# Машина Тьюринга

Выполнила: студентка группы 4252 Савельева Н.К.

# Машины Тьюринга

Машина Тьюринга - это очень простое вычислительное устройство. Она состоит из ленты бесконечной длины, разделенной на ячейки, и головки, которая перемещается вдоль ленты и способна читать и записывать символы. Также у машины Тьюринга есть такая характеристика, как состояние, которое может выражаться целым числом от нуля до некоторой максимальной величины. В зависимости от состояния машина Тьюринга может выполнить одно из трех действий: записать символ в ячейку, передвинуться на одну ячейку вправо или влево и установить внутреннее состояние.

Устройство машины Тьюринга чрезвычайно просто, однако на ней можно выполнить практически любую программу. Для выполнения всех этих действий предусмотрена специальная таблица правил, в которой прописано, что нужно делать при различных комбинациях текущих состояний и символов, прочитанных с ленты.

Алан Тьюринг (Turing) в 1936 году опубликовал в трудах Лондонского математического общества статью "О вычислимых числах в приложении к проблеме разрешения", которая наравне с работами Поста и Черча лежит в основе современной теории алгоритмов.

В 1947 г. расширил определение, описав "универсальную машину Тьюринга". Позже для решения определенных классов задач была введена ее разновидность, которая позволяла выполнять не одну задачу, а несколько.

Машина Тьюринга является расширением модели конечного автомата, расширением, включающим потенциально бесконечную память с возможностью перехода (движения) от обозреваемой в данный момент ячейки к ее левому или правому соседу.

# Описание машины Тьюринга

Заданы:

- Конечное множество состояний - Q, в которых может находиться машина Тьюринга;

- Конечное множество символов ленты - Г;

- Функция д (функция переходов или программа), которая задается отображением пары из декартова произведения Q x Г (машина находится в состоянии qi и обозревает символ i) в тройку декартова произведения Q x Г x {L,R} (машина переходит в состояние qi, заменяет символ i на символ j и передвигается влево или вправо на один символ ленты) - Q x Г-->Q x Г x {L,R} один символ из Г-->e (пустой);

- Подмножество У є Г - -> определяется как подмножество входных символов ленты, причем e є (Г - У);

- Одно из состояний - q0 є Q является начальным состоянием машины.

Решаемая проблема задается путем записи конечного количества символов из множества У є Г - Si є У на ленту: eS1S2S3S4... ... ... Sne после чего машина переводится в начальное состояние и головка устанавливается у самого левого непустого символа (q0,w) -, после чего в соответствии с указанной функцией переходов (qi,Si) - ->(qj,Sk, L или R) машина начинает заменять обозреваемые символы, передвигать головку вправо или влево и переходить в другие состояния, предписанные функций переходов.

Остановка машины происходит в том случае, если для пары (qi,Si) функция перехода не определена.

# Свойства машины Тьюринга как алгоритма

- Дискретность. Машина Тьюринга может перейти к (к + 1) - му шагу только после выполнения каждого шага, т.к именно каждый шаг определяет, каким будет (к + 1) - й шаг.
- Понятность. На каждом шаге в ячейку пишется символ из алфавита, автомат делает одно движение (Л, П, Н), и машина Тьюринга переходит в одно из описанных состояний.
- Детерминированность. В каждой клетке таблицы машины Тьюринга записан лишь один вариант действия. На каждом шаге результат определен однозначно, следовательно, последовательность шагов решения задачи определена однозначно, т.е. если машине Тьюринга на вход подают одно и то же входное слово, то выходное слово каждый раз будет одним и тем же.

- Результативность. Содержательно результаты каждого шага и всей последовательности шагов определены однозначно, следовательно, правильно написанная машина Тьюринга за конечное число шагов перейдет в состояние q0, т. е. за конечное число шагов будет получен ответ на вопрос задачи.
- Массовость. Каждая машина Тьюринга определена над всеми допустимыми словами из алфавита, в этом и состоит свойство массовости. Каждая машина Тьюринга предназначена для решения одного класса задач, т.е. для каждой задачи пишется своя (новая) машина Тьюринга.

# Сложность алгоритмов

Вычислительная сложность алгоритма часто измеряется двумя параметрами:

T (временная сложность) и S (пространственная сложность, или требования к памяти). И T, и S обычно представляются в виде функций от n, где n - это размер входных данных.

(Существую и другие способы измерения сложности: количество случайных бит, ширина канала связи, объем данных и т.п.)Обычно вычислительная сложность алгоритма выражается с помощью нотации "O большого", т. е описывается порядком величины вычислительной сложности.

Это просто член разложения функции сложности, быстрее всего растущий с ростом n, все члены низшего порядка игнорируются.

Например, если временная сложность данного алгоритма равна 4n2+7n+12, то вычислительная сложность порядка n2, записываемая как O(n2).

Временная сложность измеренная таким образом не зависит от реализации.

Не нужно знать ни точное время выполнения различных инструкций, ни число битов, используемых для представления различных переменных, ни даже скорость процессора.

Один компьютер может быть на 50 процентов быстрее другого, а у третьего шина данных может быть в два раза шире, но сложность алгоритма, оцененная по прядку величины, не изменится. Это не жульничество, при работе с алгоритмами настолько сложными, как описанные в этой книге, всем прочим можно пренебречь (с точностью до постоянного множителя) в сравнении со сложностью по порядку величины.

Эта нотация позволяет увидеть, как объем входных данных влияет на требования к времени и объему памяти.

Например, если T= O(n), то удвоение входных данных удвоит и время выполнения алгоритма. Если T=O(2n), то добавление одного бита к входным данным удвоит время выполнения алгоритма.

# Классификация алгоритмов

Обычно алгоритмы классифицируются в соответствии с их временной или пространственной сложностью.

Алгоритм называют постоянным, если его сложность не зависит от n: 0(1). Алгоритм является линейным, если его временная сложность O(n).

Алгоритмы могут быть квадратичными, кубическими и т.д. Все эти алгоритмы - полиномиальны, их сложность - O(m), где m - константа. Алгоритмы с полиномиальной временной сложностью называются алгоритмами с полиномиальным временем Алгоритмы, сложность которых равна O(tf(n)), где t - константа, большая, чем 1, а f(n) - некоторая полиномиальная функция от n, называются экспоненциальными. Подмножество экспоненциальных алгоритмов, сложность которых равна O(cf(n)), где где c - константа, а f(n) возрастает быстрее, чем постоянная, но медленнее, чем линейная функция, называется суперполиномиальным.

# Сложность проблемы

Проблемы, которые можно решить с помощью алгоритмов с полиномиальным временем, называются решаемыми, потому что для разумных входных данных обычно могут быть решены за разумное время.

(Точное определение "разумности" зависит от конкретных обстоятельств) Проблемы, которые невозможно решить за полиномиальное время, называются нерешаемыми, потому что вычисление их решений быстро становится невозможным. Нерешаемые проблемы иногда называют трудными.

Проблемы, которые могут быть решены только с помощью суперполиномиальных алгоритмов, вычислительно нерешаемы, даже при относительно малых значениях n.

Задачи можно разбить на классы в соответствии со сложностью их решения. Вот важнейшие из них и предполагаемые соотношения между ними:

P<=NP<=EXPTIME

Находящийся слева класс P включает все задачи, которые можно решить за полиномиальное время. В класс NP входят все задачи, которые можно решить за полиномиальное время только на недетерминированной машине Тьюринга (это вариант обычной машины Тьюринга, которая может делать предположения). Такая машина предполагает решение задачи - либо “удачно угадывая”, либо перебирая все предположения параллельно - и проверяет свое предположение за полиномиальное время.

Класс NP включает в себя класс P, поскольку любую задачу, решаемую за полиномиальное время на детерминированной (обычной) машине Тьюринга, можно решить и на недетерминированной за полиномиальное время, просто этап предположения опускается.

Если все задачи класса NP решаются за полиномиальное время и на детерминированной машине, то P=NP. Тем не менее, никем не доказано, что P<>NP (или P=NP). Однако, большинство специалистов, занимающихся теорией сложности, убеждены, что это классы неравны.

Можно доказать, что некоторые NP-задачи настолько же трудны, что и любая задача этого класса. Такие задачи называются NP-полными. То есть, если такая задача решается за полиномиальное время, то P=NP.

Полнота означает полный перебор, причем сложность этого перебора будет экспоненциальной или факториальной. Но следует понимать, что не всякий полный перебор имеет такую сложность. Например, если решать задачи из предыдущего выпуска полным перебором, то сложность полученных алгоритмов будет полиномиальной - O(n2) для задачи про подпоследовательности и O(n6) для задачи про подматрицы.

Наконец, существует класс задач EXPTIME. Эти задачи решаются за экспоненциальное время. В настоящее время можно доказать, что EXPTIME-полные задачи невозможно решить за детерминированное полиномиальное время. Кроме того, доказано, что P<>EXPTIME.

# Машины Тьюринга и алгоритмически неразрешимые проблемы

Первой фундаментальной теоретической работой, связанной с доказательством алгоритмической неразрешимости, была работа Курта Гёделя - его известная теорема о неполноте символических логик.

Это была строго формулированная математическая проблема, для которой не существует решающего ее алгоритма.

Усилиями различных исследователей список алгоритмически неразрешимых проблем был значительно расширен.

Сегодня принято при доказательстве алгоритмической неразрешимости некоторой задачи сводить ее к ставшей классической задаче - "задаче останова".

Имеет место быть следующая теорема: не существует алгоритма (машины Тьюринга), позволяющего по описанию произвольного алгоритма и его исходных данных (и алгоритм и данные заданы символами на ленте машины Тьюринга) определить, останавливается ли этот алгоритм на этих данных или работает бесконечно.

Таким образом, фундаментально алгоритмическая неразрешимость связана с бесконечностью выполняемых алгоритмом действий, т.е. невозможностью предсказать, что для любых исходных данных решение будет получено за конечное количество шагов.

Тем не менее, можно попытаться сформулировать причины, ведущие к алгоритмической неразрешимости, эти причины достаточно условны, так как все они сводимы к проблеме останова, однако такой подход позволяет более глубоко понять природу алгоритмической неразрешимости:

а) Отсутствие общего метода решения задачи

Проблема 1: Распределение девяток в записи числа;

Определим функцию f(n) = i, где n - количество девяток подряд в десятичной записи числа, а i - номер самой левой девятки из n девяток подряд: =3,141592… f(1) = 5.

Задача состоит в вычислении функции f(n) для произвольно заданного n.

Поскольку число является иррациональным и трансцендентным, то мы не знаем никакой информации о распределении девяток (равно как и любых других цифр) в десятичной записи числа. Вычисление f(n) связано с вычислением последующих цифр в разложении, до тех пор, пока мы не обнаружим n девяток подряд, однако у нас нет общего метода вычисления f(n), поэтому для некоторых n вычисления могут продолжаться бесконечно - мы даже не знаем в принципе (по природе числа) существует ли решение для всех n.

Проблема 2: Вычисление совершенных чисел;

Совершенные числа - это числа, которые равны сумме своих делителей, например: 28 = 1+2+4+7+14.

Определим функцию S(n) = n-ое по счёту совершенное число и поставим задачу вычисления S(n) по произвольно заданному n. Нет общего метода вычисления совершенных чисел, мы даже не знаем, множество совершенных чисел конечно или счетно, поэтому наш алгоритм должен перебирать все числа подряд, проверяя их на совершенность. Отсутствие общего метода решения не позволяет ответить на вопрос о останове алгоритма.

Проблема 3: Десятая проблема Гильберта;

Пусть задан многочлен n-ой степени с целыми коэффициентами - P, существует ли алгоритм, который определяет, имеет ли уравнение P=0 решение в целых числах?

Ю.В. Матиясевич показал, что такого алгоритма не существует, т.е. отсутствует общий метод определения целых корней уравнения P=0 по его целочисленным коэффициентам.

б) Информационная неопределенность задачи

Проблема 4: Позиционирование машины Поста на последний помеченный ящик;

Пусть на ленте машины Поста заданы наборы помеченных ящиков (кортежи) произвольной длины с произвольными расстояниями между кортежами и головка находится у самого левого помеченного ящика. Задача состоит установке головки на самый правый помеченный ящик последнего кортежа.

Попытка построения алгоритма, решающего эту задачу приводит к необходимости ответа на вопрос - когда после обнаружения конца кортежа мы сдвинулись вправо по пустым ящикам на M позиций и не обнаружили начало следующего кортежа - больше на ленте кортежей нет или они есть где-то правее? Информационная неопределенность задачи состоит в отсутствии информации либо о количестве кортежей на ленте, либо о максимальном расстоянии между кортежами - при наличии такой информации (при разрешении информационной неопределенности) задача становится алгоритмически разрешимой.

в) Логическая неразрешимость (в смысле теоремы Гёделя о неполноте)

Проблема 5: Проблема "останова" (см. теорема);

Проблема 6: Проблема эквивалентности алгоритмов;

По двум произвольным заданным алгоритмам (например, по двум машинам Тьюринга) определить, будут ли они выдавать одинаковые выходные результаты на любых исходных данных.

Проблема 7: Проблема тотальности;

По произвольному заданному алгоритму определить, будет ли он останавливаться на всех возможных наборах исходных данных.