Семинар 3.
$\mathcal{P}$ vs $\mathcal{NP}$

Составил Р. Делла Пиетра

29.2.20

1 Определения

$$DTIME(t(n)) = \{L | \exists MT\text{, разрешающая } L \text{ за } O(t(n)) \text{ тактов}\}$$

$$\mathfrak{F}_p(n) = \{\text{функции, вычислимые за } poly(n), \text{ где } n \text{ длина аргумента}\}$$

$$\mathcal{P} = \bigcup_{c=1}^{\infty} DTIME(n^c) = \{L \mid \exists \chi_L(x) \in \mathfrak{F}_p(|x|),\ \chi_L \text{ — характеристическая функция}\}$$

$$\mathcal{NP} = \{L \mid \exists R(x,y) \in \mathfrak{F}_p(|x|): \quad x \in L \iff \exists y : |y| = poly(|x|), R(x,y) = 1\}$$

$$R(x,y) \text{ — предикат, проверяющий } x, \quad y \text{ — сертификат или подсказка}$$

$$co-\mathcal{NP} = \{L \mid \exists R(x,y) \in \mathfrak{F}_p(|x|): \quad x \notin L \iff \exists y : |y| = poly(|x|), R(x,y) = 0\}$$

Более интуитивные определения такие: $\mathcal{P}$ — класс задач, которые легко решать, $\mathcal{NP}$ — для которых легко проверять решение, а $co-\mathcal{NP}$ — для которых легко опровергать решение .

2 Примеры

2.1 $\mathcal{P}$

2.1.1 $CONNECTED$

Язык описаний связных графов. Покажем, что он лежит в $\mathcal{P}$.
Граф подаётся в виде матрицы смежности, то есть описание графа длины n^2, где n — количество вершин. Для проверки связности воспользуемся алгоритмом поиска в ширину, сложность которого $O(V+E)$. Вершин n, рёбер не более n^2, поэтому сложность $O(n^2)$.
Таким образом, разрешающий алгоритм выполняется за $O(n^2) \implies CONNECTED \in \mathcal{P}$

2.1.2 $EULERCYCLE$

Язык описаний графов с эйлеровым циклом (циклом по всем рёбрам). Известно, что для того, чтобы граф содержал эйлеров цикл, необходимо и достаточно, чтобы все вершины были чётной степени и граф был связным. Связность полиномиально проверять мы уже умеем, а для проверки чётности достаточно просуммировать строки матрицы, обнулив диагонали, и проверить на чётность.

2.1.3 $EVEN$

Множество чётных чисел. Число приходит записанное в двоичной системе, то есть нужно построить алгоритм, проверяющий чётность за $poly(\log n)$ тактов. Если младший бит числа нулевой, число чётное. Построенный алгоритм работает за $O(1)$, поэтому $EVEN \in \mathcal{P}$.

2.2 $\mathcal{NP}$

2.2.1 $COMPOSITE$

Множество составных чисел. Покажем, что оно лежит в $\mathcal{NP}$.
Пусть y — число-делитель x. Предикат будет проверять, что $1 < y < x$, $y \mid x$. Считаем, что в нашей системе операции сравнения и деления происходят за один такт, поэтому наш алгоритм работает за $O(1)$. $y < x \implies |y| < |x|$.

2.2.2 $3 - COLOR$

Язык описаний графов, вершины которого можно раскрасить в 3 цвета так, чтобы соседние вершины были разных цветов. Пусть y — раскраска вершин в 3 цвета. Предикат будет проверять, что в раскраске нет соседних вершин одного цвета. Для этого надо проверить не более $\frac{n(n-1)}{2}$ пар вершин, то есть алгоритм работает за $O(n^2)$. $|y| = n$.

2.2.3 SAT

Язык выполнимых булевых формул. Формулы состоят из некоторого набора переменных $z_1, z_2, \ldots$, операций $\wedge, \vee, \neg$ и скобок.
Пусть y — набор значений для переменных (n единиц и нулей). Предикат будет подставлять значения в формулу и «схлопывать» её, используя таблицу операций:

$$z_1 \wedge (z_2 \vee \neg z_3) \quad \to \quad 0 \wedge (1 \vee \neg 0) \quad \to \quad 0 \wedge (1 \vee 1) \quad \to \quad 0 \wedge 1 \quad \to \quad 0$$

Таких схлопываний произойдёт не более, чем количество операций, а операций меньше, чем длина входа, поэтому предикат лежит в $\mathfrak{F}_p(|x|)$. Очевидно, ограничение на длину y также выполнено.

2.3 $co - \mathcal{NP}$

2.3.1 $TAUT$

Язык тавтологичных булевых формул. Пусть y — набор значений для переменных (n единиц и нулей), а предикат подставляет, «схлопывает» и проверяет результат. Если есть такой y, что формула обнуляется, это не тавтология.

3 Сводимости

Полиномиальное обобщение m-сводимости.

$$A \leqslant_p B : \quad \exists f \in \mathfrak{F}_p(|x|) : x \in A \iff f(x) \in B$$

Определим ещё несколько релевантных классов задач:
$A \in \mathcal{NP} - hard : \quad \forall B \in \mathcal{NP}\ B \leqslant A.$
$A \in \mathcal{NP} - complete\ \text{или}\ \mathcal{NP}c : \quad A \in \mathcal{NP} \bigcap \mathcal{NP} - hard.$

3.1 Теорема Кука-Левина

Язык выполнимых булевых формул в конъюнктной нормальной форме (SAT) $\mathcal{NP}$-полный.

3.2 Примеры задач

3.2.1 Доказать, что язык $CLIQUE \in \mathcal{NP}c$

Это язык описаний графов и числа k, содержащих полный подграф размера k. Легко проверить, что язык лежит в $\mathcal{NP}$: y — номера вершин, образующих клику, и предикат проверяет $\frac{n(n-1)}{2}$ ячеек в матрице. Покажем, что SAT сводится к $CLIQUE$. Сводимость можно посмотреть здесь.
Если КНФ состоит из k дизъюнктов, в построенном графе будет клика размера k тогда и только тогда, когда в дизъюнкт выполним.
Таким образом, любая $\mathcal{NP}$ задача сводится к SAT, а SAT сводится к $CLIQUE$, поэтому любая $\mathcal{NP}$ задача сводится к $CLIQUE$, то есть $CLIQUE \in \mathcal{NP}c$