Содержание

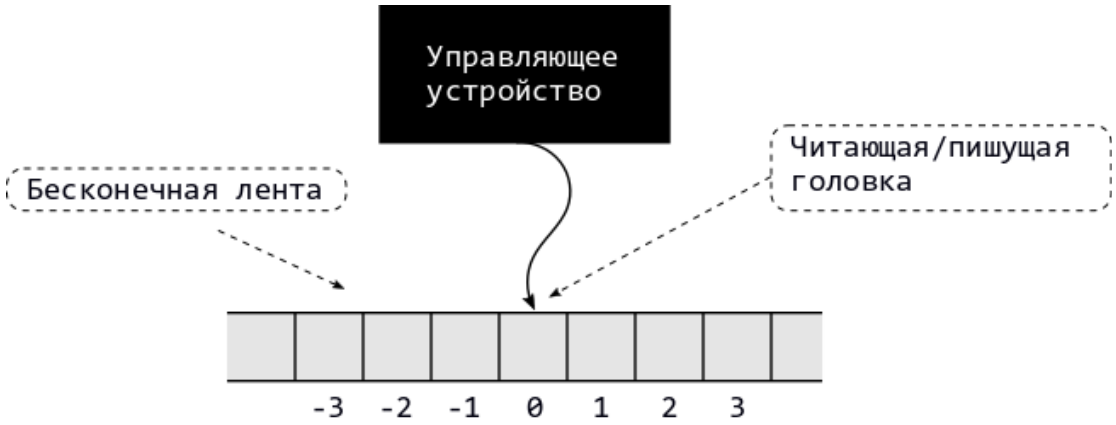
Алгоритмы и сложность

Когда мы даём определение какому-либо понятию, мы связываем его с другими понятиями, а те в свою очередь мы можем определить через другие. Рано или поздно мы придём к таким понятиям, которые нельзя определить через другие. К таким первичным понятиям относится понятие алгоритма.

На неформальном уровне можно сказать, что *алгоритм* – это точная и однозначно определённая последовательность элементарных операций, при помощи которых решаются строго определённые однотипные задачи.

Это предложение не следует принимать за точное определение. Оно лишь поясняет значение слова алгоритм, но не определяет его. Алан Тьюринг в 30-ые годы XX века предложил формализацию понятия алгоритма. Он определил алгоритм как программу, выполняемую на гипотетическом вычислительном устройстве, которое сейчас называется детерминированной однолентоточной машиной Тьюринга.

Определение. *Детерминированной однолентоточной машиной Тьюринга* (*ДМТ*) называется вычислительная модель, состоящая из управляющего устройства с конечным числом состояний, читающей и пишущей головки, которая может считывать и записывать символы на бесконечной ленте, разделённой на ячейки, которые занумерованы целыми числами, а также совершать сдвиг вправо или влево на одну ячейку.



Изначально управляющее устройство находится в начальном состоянии, головка управляющего устройства обозревает нулевую ячейку, а на ленте, начиная с нулевой ячейки, записывается *входное слово* (одна буква слова пишется в одну клетку). Далее машина совершает несколько тактов до тех пор пока управляющее устройство не перейдёт в одно из двух заключительных состояний. За один такт

а) головка считывает символ из обозреваемой ячейки и/или записывает символ в обозреваемую ячейку,

б) головка совершает сдвиг влево или вправо на одну ячейку,

в) управляющее устройство переходит в следующее состояние.

Определение. *Программа для ДМТ* (или *ДМТ-программа*) – это семёрка (Q, Σ, Γ, δ, q_0, q_{accept}, q_{reject}), где

а) Q – конечное множество состояний управляющего устройства;

б) Σ – алфавит, входных символов, не содержащий пустой символ ε;

в) Γ – алфавит символов, которые записываются на ленте, $\varepsilon \in \Gamma$ и $\Sigma \subset \Gamma$;

г) δ: $Q \times \Gamma \to Q \times \Gamma \times \{L,R\}$ – функция переходов;

д) q_0 – начальное состояние управляющего устройства, $q_0 \in Q$;

е) q_{accept}, q_{reject} – заключительные состояния управляющего устройства, при этом $q_{accept}, q_{reject} \in Q$ и $q_{accept} \neq q_{reject}$;

Входное слово x принимается машиной, если после его обработки управляющее устройство находится в заключительном состоянии q_{accept}, и не принимается, если на слове x машина завершает свою работу в заключительном состоянии q_{reject}.

Пример. Напишем программу для ДМТ, которая принимает только слова в алфавите Σ = {0, 1}, содержащие хотя бы один символ 0. Пусть

Q = {q_0, q_{accept}, q_{reject}} – множество состояний управляющего устройства;

Σ = {0, 1} – алфавит входных символов;

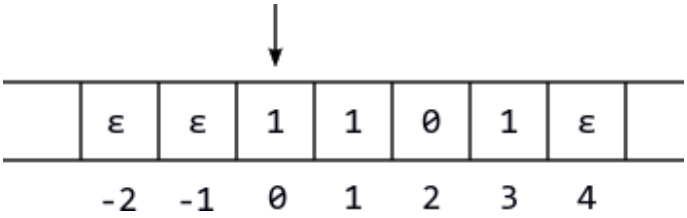
Γ = {0, 1, ε} – алфавит ленточных символов;

Функция переходов

δ	0	1	ε
q_0	$(q_{accept},0,R)$	(q_0,ε,R)	$(q_{reject},\varepsilon,R)$
q_{accept}	-	-	-
q_{reject}	-	-	-

Обработаем входное слово x = 1101.

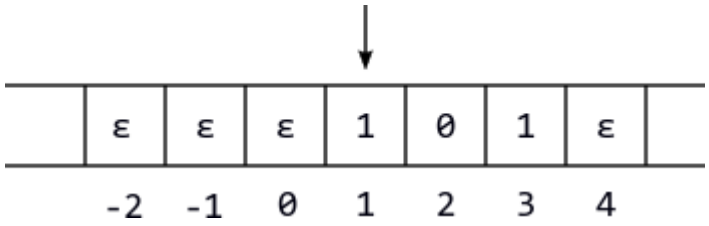
Полагаем состояние управляющего устройства, равным q_0. Записываем слово x на ленту



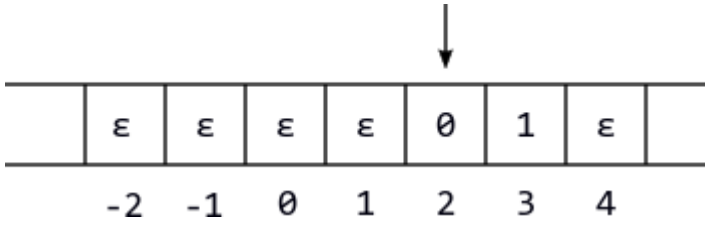
Такт 1. Текущее состояние управляющего устройства – q_0, символ в ячейке, обозреваемой головкой, - 1. Смотрим на значение функции переходов на паре (q_0, 1):

$$\delta(q_0, 1) = (q_0,\varepsilon,R).$$

Значение (q_0,ε,R) состоит из трёх компонент. Первая компонента указывает новое состояние управляющего устройства. Полагаем состояние управляющего устройства, равным q_0. Вторая компонента – это символ, который необходимо записать в ячейку, обозреваемую головкой. Головка указывает на нулевую ячейку. Мы затираем имеющееся в этой ячейке значение и записываем в неё вторую компоненту, т.е. ε. Третья компонента указывает в какую сторону необходимо сместить головку на одну ячейку, R – перемещаем головку вправо, L – перемещаем головку влево. В результате переходим к следующей конфигурации ДМТ.



Такт 2. Текущее состояние управляющего устройства – q_0, символ в ячейке, обозреваемой головкой, - 1. В соответствии с функцией переходов, мы должны установить состояние управляющего устройства в q_0, записать в обозреваемую ячейку символ ε и переместить головку вправо на одну позицию. Получим следующую конфигурацию



Такт 3. Текущее состояние управляющего устройства – q_0, символ в ячейке, обозреваемой головкой, - 0. Смотрим на значение функции переходов на паре (q_0, 0):

$$\delta(q_0, 0) = (q_{accept}, 0, R).$$

Переводим управляющее устройство в состояние q_{accept}. Записываем в обозреваемую ячейку символ 0 и перемещаем головку на одну позицию вправо. Управляющее устройство приняло одно из двух заключительных состояний. Работа ДМТ на этом завершена. Поскольку финальное состояние – q_{accept}, то входное слово x = 1101 принимается (если бы финальное состояние было q_{reject}, то слово x не принималось бы). Машина обработала слово x, совершив три такта.

Конец примера.

Упражнение. Напишите программу для ДМТ, которая принимает только слова в алфавите {0, 1}, в которых есть хотя бы два нуля.

Пусть x – входное слово в алфавите Σ. Число тактов, которое совершает машина, при обработке слова x называется временем работы машины на слове x и обозначается так $T(x)$.

Временно́й сложностью ДМТ-программы называется функция T, которая каждому натуральному числу n ставит в соответствие максимальное время работы машины на любом входном слове длины n в алфавите Σ, т.е. T: $N \to N$

$$T(n) = \max_{x \in \Sigma^+, |x|=n} T(x).$$

Пусть f: N → N – произвольная функция, которая ставит в соответствие каждому натуральному числу некоторое натуральное число. Мы будем говорить, что временная сложность T ДМТ-программы есть $O(f)$, если найдутся такие натуральные числа c и n_0, что для любого натурального числа $n > n_0$ имеет место следующее неравенство

$$T(n) \le c \cdot f(n).$$

Будем говорить, что ДМТ-программа имеет полиномиальную временную сложность, если существует полином $f(n) = a_0 n^k + a_1 n^{k-1} + \ldots + a_k$ такой, что $T = O(f)$, где T – это временная сложность ДМТ-программы.

Недетерминированная одноленточная машина Тьюринга отличается от детерминированной только тем, что в любой момент времени управляющее устройство машины может переходить сразу в несколько состояний (а не ровно в одно состояние). Таким образом, программа для недетерминированной машины Тьюринга (НДМТ) отличается от программы для ДМТ только в виде функции переходов

$$\delta: Q \times \Gamma \to 2^{(Q \times \Gamma \times \{L,R\})}.$$

Недетерминированная машина Тьюринга завершает свою работу как только хотя бы одно из её многочисленных состояний является заключительным.

Процесс вычислений на недетерминированной машине Тьюринга напоминает выращивание корневого дерева, в котором корень – это вершина, соответствующая начальному состоянию машины, при этом каждый раз когда машина принимает одновременно несколько состояний возникает несколько веток

(по одной ветке для каждого из состояний), в каждой из которых независимым образом реализуются вычисления на следующих тактах.

Проблемы

Проблема – это некоторый общий вопрос, на который следует дать ответ. Проблема содержит несколько параметров, или свободных переменных, конкретные значения которых не определены. Проблема определяется следующими компонентами:

а) список всех параметров проблемы;

б) формулировка вопроса, на который необходимо дать ответ.

Мы здесь ограничимся задачами, в которых ответами на вопрос могут быть только либо да, либо нет.

Пример. Рассмотрим примеры проблем.

1. Список параметров проблемы состоит из двух элементов, один из которых представляет собой последовательность натуральных чисел $a = a_0, a_1, \ldots, a_k$, а другой – натуральное число b. Вопрос, на который необходимо ответить: есть ли среди чисел последовательности a число b?

2. Список параметров состоит из одного элемента – графа $G = (V, E)$. Требуется ответить на следующий вопрос: является ли граф G связным?

Придавая произвольные конкретные значения параметрам проблемы мы будем получать индивидуальные задачи этой проблемы. Например, если в первой проблеме из примера выше в качестве последовательности чисел a взять последовательность 1, 2; а в качестве числа b взять число 3, то мы получим индивидуальную задачу, в которой требуется ответить на вопрос: есть ли среди чисел 1, 2 число 3?

Рассмотрим произвольную проблему П. Индивидуальные задачи проблемы П кодируются словами в алфавите $\Sigma = \{0,1,/\}$, которые используются как входные слова для машины Тьюринга. При этом любые две задачи кодируются разными словами и по коду индивидуальной задачи можно однозначно восстановить саму задачу. Обычно, индивидуальные задачи кодируются числовыми характеристиками, при этом сами числа записываются в двоичном представлении. Например, индивидуальную задачу $a = 1, 2$ и $b = 3$ можно закодировать словом x = 01/10/11, в котором первые два символа составляют двоичную запись числа 1, четвёртый и пятый символы – двоичную запись числа 2 и последние два символа – двоичную запись числа 3.

Под детерминированным (соответственно, недетерминированным) алгоритмом, решающим проблему П, мы будем подразумевать программу для детерминированной (соответственно, недетерминированной) одноленточной машины Тьюринга, которая удовлетворяет следующим условиям:

1) если в качестве входного слова взять слово, являющееся кодом произвольной индивидуальной задачи проблемы П, то детерминированная (соответственно, недетерминированная) машина Тьюринга за конечное число тактов достигнет одного из двух заключительных состояний q_{accept} или q_{reject} и остановится;

2) после обработки входного слова, являющегося кодом произвольной индивидуальной задачи (проблемы П) с ответом да, детерминированная (соответственно, недетерминированная) машина Тьюринга завершит свою работу в состоянии q_{accept};

3) после обработки входного слова, являющегося кодом произвольной индивидуальной задачи (проблемы П) с ответом нет, детерминированная (соответственно, недетерминированная) машина Тьюринга завершит свою работу в состоянии q_{reject}.

Проблема называется алгоритмически разрешимой, если существует детерминированный алгоритм, решающий проблему П.

Существуют алгоритмически неразрешимые проблемы. Одной из таких проблем является проблема самоприменимости, которая формулируется следующим образом. Зафиксируем алфавит входных символов, который включает в себя 0, 1 и возможно ещё некоторое конечное число символов. Также зафиксируем алфавит ленточных символов Г. Тогда множество детерминированных алгоритмов является счётным, т.е. все детерминированные алгоритмы можно занумеровать натуральными числами. Проблема самоприменимости включает один параметр n – номер алгоритма. Требуется ответить на вопрос: остановится ли ДМТ с алгоритмом под номером n на входном слове x, где x – двоичная запись числа n.

Утверждение. *Проблема самоприменимости алгоритмически неразрешима*.

Без доказательства.

Классы проблем P и NP

Будем говорить, что проблема П решается за полиномиальное время, если существует детерминированный алгоритм с полиномиальной временной сложностью, решающий проблему П. Проблемы, решаемые за полиномиальное время, составляют класс P. Класс NP образуют все проблемы, которые могут быть решены недетерминированными алгоритмами с полиномиальной временной сложностью.

Поскольку детерминированные алгоритмы одновременно являются и недетерминированными, то класс проблем P целиком содержится в классе проблем NP. Справедливость обратного включения до сих пор не установлена.

Проблема П полиномиально сводится к проблеме Г, если существует ДМТ-программа с полиномиальной временной сложностью, которая преобразовывает коды индивидуальных задач проблемы П в коды индивидуальных задач проблемы Г

(т. е. такая ДМТ-программа, которая один код, записанный на ленте, в результате выполнения конечного числа тактов, затирает и на его месте пишет другой код) так, что код индивидуальной задачи с ответом да переходит в код индивидуальной задачи с ответом да, а код индивидуальной задачи с ответом нет переходит в код индивидуальной задачи с ответом нет.

Если существует детерминированный алгоритм *А* с полиномиальной временной сложностью, решающий проблему Г и проблема П полиномиально сводится к проблеме Г, то также существует детерминированный алгоритм с полиномиальной временной сложностью, решающий проблему П. Детерминированный алгоритм с полиномиальной временной сложностью, решающий проблему П, сперва преобразовывает индивидуальные задачи проблемы П в индивидуальные задачи проблемы Г, которые решает с помощью детерминированного алгоритма *А*.

Проблема из класса NP, к которой полиномиально сводятся все проблемы из класса NP, называется NP-полной. NP-полные проблемы являются самыми сложными проблемами в классе NP. Наличие детерминированного алгоритма с полиномиальной временной сложностью для решения любой NP-полной проблемы гарантирует существование детерминированных алгоритмов с полиномиальными временными сложностями для всех проблем из класса NP.