МФТИ, сложность вычислений, осень 2023
Семинар 07. Диагонализация

Определение. Функция $T : \mathbb{N} \to \mathbb{N}$ называется конструируемой (по времени), если $\forall n \in \mathbb{N} : T(n) \geqslant n$, а также существует машина Тьюринга M, вычисляющая $T(n)$ на входе 1^n за время $O(T(n))$.
Определение. Функция $S : \mathbb{N} \to N$ называется конструируемой по памяти, если $\forall n \in \mathbb{N} : S(n) \geqslant \log_2 n$, а также существует машина Тьюринга, вычисляющая $S(n)$ на входе 1^n, используя память $O(S(n))$.
Теорема Пусть f, g — конструируемые по времени функции, причём $f(n) \log f(n) = o(g(n))$. Тогда $\mathbf{DTIME}(f(n)) \subsetneq \mathbf{DTIME}(g(n))$.
Теорема Пусть f, g — конструируемые по памяти функции, причём $f(n) = o(g(n))$. Тогда $\mathbf{DSPACE}(f(n)) \subsetneq \mathbf{DSPACE}(g(n))$.
Теорема Пусть f, g — конструируемые по времени функции, причём $f(n+1) = o(g(n))$. Тогда $\mathbf{NTIME}(f(n)) \subsetneq \mathbf{NTIME}(g(n))$.

---

1. Пусть O — некий язык. Пусть $\mathcal{C}$ — множество языков, полиномиально сводящихся к O. Чем отличаются $\mathcal{C}$ и $\mathbf{P}^O$?
2. Предположим, что существует полиномиальный алгоритм, который на входе (φ_1, φ_2) выдаёт правильный ответ на вопрос выполнимости каждой из формул φ_1, φ_2 по отдельности, обратившись к оракулу SAT не более одного раза. Докажите тогда, что $\mathbf{P} = \mathbf{NP}$.
3. Докажите, что в определении конструируемости фразу «на входе 1^n» можно заменить на
а) на любом входе длины n;
б) на входе n, записанном в двоичной записи.
4. Докажите, что если язык распознаётся за $o(n)$, то он распознаётся и за $O(1)$.
5*. Докажите, что если язык распознаётся на одноленточной машине Тьюринга за время $o(n \log n)$, то он является регулярным. Покажите, что для многоленточных машин это уже неверно.
6. Докажите теорему Бородина — Трахтенброта: если $g(n) \geqslant n$ — вычислимая функция, то существует вычислимая $t(n) \geqslant n$, для которой $\mathbf{DTIME}(t(n)) = \mathbf{DTIME}(g(t(n)))$.
Отсюда следует, например, что существует [не конструируемая по времени] функция $T(n)$, такая что всякий язык, разрешимый за время $O(T(n))$, также разрешим за $O(\log T(n))$.
7. Пусть $k \geqslant 1$. Докажите, что $\mathbf{P} \neq \mathbf{DSPACE}(n^k)$. Докажите, что $\mathbf{NP} \neq \mathbf{DSPACE}(n^k)$.
8. Пусть f, g, h — конструируемые по времени функции. Пусть $\mathbf{DTIME}(f(n)) = \mathbf{DTIME}(g(n))$. Докажите, что $\mathbf{DTIME}(f(h(n))) = \mathbf{DTIME}(g(h(n)))$. При необходимости можете считать, что f, g, h — строго монотонно возрастают.
9. Пусть $BB(n)$ — максимальное (конечное) время работы машины Тьюринга не более чем с n состояниями на пустом входе. Покажите, что $BB(n)$ не является вычислимой.
10. Пусть C — случайный оракул, конструируемый следующим образом. Для каждого n (независимо от остальных) в C с вероятностью $1/2$ не лежит ни одно слово длины n, а с вероятностью $1/2$ лежит ровно одно случайное слово длины n. Докажите, что $\mathbf{P}^C \neq \mathbf{NP}^C$ с вероятностью 1.
11*. Докажите, что существует универсальная машина Тьюринга, которая на вход принимает (α, x) и моделируем поведение M_α на входе x, причём если M_α останавливается за время $T(n)$, то универсальная машина останавливается за $O(T(n) \log T(n))$ [константа внутри O зависит от α].

1. При сводимости можно запрашивать оракул O только один раз, а в $\mathbf{P}^O$ число запросов полиномиально.

2. Из условия следует, что существует такой полиномиальный алгоритм B, который конструирует $\psi = \psi(\varphi_1, \varphi_2)$, из (не-)выполнимости которой можно вывести (не-)выполнимость φ_1 и φ_2. Аналогично, можно применить B многократно и свети вопрос о выполнимости полиномиального числа формул к выполнимости всего одной.

Значит, существует такой алгоритм A, принимающий список формул $\varphi_1, \ldots, \varphi_k$ и бит b, такой что $A(\varphi_1, \ldots, \varphi_k, b)$ выдаёт правильные ответы на вопросы о выполнимости всех k формул по крайней мере при одной значении бита b (этот бит отвечает за выполнимость $\psi(\varphi_1, \ldots, \varphi_k)$).

Научимся тогда проверять φ на выполнимость за полиномиальное время. Будем постепенно фиксировать значения переменных в φ, получая тем самым новое множество формул, которые нужно проверить на выполнимость. Применим к ним A с обоими значениями b. Тогда все формулы разобьются не более чем на 4 класса. Если две формулы имеют одинаковую выполнимость при $b = 0$ и $b = 1$, то достаточно оставить лишь одну из них. Тем самым, на каждом шаге остаётся не более 4 формул. В конце каждую из них проверяем на выполнимость.

3.

а) Можно не различать единицы и нули.

б) Можно сначала превратить n в 1^n за $O(n)$.

4. Рассмотрим такой n_0, что на всех входах длины n_0 машина останавливается менее чем за n_0 шагов. Тогда машина не отличает входы длины n_0 от входов большей длины.

5*. См. https://core.ac.uk/download/pdf/81988943.pdf. В качестве примера можно рассмотреть $\{0^n 1^n \mid n$ — уелое положительное$\}$.

6. Идея следующая: будем строить t так, что если машина не остановилась за $t(n)$ шагов, но не остановится и за $g(t(n))$. Занумеруем все машины Тьюринга. Пусть $\mathrm{time}_i(n)$ — максимальное время работы M_i среди всех входов длины n (возможно, это число равно $+\infty$). Тогда подойдёт

$$t(n) = \min\left\{k \mid k > t(n-1) \text{ и } \forall i \in \{1, 2, \ldots, n\} : \mathrm{time}_i(n) < k \text{ или } \mathrm{time}_i(n) > n \cdot g(k)\right\}.$$

Далее, покажем, что если $A \in \mathbf{DTIME}(g(t(n)))$, то $A \in \mathbf{DTIME}(t(n))$. Действительно, пусть A распознаётся машиной M_i за время $\leqslant C \cdot g(t(n))$. При $n > \max\{C, i\}$ неравенство $\mathrm{time}_i(n) > n \cdot g(k)$ (для $k = t(n)$) выполняться не может, так что выполняется $\mathrm{time}_i(n) < k$.

В частном случае, достаточно выбрать $g(n) = 2^n$, затем положить $T(n) = 2^{t(n)}$.

7. Пусть $\mathbf{P} = \mathbf{DSPACE}(n^k)$. Пусть $A \in \mathbf{DSPACE}(n^{k+1})$. Тогда $A' = \{x01^{|x|^{k+1}} \mid x \in A\} \in \mathbf{DSPACE}(n)$, то есть $A' \in \mathbf{P}$. Но тогда и $A \in \mathbf{P}$. Значит, $\mathbf{DSPACE}(n^{k+1}) \subset \mathbf{DSPACE}(n^k)$, что неверно.

8. Примените пэддинг.

9. Пусть есть машина M с k состояниями, вычисляющая $BB(\cdot)$. Построим семейство машин M_n: сначала она выписывает на вход число n, затем вычисляем $BB(n)$ и проделывает $BB(n)$ действий (например, вычитает из числа единицу, пока оно не станет нулём). Ясно, что для такой машины достаточно $k + O(\log n) + O(1)$ состояний. Но время её работы превосходит $BB(n)$.

10. Рассмотрим язык $U = \{1^n \mid \exists x \in \{0,1\}^n \cap C\}$. Ясно, что $U \in \mathbf{NP}^C$: в качестве сертификата достаточно передать то самое слово из C.

Однако алгоритмы из $\mathbf{P}^C$ на тех входах 1^n, на которых существует слово $x \in \{0,1\}^n \cap C$, зададут вопрос $x \stackrel{?}{\in} C$ с вероятностью $\frac{\mathrm{poly}(n)}{2^n}$, а значит, вернут такой же ответ, как если бы такого x не существовала.

11*. См. книгу Ароры — Барака (https://theory.cs.princeton.edu/complexity/book.pdf), раздел 1.A.