ВВЕДЕНИЕ В ТЕОРИЮ АЛГОРИТМОВ

Компьютерная программа делает то, что вы приказали ей сделать, а не то, что вы хотели, чтобы она сделала.

Д. Грир

Основы программирования

Содержание

- Неформальное определение алгоритма
- Формальное определение алгоритма
  - Машина Тьюринга
  - Нормальные алгорифмы Маркова
- Проблема алгоритмической неразрешимости
- Методы разработки алгоритмов

Алгоритм

- *Алгоритм* –процесс последовательного построения величин, идущий в дискретном времени таким образом, что в начальный момент задается исходная конечная система величин, а в каждый следующий момент система величин получается по определенному закону из системы величин, имевшихся в предыдущий момент времени.

  [акад. А.И. Мальцев,
  Алгоритмы и рекурсивные функции.
  М.: Наука, 1986].

Алгоритм

- *Алгоритм* – точное предписание, задающее вычислительный процесс, который начинается с произвольного исходного данного (входящего в совокупность возможных для данного алгоритма исходных данных) и направлен на получение полностью определяемого этим данным результата.

[Математическая энциклопедия.
Под ред. акад. И.М. Виноградова.
М: Изд-во «Советская энциклопедия», 1977]

Алгоритм

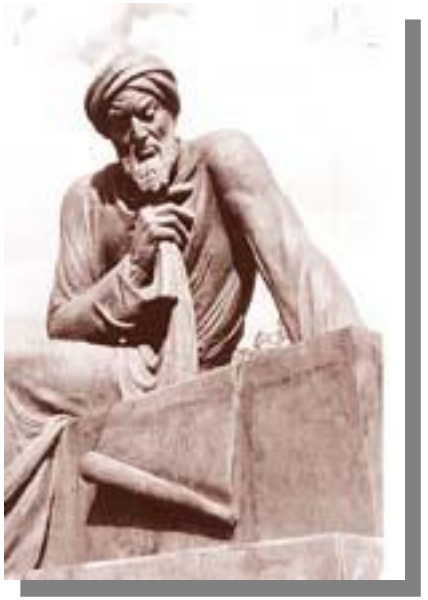
- *Алгоритм* – точное предписание исполнителю совершить определенную последовательность действий для достижения поставленной цели за конечное число шагов .

Дано:
a, *b*, *c* – вещественные
Найти:
корни уравнения $ax^2+bx+c=0$

```
Алгоритм КОРНИ;
вход вещ a, b, c;
выход вещ x1, x2;
перем вещ D;
начало
  D:=b*b-4*a*c;
  если D<0 то
    аварвыход("Корней нет");
  иначе
    x1:=(-b+sqrt(D))/(2*a);
    x2:=(-b-sqrt(D))/(2*a);
  конец если
конец
```

Ал-Хорезми

- *Абу-Джафар Мухаммед ибн Муса ал-Хорезми (783-850 гг.)* – арабский математик.
- Главные труды:
  - *«Китаб ал-Джебр»* – общие правила решения арифметических задач при помощи уравнений
  - *«Об индийском счете»* – правила записи натуральных чисел с помощью арабских цифр и правила действий над ними «столбиком».



Свойства алгоритма

1. *Дискретность*: алгоритм состоит из конечного числа шагов, которые выполняются в дискретном времени.
2. *Элементарность шагов*: объем работы на любом из шагов алгоритма не превышает некоторой константы, зависящей от характеристик исполнителя алгоритма, но не от входных данных и промежуточных результатов алгоритма.
3. *Определенность (детерминированность)*: результаты выполнения каждого шага алгоритма вычисляются однозначно и не зависят от случайных факторов.
4. *Конечность (финитность)*: алгоритм должен приводить к решению задачи за конечное число шагов (либо останавливаться из-за невозможности получить решение).
5. *Массовость*: алгоритм должен быть применим для некоторого класса задач, различающихся лишь исходными данными.

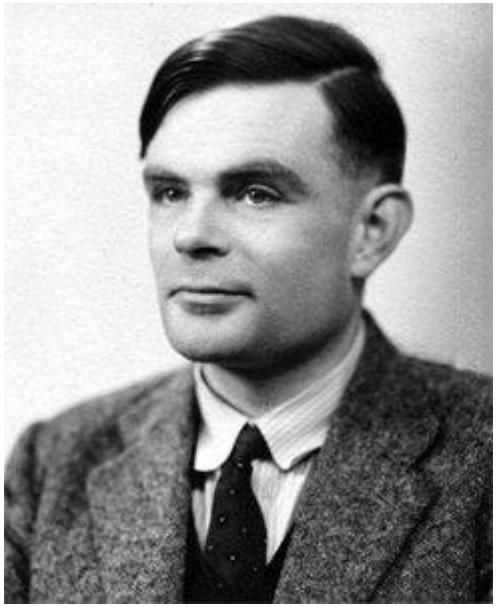
Пример алгоритма


```
Алгоритм МАКСИМУМ_В_МАССИВЕ;
вход вещ мас A[1..N];
выход вещ Max;
перем цел i;
начало
  Max:=A[1];
  i:=2;
  пока i<=N
    если Max<A[i] то
       Max:=A[i];
    конец если
    i:=i+1;
  конец пока
конец
```

1. **Дискретность**
   Очевидна
2. **Элементарность шагов**
   Количество операций ≤6
3. **Определенность**
   Очевидна
4. **Конечность**
   Очевидна
5. **Массовость**
   Алгоритм применим для любого натурального значения **N**

Машина Тьюринга

- *Машина Тьюринга* – абстрактная вычислительная машина (абстрактный исполнитель), предложенная А. Тьюрингом для формализации понятия алгоритма.
- Алан Тьюринг (1912-1954) – английский математик.
  - Расшифровка кода "Энигмы"
  - Тест Тьюринга
  - Премия Тьюринга

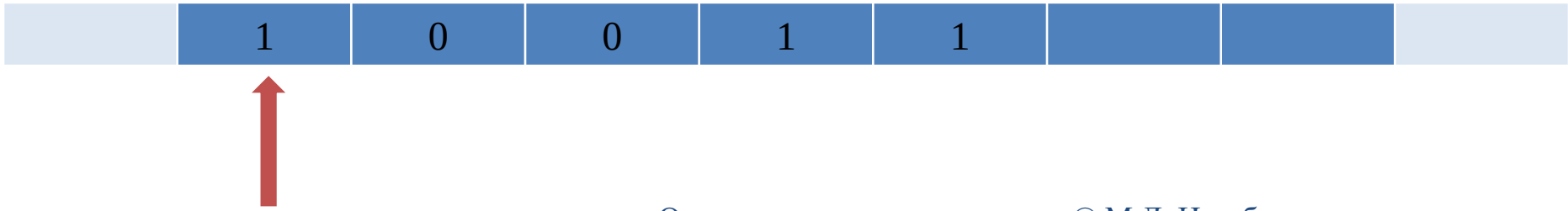


Машина Тьюринга

- *Алфавит входных символов* машины Тьюринга
  - $A = \{a_0, a_1, ..., a_m\}$.
  - a_0 – пустой символ.
- *Алфавит состояний* машины Тьюринга
  - $Q = \{q_0, q_1, ..., q_p\}$.
  - q_0 – завершающее состояние.
  - q_1 – начальное состояние.

Машина Тьюринга

- *Лента* для хранения обрабатываемых данных и результатов
  - Разделена на ячейки, не ограничена в обе стороны.
- *Автомат*, выполняющий обработку данных
  - Головка чтения-записи, управляемая программой.
  - Действия автомата:
    - Передвинуть головку чтения-записи на 1 символ влево (вправо) или оставить ее на месте, прочитав символ в текущей ячейке ленты.
    - Записать новый символ в текущую ячейку (затерев старый символ).
    - Изменить состояние машины Тьюринга.

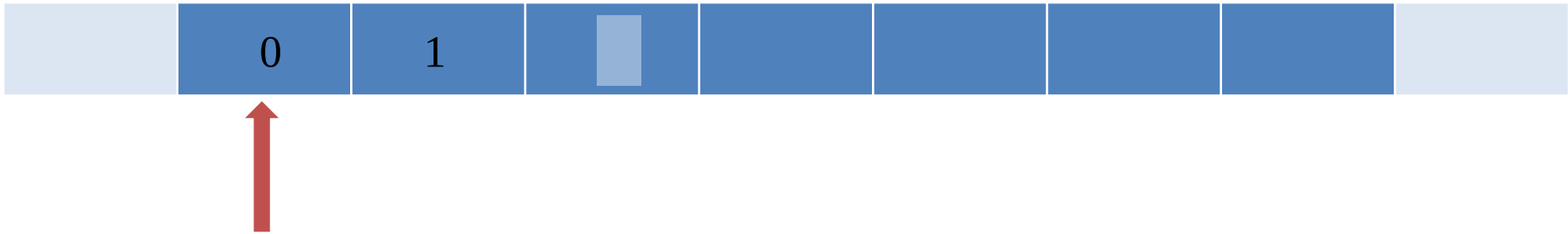


Программа машины Тьюринга

	a_0	a_1	…	a_i	…	a_m
q_0						
q_1						
…						
q_j				$a_k \begin{Bmatrix} Л \\ П \\ Н \end{Bmatrix} q_n$		
…						
q_p						

Программа машины Тьюринга

	a_0	0	1
q_0	a_0 Н q_1	1 П q_0	0 П q_0
q_1			



Пример программы для МТ

- На ленте записано целое десятичное число. Построить МТ, которая прибавляет единицу к заданному числу.
  - В начальном состоянии МТ обозревает правую цифру числа.

	a_0	0	1	2	3	4	...	7	8	9
q_1	1 Н q_0	1 Н q_0	2 Н q_0	3 Н q_0	4 Н q_0	5 Н q_0	...	8 Н q_0	9 Н q_0	0 Л q_1

Пример программы для МТ

- На ленте – последовательность символов "+". Построить МТ, которая заменяет каждый второй символ "+" на "–".

- Состояния
  - *q1* – поиск правого конца числа
  - *q2* – пропуск знака “+”, при достижении конца последовательности – останов.
  - *q3* – замена знака “+” на знак “–”

	a_0	+	–
q_1	a_0 Л q_2	+ П q_1	
q_2	a_0 Н q_0	+ Л q_3	
q_3	a_0 Н q_0	– Л q_2	

Пример программы для МТ

- На ленте – натуральное десятичное число n >1. Разработать МТ, которая уменьшает n на 1.
  - В начальном состоянии МТ обозревает правую цифру числа.

	a_0	0	1	2	...	8	9
q_1		9 Л q_1	0 Н q_0	1 Н q_0	...	7 Н q_0	8 Н q_0

Пример программы для МТ

- На ленте – натуральное десятичное число $n > 1$. Разработать МТ, которая уменьшает n на 1, при этом в выходном слове старшая цифра не должна быть 0.
  - Например, если входным словом было "100", то выходным словом должно быть "99", а не "099".
  - В начальном состоянии МТ обозревает правую цифру числа.

	a_0	0	1	2	...	8	9
q_1		9 Л q_1	0 Л q_2	1 Н q_0	...	7 Н q_0	8 Н q_0
q_2	a_0 П q_3	0 Н q_0	1 Н q_0	2 Н q_0	...	8 Н q_0	9 Н q_0
q_3		a_0 Н q_0					

Пример программы для МТ

- На ленте – строка из открывающих и закрывающих скобок. Построить МТ, которая удаляет пары расположенных подряд скобок "( )".
  - Например, дано ") (( ) (( )", надо получить ") . . . ((".
  - В начальном состоянии МТ обозревает крайний левый символ строки.

	a_0	(	)
q_1	a_0 Н q_0	(П q_2	) П q_1
q_2	a_0 Н q_0	(П q_2	) Л q_3
q_3	a_0 Н q_0	a_0 П q_3	a_0 П q_1

Композиция машин Тьюринга

- Последовательное соединение

T1	a_0	$a1_1$	…	$a1_m$
$q1_0$				
$q1_1$				
…				
$q1_p$				

T2	a_0	$a2_1$	…	$a2_n$
$q2_0$				
$q2_1$				
…				
$q2_k$				

Композиция машин Тьюринга

- Последовательное соединение

T1T2	a_0	$a1_1$	…	$a1_m$	$a2_1$	…	$a2_n$
$q1_0$	$q2_1$ …	$q2_1$ …		$q2_1$ …			
$q1_1$							
…							
$q1_p$							
$q2_0$							
$q2_1$							
…							
$q2_k$							

Композиция машин Тьюринга

- Итерация

T1	a_0	a_1	…	a_m
q_0				
q_1				
…				
q_p				

Композиция машин Тьюринга

- Итерация

T1T1…	a_0	a_1	…	a_{m-1}	a_m
q_0					
q_1					
…					
q_p					
q_{p+1}	q_1 …	q_1 …		q_1 …	

Программы МТ и неформальное определение алгоритма

- Программы для машины Тьюринга подпадают под неформальное определение алгоритма и обладают основными свойствами алгоритмов
  1. Дискретность
  2. Элементарность шагов
  3. Определенность (детерминированность)
  4. Конечность (финитность)
  5. Массовость

Тезис (гипотеза) Тьюринга

- Всякий алгоритм может быть реализован в виде программы для машины Тьюринга.

Нормальные алгорифмы Маркова

- В 1951 г. советский математик А.А. Марков предложил формализацию понятия алгоритма, названную нормальным алгорифмом.
- *Нормальный алгорифм Маркова (НАМ)* – вербальный (предназначенный для применения к словам в различных алфавитах) алгоритм, который задается с помощью пары
  - Алфавит алгорифма (к словам которого алгорифм будет применяться)
  - Схема алгорифма – конечный упорядоченный набор формул подстановки вида
    - $L \rightarrow R$ (простая формула; L – левая часть, R – правая часть)
    - $L \rightarrow .R$ (заключительная формула)

А.А. Марков (1903-1979)

Применение алгорифма к слову

1. Выбрать формулы подстановки, левая часть ***L*** которых входит в слово.
   Если таких формул нет, перейти к шагу 4.
   Из выбранных формул взять формулу с наименьшим номером.
2. Из всех возможных представлений слова в виде *left**L**right* выбрать такое, при котором подстрока *left* – самая короткая. Выполнить замену ***L*** на ***R***, считать слово *left**R**right* результатом текущего шага алгорифма.
3. Если выбранная формула является заключительной, то перейти к шагу 4. Иначе перейти к шагу 1.
4. СТОП.

Примеры НАМ

- Приписывание слова слева: *Ballast*(W)=ballastW
  1. Λ→.ballast
- Тождественная функция: *Identity*(W)=W
  1. Λ→.Λ
- "Стиратель": *Erase(W)*=Λ
  1. a→Λ
- "Обнулитель": *Zero(W)*=0
  1. a→Λ
  2. Λ→.0
- Упорядочение строки в алфавите {a,b,c}
  1. ba→ab
  2. ca→ac
  3. cb→bc

Пример НАМ

- Перевод слова из бинарной формы записи в унарную.
- Схема
  1. $|0 \to 0||$
  2. $1 \to 0|$
  3. $0 \to \Lambda$
- Пример применения
  1. $101 \quad {}^{2}\!\to 0|01$
  2. $0|01 \quad {}^{1}\!\to 00||1$
  3. $00||1 \quad {}^{2}\!\to 00||0|$
  4. $00||0| \quad {}^{1}\!\to 00|0|||$
  5. $00|0||| \quad {}^{1}\!\to 000|||||$
  6. $000||||| \quad {}^{3}\!\to 00|||||$
  7. $00||||| \quad {}^{3}\!\to 0|||||$
  8. $0||||| \quad {}^{3}\!\to |||||$

Пример НАМ

- Удваивание слова в бинарной форме записи.
- Схема

1. $ab+ \to b+a$
2. $ca \to a+ac$
3. $+ \to *$
4. $* \to \Lambda$
5. $c \to .\Lambda$
6. $\Lambda \to c$

- Пример применения

1. $1001 \xrightarrow{6} c1001$
2. $\mathbf{c1}001 \xrightarrow{2} 1+1c001$
3. $1+1\mathbf{c0}01 \xrightarrow{2} 1+10+0c01$
4. $1+\mathbf{10}+0c01 \xrightarrow{1} 1+0+10c01$
5. $1+0+10\mathbf{c0}1 \xrightarrow{2} 1+0+100+0c1$
6. $1+0+1\mathbf{00}+0c1 \xrightarrow{1} 1+0+10+00c1$
7. $1+0+\mathbf{10}+00c1 \xrightarrow{1} 1+0+0+100c1$
8. $1+0+0+100\mathbf{c1} \xrightarrow{2} 1+0+0+1001+1c$
9. $1+0+0+10\mathbf{01}+1c \xrightarrow{2} 1+0+0+101+0+1c$
10. $1+0+0+1\mathbf{01}+0+1c \xrightarrow{1} 1+0+0+11+00+1c$
11. $1+0+0+\mathbf{11}+00+1c \xrightarrow{1} 1+0+0+1+1+00+1c$
12. $1+0+0+1+1+\mathbf{00}+1c \xrightarrow{1} 1+0+0+1+1+0+0+1c$
13. $1+0+0+1+1+0+0+1c \xrightarrow{3} 1*0*0*1*1*0*0*1c$
14. $1*0*0*1*1*0*0*1c \xrightarrow{4} 10011001c$
15. $10011001c \xrightarrow{5} 10011001$

НАМ и неформальное определение алгоритма

- Нормальные алгорифмы Маркова подпадают под неформальное определение алгоритма и обладают основными свойствами алгоритмов
  1. Дискретность
  2. Элементарность шагов
  3. Определенность (детерминированность)
  4. Конечность (финитность)
  5. Массовость

Принцип нормализации (гипотеза) Маркова

- Всякий алгоритм может быть реализован в виде нормального алгорифма.

Эквивалентность МТ и НАМ

- Машины Тьюринга и нормальные алгорифмы Маркова являются эквивалентными алгоритмическими системами, т.е. описывают одно и то же множество алгоритмов.
- Любая алгоритмическая проблема, неразрешимая в терминах машины Тьюринга, окажется неразрешимой в терминах нормальных алгорифмов Маркова.

Алгоритмическая разрешимость

- *Алгоритмическая проблема* связна с построением алгоритма для решения задач некоторого класса.
- Всякая ли алгоритмическая проблема разрешима?
  - Вплоть до начала XX в. среди научного сообщества математиков считалось, что любая математическая проблема разрешима.
  - Примеры неразрешимых проблем:
    - 10-я проблема Д. Гильберта (неразрешимость доказана Ю.В. Матиясевичем): Найти способ установления разрешимости диофантова уравнения с произвольными целыми коэффициентами $P(x_1, \ldots, x_n)=Q(x_1, \ldots, x_n)$ (например, $ax^n+by^n=cz^n$) в целых рациональных числах
    - Проблема выводимости (неразрешимость доказана А. Черчем): Найти способ установления выводимости произвольного слова W2 из произвольного слова W1 с помощью произвольной системы правил подстановки.

Самоприменимость алгоритма

- Алгоритм называется *самоприменимым*, если он применим к слову, которое является его описанием.
- Пример самоприменимого алгоритма
  - Алгоритм
    1. a→b
    2. c→de
    3. ab→f
  - Применение к себе
    1. Подстановка 1: b→b, c→de, bb→f
    2. Подстановка 2: b→b, de→de, bb→f
    3. Стоп
- Пример несамоприменимого алгоритма
  - Алгоритм
    1. a→b
    2. b→a
  - Применение к себе
    1. Подстановка 1: b→b, b→a
    2. Подстановка 2: a→b, a→a
    3. Подстановка 1: b→b, b→a
    4. и т.д.

Распознавание самоприменимости алгоритмически неразрешимо

- Доказательство (от противного)
  - Пусть существует алгоритм A, распознающий самоприменимость:
  - Построим алгоритм B

$$A(P) = \begin{cases} 0, \text{ если } P \text{ не самоприменим} \\ 1, \text{ если } P \text{ самоприменим} \end{cases}$$

$$B(A) = \begin{cases} \text{не останавливается, если } A \text{ самоприменим} \\ 0, \text{ если } A \text{ не самоприменим} \end{cases}$$

т.е. B применим к самонеприменимым алгоритмам и не применим к самоприменимым.

Распознавание самоприменимости алгоритмически неразрешимо

- Применим алгоритм *B* к самому себе
  - если *B*(*B*)=0, то *B* самоприменим; противоречие с построением *B*.
  - если *B*(*B*) не останавливается, то *B* не самоприменим и по построению должен дать 0; противоречие.
- Таким образом, алгоритм *B* не может существовать.
- Следовательно, не может существовать алгоритм *A*, распознающий самоприменимость (*B* – пример алгоритма, к которому не применим алгоритм *A*).

Методы разработки алгоритмов

- Сведение задачи к подзадачам
  - разложение задачи в последовательность разнородных подзадач ("разделяй и властвуй")
  - разложение задачи в последовательность однородных подзадач (итерация)
  - сведение задачи к самой себе (рекурсия)
- Метод последовательных приближений
- Решение обратной задачи
- Метод полного перебора
- Эвристика

Заключение

- Интуитивное (неформальное) определение алгоритма
  1. Дискретность
  2. Элементарность шагов
  3. Определенность
  4. Конечность
  5. Массовость
- Формальное определение алгоритма
  - Машина Тьюринга (МТ)
  - Нормальные алгорифмы Маркова (НАМ)
  - Эквивалентность МТ и НАМ
- Самоприменимость алгоритма и проблема алгоритмической неразрешимости
- Методы разработки алгоритмов