Уточнение понятия алгоритм и его формализации

- В широком смысле слова алгоритм– это текст, который в определенных обстоятельствах может привести к однозначному развитию событий– процессу выполнения алгоритма.

- Каждый алгоритм служит для решения некоторого класса задач.
- Задачи должны быть записаны на некотором языке.
- Результат применения алгоритма– решение задачи– также должен быть записан на вполне определенном языке.
- Таким образом, в процессе выполнения алгоритма текст задачи преобразуется в текст ее решения.

Свойства алгоритма

- ***Дискретность.*** Алгоритм – это процесс последовательного построения выражений таким образом, что в начальный момент задается исходное конечное выражение, а в каждый следующий момент выражение получается по определенному закону выражения, имевшегося в предыдущий момент времени.

- ***Детерминированность.*** Выражение, получаемое в какой-то не начальный момент, однозначно определяется выражением, полученным в предшествующие моменты времени.

- ***Элементарность шагов алгоритмов.*** Закон получения последующего выражения из предшествующего должен быть простым. (**Для исполнителя!**).

- ***Массовость алгоритма.*** Начальное выражение может выбираться из некоторого потенциально бесконечного множества. Иначе говоря, алгоритм должен обеспечивать решение некоторому множеству (классу) задач с различными параметрами (коэффициентами).

- ***Результативность алгоритма.*** Последовательный процесс построения выражений языка должен быть конечным и давать результат, то есть решение задачи.

- Основная задача теории алгоритмов – это решение ***проблемы алгоритмической разрешимости,*** а не поиск правила (способа/метода) ее решения.

- Теория алгоритмов дает ответ на вопрос «Данная задача имеет решение?», и не отвечает на вопрос «Как решается данная задача?»

- В рамках такого подхода к определению понятия алгоритма можно определить ***три основных направления:***
  - Первое направление связано с уточнением понятия эффективно вычисляемой функции. В результате был выделен класс так называемых ***рекурсивных функций***.

– Второе направление связано с машинной математикой. Здесь сущность понятия алгоритма раскрывается путем рассмотрения процессов, осуществляемых в некой механистической абстрактной конструкции - машине.

Впервые это было сделано Тьюрингом, который предложил общую и вместе с тем самую простую концепцию вычислительной машины. Ее описание было дано Тьюрингом в 1937 г. А это направление в теории алгоритмов получило название - ***машина Тьюринга.***

– Третье направление связано с понятием нормальных алгоритмов, введенным и разработанным российским математиком А. А. Марковым.

Это направление получило название ***нормальные алгоритмы Маркова.***

- Третье направление связано с понятием нормальных алгоритмов, введенным и разработанным российским математиком А. А. Марковым. Это направление получило название ***нормальные алгоритмы Маркова.***

Частично рекурсивные функции

- Таким образом процесс алгоритмического решения задачи должен быть дискретным. Он распадается на элементарные шаги и представляет собой цепочку преобразований вида $T_0 \to T_1 \to \ldots \to T_k,$

- где T_0 – текст, представляющий задачу, а T_k – текст, дающий ее решение. Преобразование текста на каждом шаге производится по предписаниям, которые берутся из конечного и фиксированного раз и навсегда списка.

- Поскольку, тексты (слова над конечным алфавитом) могут быть занумерованы, то цепочка текстов «задача– решение» превратится в числовую цепочку их номеров:

$$n_0 \to n_1 \to \ldots \to n_k.$$

- Такой цепочке можно поставить в соответствие числовую функци $y=f(x)$, реализующую отображени $n_0 \to n_k$,

- определенную на множестве номеров задач $X \subset N$ и принимающую значения в N.

- Алгоритм $y=f(x)$, вает не только саму функцию но и способ ее пошагового вычисления.

- Далее, если не сделано специальных оговорок, мы будем предполагать, что рассматриваемые функции $y=f(x_1,x_2,\ldots,x_n)$ являются числовыми, их значения и аргументы принадлежат множеству натуральных чисел $N=\{0,1,2,\ldots\}$.

- Если функция $y=f(x_1,x_2,\ldots,x_n)$ определена на собственном подмножестве N^n, множества

  то будем называть ее частично рекурсивной.

Определим простейшие функции и элементарные операции над функциями.

Простейшие функции:

1) $S(x)=x+1$ (*функция следования*);

2) $O(x)=0$ (*тождественный ноль*);

3) $Pr[n;i](x_1,x_2,\ldots,x_n)=x_i$, $n\geq 1$ (*проекции*).

Заметим, что проекции $Pr[n;i]$ составляют бесконечное семейство функций.

Элементарные операции над частичными функциями.

- *1. Суперпозиция(или композиция).*

  Пусть даны частичная функция

  $$g(x_1,x_2,\ldots,x_m)$$

  и частичные функции

  $$f_1(x_1,x_2,\ldots,x_n),\ \ldots,\ f_m(x_1,x_2,\ldots,x_n).$$

Суперпозицией функций g и $f_1,\ldots,f_m$ называется частичная функция

$$h(x_1,x_2,\ldots,x_n)=g(f_1(x_1,x_2,\ldots,x_n),\ldots,f_m(x_1,x_2,\ldots,x_n)),$$

которая задается на наборе $(x_1,x_2,\ldots,x_n)$ указанной формулой, если определены все значения $y_1=f_1(x_1,x_2,\ldots,x_n)$, …, $y_m=f_m(x_1,x_2,\ldots,x_n)$ и значение $g(y_1,y_2,\ldots,y_m)$.

- В противном случае функция $h(x_1,x_2,\ldots,x_n)$ считается неопределенной. Для функции h, полученной суперпозицией функций g и $f_1,\ldots,f_m$,
- $h=Cn[g;f_1,\ldots,f_m]$. овать обозначение

Примеры.

$Cn[S;S](x)=S(S(x))=S(x+1)=x+2;$

$Cn[S;Cn[S;S]](x)=x+3;$

$Cn[O;Pr[n;1]](x_1,x_2,\ldots,x_n)=O(Pr[n;1](x_1,x_2,\ldots,x_n))=O(x_i)=0.$

2.Рекурсия

- Начнем с частных случаев.
- Пусть заданы функци $g(y,z)$ и число a. Уравнения:

$$h(0)=a;\ h(y+1)=g(y,h(y))$$

- однозначно определяют функци $h(y)$.

- Последовательно вычисляя, находим:

$$h(1)=g(0,a);\ h(2)=g(1,h(1));\ \ldots$$

Пример. При $a=1$, $g(y,z)=y\cdot z$ уравнения

$$h(0)=1;\ h(y+1)=(y+1)h(y)$$

определяют функцию

$$h(y)=y!=1\cdot 2\cdot 3\cdot\ldots\cdot y.\square$$

Пусть даны функции $f(x)$ и $g(x,y,z)$. Говорят, что функция $h(x,y)$ получается из функций f и g (*примитивной*) *рекурсией*, если функция h удовлетворяет уравнениям:

$$h(x,0)=f(x);\ h(x,y+1)=g(x,y,h(x,y)).$$

Ясно, что функция h определена однозначно. Имеем:

$$h(x,1)=g(x,0,h(x,0))=g(x,0,f(x));$$

$$h(x,2)=g(x,1,h(x,1)); \ldots .$$

Для функции $h(x,y)$, полученной рекурсией из функций f и g, будем использовать обозначение $h=Rc[f;g]$.

Примеры. Пусть $f(x)=x$ (то есть $f=Pr[1;1]$) и $g(x,y,z)=z+1$ (то есть $g=Cn[S;Pr[3,3]]$). Тогда функция $sum=Rc[f;g]$ определяется уравнениями

$$sum(x,0)=x; \quad sum(x,y+1) = g(x,y,sum(x,y)) = sum(x,y)+1.$$

Имеем:

$$sum(x,1)=sum(x,0)+1=x+1;$$

$$sum(x,2)=sum(x,1)+1=(x+1)+1=x+2, \ldots$$

и, вообще, $sum(x,y)=x+y$.

Пусть $g(x,y,z)=z+x$ (то есть $g=Cn[\text{sum};Pr[3;3],Pr[3;1]]$). Определим функцию $\text{prod}(x,y)$ как $\text{prod}=Rc[O;g]$. Тогда h удовлетворяет уравнениям

$$\text{prod}(x,0)=0;\ \text{prod}(x,y+1)=h(x,y)+x.$$

Получаем:

$$\text{prod}(x,1)=h(x,0)+x=0+x=x;$$

$$\text{prod}(x,2) = h(x,1)+x = x+x = x\cdot 2;\ \ldots,$$

и, вообще, $\text{prod}(x,y)=xy$.□

Вообще говоря, в определении по рекурсии $h=Rc[f;g]$ можно считать, что x представляет собой набор аргументов $(x_1,x_2,\ldots,x_n)$. Тогда рекурсия по функциям $f(x_1,x_2,\ldots,x_n)$, $g(x_1,x_2,\ldots,x_n y,z)$ определяет функцию $h(x_1,x_2,\ldots,x_n,y)$.

Функции, которые могут быть получены из простейших функций операциями суперпозиции и рекурсии, называются *примитивно рекурсивными*.

3. Минимизация.

Пусть $f(x_1,x_2,\ldots,x_n,y)$ всюду определенная функция. Определим функцию $h(x_1,x_2,\ldots,x_n,z)$ условием:

если имеются значения y такие, что $f(x_1,x_2,\ldots,x_n,y)=z$, то $h(x_1,x_2,\ldots,x_n,z)$ есть наименьшее из таких значений; в противном случае $h(x_1,x_2,\ldots,x_n,z)$ не определено.

Для так определенной функции h будем писать $h=Mn[f]$.

Если функция $f(x_1,x_2,\ldots,x_n,y)$ частичная, то $h=Mn[f]$ определяется следующими условиями:

$h(x_1,x_2,\ldots,x_n,z)=y$, если $f(x_1,x_2,\ldots,x_n,y)=z$, причем $f(x_1,x_2,\ldots,x_n,t)$ определено и отлично от z для всех $t<y$;

в противном случае $h(x_1,x_2,\ldots,x_n,z)$ не определено.

Пример. Рассмотрим функцию $h=Mn[\text{sum}]$. Имеем

$$h(x,z)=\min\{y \mid x+y=z\}.$$

Если $x\le z$, то $h(x,z)=z-x$, в противном случае $h(x,z)$ не определено. Обозначим эту функцию через dif.□

- Частичные функции, которые могут быть получены из простейших с помощью конечного числа операций суперпозиции, рекурсии и минимизации, называются рекурсивными (или частично рекурсивными).

- Всюду определенные частично рекурсивные функции называются общерекурсивными.

- Запись частично рекурсивной функции с помощью простейших функций и операций будем называть рекурсивной схемой. Рекурсивная схема фактически задает алгоритм вычисления функции.

- По рекурсивной схеме функции f может быть построено ее рекурсивное описание: конечная последовательность частичных функций

$$f_0, f_1, \ldots, f_n$$

такая, что $f_n = f$, каждая функция в этой последовательности либо является простейшей, либо получается применением одной из элементарных операций к некоторым из предшествующих ей функций.

- Одна и та же функция может быть определена с помощью разных рекурсивных схем. Это согласуется с представлением о том, что одну и ту же функцию можно вычислять по-разному.

Пример. Имеем следующую рекурсивную схему для функции dif из предыдущего примера:

$$\text{dif}=Mn[Rc[Pr[1;1]; Cn[S;Pr[3,3]]]].$$

Из этой формулы без труда можно получить рекурсивное описание функции dif:

$$Pr[1;1],\ Pr[3,3],\ S,\ Cn[S;Pr[3,3]],\ Rc[Pr[1;1]],$$

$$Mn[Rc[Pr[1;1]; Cn[S;Pr[3,3]]]]=\text{dif}.$$

- Рекурсивная схема представляет собой слово над счетным алфавитом, содержащим в качестве символов натуральные числа, обозначения для простейших функций, элементарных операций, скобки, запятую и точку с запятой. Следовательно, множество рекурсивных схем счетно. Вместе с ним счетно и множество частично рекурсивных функций.

Вычислимость и разрешимость

- Отметим, что традиционно считающиеся вычислимыми функции имеют рекурсивные описания и, значит, частично рекурсивны. Обычно используемые вычислительные схемы также реализуются с помощью простейших функций и элементарных операций. Все это и ряд других соображений приводит к следующей формулировке.

- **Тезис Черча.** Числовая функция тогда и только тогда алгоритмически вычислима, когда она частично рекурсивна.
- Построим пример невычислимой функции. Начнем с некоторых общих определений и замечаний.

- Подмножество множества натуральных чисел $M \subset N$ называется разрешимым, если его характеристическая функция

$$\chi_M(x)=1 \text{ при } x \in M,$$

$$\chi_M(x)=0 \text{ при } x \notin M$$

рекурсивна.

- Содержательно разрешимость множества ***M*** означает, что существует алгоритм, позволяющий по любому числу ***x*** определить за конечное число шагов, принадлежит это число множеству ***M*** или нет.

- Подмножество множества натуральных чисел $M \subset N$ называется перечислимым, если оно является областью значений некоторой общерекурсивной функции f.
- Перечислимость множества M означает, что его элементы могут быть последовательно выписаны (возможно с повторениям) с помощью некоторой эффективной процедуры.

- **Утверждение:** Всякое непустое разрешимое множество ***M*** является перечислимым.
- Доказательство. Определим перечисляющую функцию ***f***. Пусть ***m***– произвольный элемент множества ***M***. Определяем по рекурсии:

$$f(0)=m; \qquad f(x+1)=\chi_M(x)(x+1)+(1-\chi_M(x+1))f(x),$$

то есть $f(x+1)=x+1$, если $x+1\in M$, и $f(x+1)=f(x)$, если $x+1\notin M$.

- Обратное, вообще говоря, неверно. Не всякое перечислимое множество является разрешимым. Перечислимое множество разрешимо лишь в том случае, когда перечислимо также и его дополнение.

- Поскольку, что частично рекурсивные функции можно эффективно перенумеровать, используя их рекурсивные описания, то некоторые номера соответствуют общерекурсивным функциям. Обозначим множество таких номеров через ***M*** и покажем, что множество ***M*** неперечислимо.

- **Теорема.** Множество номеров общерекурсивных функций не перечислимо.
- Доказательство. Предположим противное. Пусть $\varphi(x)$ – общерекурсивная функция, множеством значений которой является M. Тогда последовательность $\varphi(0),\ \varphi(1),\ \varphi(2),\ldots$ содержит номера всех общерекурсивных функций, и только их.
- Определим функцию $g(x)$ формулой

$$g(x)=f_{\varphi(x)}(x)+1.$$

- Это определение дает алгоритм вычисления значений функции $g(x)$. В соответствии с тезисом Черча, функция $g(x)$ частично рекурсивна, и, значит, общерекурсивна, поскольку функция $g(x)$ определена для любого x. Значит, функция $g(x)$ должна получить свой номер при перечислении с помощью $\varphi(x)$.

- Вообще неперечислимые и неразрешимые семейства функций– это не «экзотика», а, скорее, норма.
- Приведем без доказательства следующую теорему.
- ***Терема (Райс).*** Никакое нетривиальное семейство вычислимых функций не является алгоритмически разрешимым.

- Иными словами, если *C*– некоторое семейство вычислимых функций такое, что есть функции, входящие в это семейство, а есть и не входящие в него, то множество номеров функций из *C* неразрешимо. Не существует алгоритма, который бы позволял по номеру функции сказать, входит она в *C* или нет.

- Так, по номеру функции нельзя узнать, является ли она монотонной, периодической и т.п. Заметим, что, нумеруя частично рекурсивные функции, мы на самом деле нумеровали их рекурсивные описания, то есть вычисляющие их алгоритмы.

- Теорема Райса утверждает, что по номеру алгоритма нельзя узнать, периодична ли, например, функция, вычисляемая в соответствии с этим алгоритмом.

Машина Тьюринга

- Если для решения некоторой массовой проблемы известен алгоритм, то для его реализации необходимо лишь четкое выполнение предписаний этого алгоритма. Автоматизм, необходимый при реализации алгоритма, приводит к мысли о передаче функции человека, реализующий алгоритм, машине.

- Идею такой машины предложил в 1937 году английский математик А. Тьюринг.

- Машина Тьюринга включает в себя:
- ***Внешний алфавит*** - конечное множество символов $A = \{a_0, a_1, a_2, \ldots, a_n\}$. В этом алфавите в виде слова кодируется та информация, которая подается в машину. Машина перерабатывает информацию, поданную в виде слова, в новое слово. Обычно символ ***Внешний алфавит*** - конечное множество символов обозначает a_0 пробел.

- ***Внутренний алфавит*** - конечное множество символов $Q = \{q_0, q_1, q_2, \ldots, q_m\}$. Для любой машины число состояний фиксировано. Два состояния имеют собое назначение q_1 - начальное состояние q_0 машины, - заключительное состояние (стоп-состояние).

- ***Операторы перемещения*** T={Л, П, Н}. Л, П, Н – это символы сдвига «влево», «вправо» и «на месте».
- ***Бесконечная лента*** Бесконечная лента характеризует память машины. Она разбита на клеточки. В каждую клеточку может быть записан только один символ из внешнего алфавита.

- ***Управляющая головка.*** Управляющая головка (УГ) передвигается вдоль ленты и может останавливаться напротив какой-либо клетки, т. е. считывать символ

- ***Логическое устройство.*** В зависимости от текущего внутреннего состояния, и считанного с ленты символа, переходит в новое внутренне состояние, и «премещает» управляющую головку.

- ***Программа машины Тьюринга (Р) -*** совокупность всех команд, Программа представляется в виде таблицы и называется ***Тьюринговой функциональной схемой.***
- ***Например:***

	a_0	a_1	a_2
q_1	$a_0 П q_1$	$a_1 П q_1$	$a_2 Л q_2$
q_2	$a_1 П q_2$	$a_2 Н q_0$	$a_0 Н q_0$

- Таким образом, машина Тьюринга может быть представлена в виде четверки:

$$MT =< A, Q, T, P >$$

- Информация, хранящаяся на ленте, является набором символов из внешнего алфавита. Начальное состояние управляющей головки характеризуется символом внутреннего алфавита q_1.

- Работа машины складывается из тактов. В течение любого такта машина Тьюринга осуществляет следующие действия: машина Тьюринга находится во внутреннем состоянии q_i, считывает входной символ a_j и по таблице работы совершает операцию сдвига t_l, переходя в состояние q_k, при этом входное слово a_j заменяется на a_m:

$$a_j q_i \rightarrow a_m t_l q_k$$

- Если в результате операции машина перейдет в состояни q_0 , то работа машины останавливается. Если q_0 состояние недостижимо, то значит по данному входному слову машина Тьюринга не достигает конечного состояния и алгоритма для данного входного слова не существует.

- ПРИМЕР
- Построим машину Тьюринга, которая будет стирать последнюю единицу в последовательности единиц.

- Внешний алфавит - $A = \{a_0, 1\}$. Внутренний алфавит - $Q = \{q_0, q_1, q_2\}$, при этом состояние q_1 сохраняется до тех пор, пока не будет найден конец последовательности единиц, состояние q_2 - стирание последней единицы.

- При этом следует заметить, что ситуация $a_0 q_2$ в работе машины Тьюринга невозможна, поэтому соответствующая клеточка доопределена произвольно, например $a_0 \Pi q_2$.

- Начальное состояни q_1 , головка установлена на первой единице последовательности единиц. Рабочая программа машины Тьюринга имеет вид:

	1	a_0
q_1	$1\ П\ q_1$	$a_0\ Л\ q_2$
q_2	$a_0\ Н\ q_0$	$a_0\ П\ q_2$

- Проверим работоспособность машины Тьюринга:

1. $\overline{a_0}\,\overline{1}\,1\,a_0$, q_1
2. $a_0\,11\,a_0$, q_1
3. $a_0\,11\,a_0$, q_1
4. $a_0\,11\,a_0$, q_2
5. $a_0\,10\,a_0$, q_0

- ***Тезис А. Черча.*** Если функция ***выполнима***, то она всегда может быть представлена в виде машины Тьюринга.

Нормальные алгоритмы Маркова

- Нормальный алгоритм Маркова представляет собой систему подстановок

$$P \to (\bullet)\, Q,$$

где P, Q - слова из символов входного алфавита A

$\to (\bullet)$ - оператор подстановки, при этом :

$\to$ -простая подстановка

$\to \bullet$ - заключительная подстановка.

- ***Слово z*** считается ***включенным в слово y***, если ***y*** может быть представлено как:

$$x_1 z x_2,$$

где x_1, x_2 - любые символы входного алфавита. A

Работа нормального алгоритма Маркова:

- Исходное слово просматривается слева направо с целью выявления вхождения первого правила подстановки. Как только находится первое вхождение первого правила подстановки, оно заменяется по этому правилу и исходное слово снова просматривается с первого символа по первому правилу подстановки.

- После того, как первое правило больше не встречается в данном слове, аналогично применяется второе правило подстановки.
- Работа алгоритма заканчивается тогда, когда ни одна из подстановок не применима, либо использована заключительная подстановка.

- ПРИМЕР
- Построить нормальный алгоритм Маркова, стирающий последовательность единиц.
- Нормальный алгоритм Маркова для данной задачи представляет собой две подстановки :

1. $11 \to 1$
2. $1a_0 \to \bullet\, a_0 a_0$

- Первая подстановка стирает все единицы до последней. Вторая (заключительная) подстановка заменяет последнюю единицу пробелом .

- ***Тезис А. Черча.*** Если функция ***выполнима***, то она может быть представлена в виде нормального алгоритма Маркова.
-
- ***Заключительный тезис А. Черча.*** Если функция ***выполнима***, то она может быть представлена в виде либо общерекурсивной функции, либо машины Тьюринга, либо в виде нормального алгоритма Маркова.

- Один из видов чертежей– графы, которые, сохранив присущую чертежам наглядность, допускают точное теоретико-множественное описание и тем самым становятся объектом математического исследования.