Лекция 3

Иерархия по памяти

(Конспект: М. Плискин)

Определение 3.1. Для данной функции f, $\mathbf{DTime}(f(n))$ обозначает класс языков, принимаемых детерминированными машинами Тьюринга, заканчивающими свою работу за время, не превосходящее $f(n)$, где n — длина входа[1]. $\mathbf{DSpace}(f(n))$ обозначает класс языков, принимаемых детерминированными машинами Тьюринга, использующие не более $f(n)$ ячеек памяти на рабочих лентах (входная *readonly* лента — не в счет). Аналогично определяются и другие подобные этим классы, только буква $\mathbf{D}$ заменяется на другие[2]. Если вместо функции f указан класс функций (например, $O(n)$), подразумевается объединение соответствующих классов: $\mathbf{DTime}(\mathcal{F}) = \bigcup\limits_{f\in\mathcal{F}} \mathbf{DTime}(f)$.

Определение 3.2. Для любого вычислительного устройства M обозначим $L(M)$ принимаемый им язык (понятие *принятия* строки различно для разных типов вычислительных устройств).

3.1 Небольшое количество памяти, помогающее в вычислениях

Замечание 3.1. Отметим, что машина Тьюринга с лентой длины $O(1)$ есть конечный автомат.

[1]Вообще, если не сказано иного, n везде обозначает именно длину входа.

[2]Неполный список сокращений: $\mathbf{N}$ — недетерминированный, $\mathbf{R}$ — вероятностный с ограниченной односторонней ошибкой, $\mathbf{B}$ — вероятностный с ограниченной двусторонней ошибкой, $\mathbf{BQ}$ — квантовый с ограниченной двусторонней ошибкой. Значение этих слов будет пояснено позже.

Теорема 3.1. *Покажем, что* $\mathbf{DSpace}(O(\log\log n)) \neq \mathbf{DSpace}(O(1))$.

Доказательство. Опишем язык D:

$$D = \left\{ \underbrace{\overbrace{0\dots00}^{=0}}_{k}\$\underbrace{\overbrace{0\dots01}^{=1}}_{k}\$\dots\$\underbrace{\overbrace{1\dots11}^{=2^k-1}}_{k}\$ \;\middle|\; k \in \mathbb{N} \right\}.$$

Покажем, что данный язык не принадлежит классу $\mathbf{DSpace}(O(1))$. Для этого воспользуемся так называемой леммой о разрастании (pumping lemma), доказательство которой можно найти в любой книге по теории формальных языков.

> **Лемма 3.1.** *Для любого детерминированного конечного автомата A существует такое $n \in \mathbb{N}$, что каждая цепочка $w \in L(A)$ длины не менее n представима в виде $w = \alpha\beta\gamma$, причем $\alpha\beta^r\gamma \in L(A)$ для любого $r \in \mathbb{N}$.*

Допустим, что наш язык D принадлежит классу $\mathbf{DSpace}(O(1))$, то есть он распознается некоторым детерминированным конечным автоматом. Тогда для него выполняется лемма о разрастании. Рассмотрим строку длины не менее n и попытаемся найти повторяющуюся подстроку β. Поскольку с ростом строки начальный блок из нулей должен расти, строка β должна начинаться с нуля (причем это нуль, стоящий до первой единицы). По аналогичной причине она должна оканчиваться на единицу. Но тогда для любого $r \in \mathbb{N}$ строка $\alpha\beta^r\gamma$ начинается на одно и то же количество нулей, что для достаточно больших r противоречит $\alpha\beta^r\gamma \in L(A)$. (Другое доказательство: количество символов \$ в каждом слове языка D является степенью двойки, значит, по лемме о разрастании все они должны входить в строку β, что противоречит тому, что расстояние между двумя символами \$ увеличивается с ростом длины строки.) Таким образом, $D \notin \mathbf{DSpace}(O(1))$.

Построим теперь машину Тьюринга, принимающую наш язык, используя $O(\log\log n)$ ячеек памяти. Пусть на вход машине подана строка из нашего языка, соответствующая некоторому натуральному k. Машина будет в этом случае работать в k этапов. На i-м этапе задачей машины будет проверить, что группы из i последних битов (счетчик i занимает $O(\log k) = O(\log\log n)$ битов), взятых из каждой из 2^k подстрок нашей строки, заключенных между знаками \$, образуют арифметическую прогрессию по модулю 2^i с разностью 1. Например, для $i = 1$ надо проверить, что последние биты в каждой группе чередуются: $0, 1, 0, 1, \dots, 0, 1$. Для $i = 2$ надо убедиться в том, что два последних бита образуют соответствующую последовательность: $00, 01, 10, 11, 00, 01, \dots, 11$. По окончании всех k этапов следует проверить, что $(k+1)$-й символ каждой группы — действительно \$. Очевидно, что для строк из D все эти проверки пройдут. Если же строка не принадлежит D, то рано или поздно алгоритм обнаружит нарушение структуры, и строка будет отвергнута.

Единственный момент, который не был рассмотрен до сих пор — это собственно организация процесса сравнения групп битов. Пусть у нас есть группа битов A, кодирующая некоторое число s, и группа битов B, кодирующая число t. Нам необходимо убедиться, что $t = s+1$. Для этого достаточно подсчитать длину непрерывной последовательности единиц в A, начинающейся с крайнего правого бита, и проверить, что в B справа стоит точно такое же количество нулей. Следующие цифры должны быть различны, а оставшаяся часть должна совпадать. Зная i и k, нетрудно (при помощи счетчика, не превосходящего k) добраться с нужной позиции A до нужной позиции B, и наоборот (вспомним, что A и B начинаются сразу за соседними символами \$ или, для самой первой группы битов, началом строки). Очевидно, что размер нужного нам счетчика — также $O(\log\log n)$. $\square$

3.2 Совсем маленькое количество памяти, не помогающее в вычислениях

Теорема 3.2. $\forall \varepsilon > 0$ $\mathbf{DSpace}(O((\log\log n)^{1-\varepsilon})) = \mathbf{DSpace}(O(1))$.

Доказательство. Рассмотрим машину, которая затрачивает не более $S(n) = O((\log\log n)^{1-\varepsilon})$ памяти на каждом входе длины n, и покажем, что, начиная с некоторой длины входа N наша машина использует одно и то же количество ячеек ленты независимо от длины входа.

Для дальнейшего рассмотрения введем следующее определение: *псевдоконфигурацией* машины Тьюринга мы будем называть конфигурацию, в которой вместо позиции считывающей головки на *входной* ленте хранится сам символ, на который эта головка указывает.

Отметим сначала, что если алфавит машины Тьюринга есть множество $\{0,1\}$, то общее количество псевдоконфигураций есть $C(n) = O(S(n)2^{S(n)})$ (здесь первое $S(n)$ нужно для того, чтобы учесть номер позиции на *рабочей* ленте). Рассмотрим теперь последовательность псевдоконфигураций для состояния, характеризующегося некоторым *фиксированным* положением читающей головки на ленте. Ясно, что нас интересуют лишь последовательности длины не более $C(n)$ (поскольку позиция головки на входной ленте зафиксирована, более длинные последовательности псевдоконфигураций неизбежно ведут к зацикливанию, так как их количество — это количество разных *конфигураций*). Таким образом, общее число таких последовательностей можно оценить как

$$C(n)^{C(n)} = O\left(S(n)2^{S(n)}\right)^{O\left(S(n)2^{S(n)}\right)} = O\left(S(n)2^{(S(n))^2\cdot 2^{S(n)}}\right) =$$

$$= O(n^{(\log\log n)^{2-2\varepsilon}\cdot\log\log\log n/(\log n)^{\varepsilon}}) = o(n).$$

Для всех n, больших некоторого $N \in \mathbb{N}$, эта функция $< \frac{n}{2}$. Мы утверждаем, что наша машина не использует более $S(N)$ ячеек ленты ни для какого входа длиннее N.

Пусть x' — самый короткий вход (из входов длины $> N$), на котором используется более $S(N)$ памяти; n' — его длина. Заметим, что x' можно представить в виде

$$\begin{array}{lccccccccc} x' & = & \alpha & a & \beta & a & \gamma & a & \delta & \\ \text{(номера позиций:} & & & i & & j & & k & &), \end{array}$$

где позициям i, j, k соответствует одна и та же последовательность псевдоконфигураций (для каждой позиции рассматриваются псевдоконфигурации, существующие в моменты, когда головка находится именно в этой позиции). Такое представление возможно, так как есть n' позиций входной ленты, но менее $n'/2$ последовательностей псевдоконфигураций.

Покажем теперь, что каждая из псевдоконфигураций, встречающихся при работе на строке x', встречается также и при работе хотя бы на одной из строка $\alpha a\gamma a\delta$ и $\alpha a\beta a\delta$ (это будет означать, что машина затрачивает на x' столько же памяти, сколько и на более короткой строке, что противоречит предположению о минимальности x'). Именно, соответствующая псевдоконфигурация должна встретиться при работе на соответствующей строке (в момент чтения β — при работе на $\alpha a\beta a\delta$, в момент чтения γ — при работе на $\alpha a\gamma a\delta$).

Сравним работу машины на $\alpha a\gamma a\delta$ и на x'. Соответствующие «протоколы» работы не будут отличаться до тех пор, пока машина не сдвинется впервые правее i-й позиции входной ленты (пусть до этого момента последовательность псевдоконфигураций в i-й позиции — $C_1, C_2, \ldots, C_t$). Дальнейшее поведение машины на x' нас не интересует до тех пор, пока она не выйдет за пределы подстроки $a\beta a$. Имеется два способа выйти: направо (правее k-й позиции) и налево. Рассмотрим первый случай.

Заметим, что при работе на x' последовательность псевдоконфигураций в k-й позиции — та же самая: $C_1, C_2, \ldots, C_t$. Следовательно, машина впервые сдвинется правее k-й позиции в конфигурации C_t. Ее дальнейшая работа не будет отличаться от дальнейшей работы на $\alpha a\gamma a\delta$, поскольку псевдоконфигурации и считываемые символы на входной строке совпадают — и так до того момента, пока головка не сдвинется левее соответствующей (в одном случае — i-й, в другом — k-й) позиции, и т. д. (Случай, когда первый выход из β — налево, аналогичен.)

Случай $\alpha a \beta a б$ также может быть рассмотрен аналогично. $\square$

3.3 Большее количество памяти, помогающее в вычислениях

Определение 3.3. Для классов, определяемых с ограничением по памяти, мы будем рассматривать ограничения, являющиеся *«space constructible»* функциями. Функция f является таковой, если существует детерминированная машина Тьюринга, вычисляющая (двоичное представление) $f(n)$ на входе $\underbrace{1\ldots1}_{n}$ и затрачивающая на это память не более $f(n)$.

Замечание 3.2. Другие классы могут оказаться совершенно неестественными; даже большая невычислимая добавка к ограничению на самом деле может не добавить силы вычислениям.

Теорема 3.3. $\mathbf{DSpace}(S_1(n)) \neq \mathbf{DSpace}(S_2(n))$, *где* $S_1(n) = o(S_2(n))$ *и* $S_1(n) \geq \log n$ *для всех* $n > n_0$.

Доказательство. Рассмотрим следующий язык (обозначая $\langle M\rangle$ запись машины M — т.е., по существу, ее функции перехода):

$$\left\{ x = \langle M\rangle 01^k \;\middle|\; \begin{array}{l} k \in \mathbb{N} \cup \{0\}, \\ |\langle M\rangle| < \dfrac{1}{10} S_2(|x|), \\ M \text{ отвергает } x \text{ с памятью } \leq \dfrac{1}{10} S_2(|x|) \end{array} \right\}.$$

Этот язык мы можем распознать, используя $S_2(|x|)$ памяти. Покажем, что его нельзя распознать, потратив только $S_1(|x|)$ памяти. Действительно, пусть существует машина M_1, которая производит такое распознавание. Тогда для некоторого N и для всех $n > N$ справедливо $S_1(n) < \frac{1}{10} S_2(n)$. Рассмотрим число n, большее N и большее $|\langle M_1\rangle|$. Если строка M_1 принимает строку $\langle M_1\rangle 01^{n-|\langle M_1\rangle|-1}$, то она по определению нашего языка ему не принадлежит, а если отвергает, то принадлежит! $\square$

Можно доказать аналогичную теорему об иерархии по времени.

Определение 3.4. Для классов, определяемых с ограничением по времени, мы будем рассматривать ограничения, являющиеся *«time constructible»* функциями. Функция f является таковой, если существует детерминированная машина Тьюринга, вычисляющая (двоичное представление) $f(n)$ на входе $\underbrace{1\dots1}_{n}$ и затрачивающая на это не более $f(n)$ шагов.

Теорема 3.4. $\mathbf{DTime}(S_1(n)) \neq \mathbf{DTime}(S_2(n))$, *где*

$$\frac{S_1(n)\log S_2(n)}{S_2(n)} \to 0$$

и $S_1(n) \geq O(n)$ *начиная с некоторого места.*

Упражнение 3.1. Докажите теорему 3.4. $\square$