–SAT: в формуле КНФ раскрываем скобки в силу дистрибутивности операций конъюнкции и дизъюнкции. Доказать корректность сводимости.

3) Поскольку мы свели $\mathcal{NP}$–полную задачу к полиномиально разрешимой, то $\mathcal{P} = \mathcal{NP}$. Есть ли ошибка в рассуждениях?

Задача 3.3 (ну теперь-то $\mathcal{P} = \mathcal{NP}$?):

Построим алгоритм для Subset sum:

Возьмём массив $S[1 \dots n]$ и число T, для каждого значения $1 \leqslant i \leqslant n$ и каждого значения $0 \leqslant t \leqslant T$ определим функцию

$$Subsum(i,t) = \begin{cases} \text{True}, & \text{если некоторое подножество массива } S[i \dots n] \text{ имеет сумму } t, \\ \text{False}, & \text{иначе.} \end{cases}$$

Нам нужно вычислить $Subsum(1,T)$, воспользуемся стандартным трюком динамического программирования:

$$Subsum(i,t) = \begin{cases} \text{True}, & \text{если } t = 0 \text{ или } i = n, t = S[n], \\ \text{False}, & \text{если } t < 0 \text{ или } i > n, \\ Subsum(i+1,t) \vee Subsum(i+1,t-S[i]), & \text{иначе.} \end{cases}$$

Нам нужно вычислить $Subsum(1,T)$.

1) Корректен ли алгоритм, решающий Subset sum? Всего возможны nT значений для нетривиального вычисления функции, так что мы можем вычислять функцию рекурсивно и записывать результаты в таблицу размера $n \times T$. Каждое следующее значение элементарно выражается через предыдущие.

2) Оценить сложность алгоритма. Верно ли, что общая сложность алгоритма есть $O(nT)$?

3) Поскольку алгоритм полиномиален, а задача Subset sum $\mathcal{NP}$–полна, то $\mathcal{P} = \mathcal{NP}$. Есть ли ошибка в рассуждениях?

Задача 3.4 (есть что-то особое в цифре 2):

Описать полиномиальные алгоритмы для решения задач:

а) 2–Partition

б) 2–Color

в) 2–SAT

Задача 3.5:

Построить полиномиальную сводимость:

а) Hampath к Hamcycle

б) Hamcycle к Hampath

Задача 3.6:

Мы рассматривали неориентированные графы на гамильтоновость. Рассмотрим теперь гамильтоновы ориентированные графы. Всюду приставка Dir означает ориентированность графа.

Построить полиномиальную сводимость:

а) Hampath к DirHampath

б) DirHampath к Hampath

Можете повторить то же самое для циклов.

Задача 3.7: где начинается сложность?:

Для каждого натурального k определим язык k–Clique всех графов, в которых есть клика размера k.

1) Язык 1–Clique полиномиален?

2) Язык 2–Clique полиномиален?

3) Язык 3–Clique полиномиален?

4?) Для каких k язык k–Clique $\mathcal{NP}$-полон?

???) Известно, что Clique $\in \mathcal{NPC}$, но из входа (G, k) можно сперва выделить k, а затем запустить алгоритм решения соответствующего k–Clique. Как тогда может быть, что Clique $\in \mathcal{NPC}$?

Задача 3.8: Простые сводимости

Доказать, что следующие языки/задачи принадлежат $\mathcal{NPC}$.

1) Графы, имеющие клику ровно на половине вершин.

2) Графы, имеющие вершинное покрытие ровно на половине вершин.

3) Дан граф G и число k, существует ли простой путь в графе G длины как минимум k?

4) Дан граф G, есть ли в нём простой путь, проходящий ровно по разу через все, кроме 2023 вершин?

5) Дан граф G, есть ли в нём остовное дерево со степенью всех вершин не более 42?

Задача 3.9:

Напомним, задача останова Halt неразрешима.

1) Верно ли, что Halt $\in \mathcal{NPC}$?

2) Верно ли, что Halt $\in \mathcal{NP}$-hard?

Задача 3.10:

Замкнут ли $\mathcal{NPC}$ относительно:

а) объединения,

б) пересечения,

в) конкатенации,

г) итерации (звёздочки Клини)?

д) Если $L_1 \in \mathcal{NPC}$ и $L_2 \in \mathcal{NPC}$, то верно ли, что $L_1 \times L_2 \in \mathcal{NPC}$?