Математическая логика и теория алгоритмов
Лекция 10
Теория алгоритмов. Машины Тьюринга и Поста

Куценко Дмитрий Александрович

Белгородский государственный технологический университет
имени В. Г. Шухова

Институт информационных технологий и управляющих систем

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

12 апреля 2013 г.

Машина Тьюринга

Машина Тьюринга — это абстрактный исполнитель алгоритма, состоящий из следующих компонент:

Машина Тьюринга

Машина Тьюринга — это абстрактный исполнитель алгоритма, состоящий из следующих компонент:

1. Бесконечная в обе стороны лента, разделённая на ячейки.

Машина Тьюринга

Машина Тьюринга — это абстрактный исполнитель алгоритма, состоящий из следующих компонент:

1. Бесконечная в обе стороны лента, разделённая на ячейки. Лента представляет собой внешнюю память.

Машина Тьюринга

Машина Тьюринга — это абстрактный исполнитель алгоритма, состоящий из следующих компонент:

1. Бесконечная в обе стороны лента, разделённая на ячейки. Лента представляет собой внешнюю память.
   Каждая ячейка ленты находится в одном и только в одном состоянии из множества $\mathscr{A} = \{a_0, a_1, \ldots, a_m\}$, называемого внешним алфавитом.

Машина Тьюринга

Машина Тьюринга — это абстрактный исполнитель алгоритма, состоящий из следующих компонент:

1. Бесконечная в обе стороны лента, разделённая на ячейки. Лента представляет собой внешнюю память.
   Каждая ячейка ленты находится в одном и только в одном состоянии из множества $\mathscr{A} = \{a_0, a_1, \ldots, a_m\}$, называемого внешним алфавитом.
   Состояние a_0 называется пустым и часто обозначается символом Λ.

Машина Тьюринга

Машина Тьюринга — это абстрактный исполнитель алгоритма, состоящий из следующих компонент:

1. Бесконечная в обе стороны лента, разделённая на ячейки. Лента представляет собой внешнюю память.
   Каждая ячейка ленты находится в одном и только в одном состоянии из множества $\mathscr{A} = \{a_0, a_1, \ldots, a_m\}$, называемого внешним алфавитом.
   Состояние a_0 называется пустым и часто обозначается символом Λ.
2. Внутренняя память, принимающая одно из состояний, входящих в множество $\mathscr{Q} = \{q_0, q_1, \ldots, q_n\}$, называемое внутренним алфавитом.

Машина Тьюринга

Машина Тьюринга — это абстрактный исполнитель алгоритма, состоящий из следующих компонент:

1. Бесконечная в обе стороны лента, разделённая на ячейки. Лента представляет собой внешнюю память.
   Каждая ячейка ленты находится в одном и только в одном состоянии из множества $\mathscr{A} = \{a_0, a_1, \ldots, a_m\}$, называемого внешним алфавитом.
   Состояние a_0 называется пустым и часто обозначается символом Λ.
2. Внутренняя память, принимающая одно из состояний, входящих в множество $\mathscr{Q} = \{q_0, q_1, \ldots, q_n\}$, называемое внутренним алфавитом.
   Состояние q_0 называется «стоп».

Машина Тьюринга

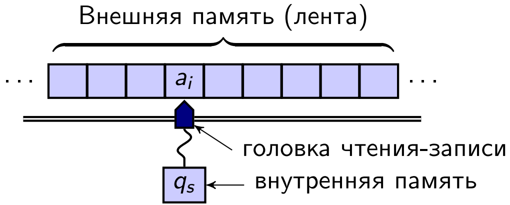
3. Головка чтения-записи, двигающаяся вдоль ленты и считывающая или записывающая содержимое ячейки, напротив которой она останавливается.

Машина Тьюринга

3. Головка чтения-записи, двигающаяся вдоль ленты и считывающая или записывающая содержимое ячейки, напротив которой она останавливается.
4. Механического устройства, передвигающего головку и меняющего состояния внешней и внутренней памяти.

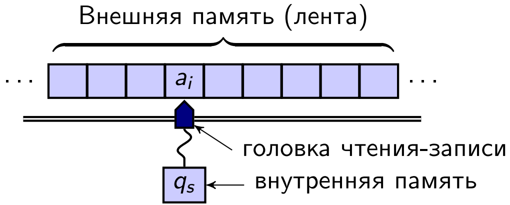
Машина Тьюринга

3. Головка чтения-записи, двигающаяся вдоль ленты и считывающая или записывающая содержимое ячейки, напротив которой она останавливается.
4. Механического устройства, передвигающего головку и меняющего состояния внешней и внутренней памяти.



Машина Тьюринга

3. Головка чтения-записи, двигающаяся вдоль ленты и считывающая или записывающая содержимое ячейки, напротив которой она останавливается.
4. Механического устройства, передвигающего головку и меняющего состояния внешней и внутренней памяти.



Если головка в состоянии q стоит напротив ячейки с номером k, то изменения состояния внутренней памяти и состояния ячейки происходят одновременно.

Команды машины Тьюринга

Работа машины Тьюринга осуществляется посредством команд, которые выполняет механическое устройство.

Команды машины Тьюринга

Работа машины Тьюринга осуществляется посредством команд, которые выполняет механическое устройство. Команда имеет один из следующих трёх возможных видов:

Команды машины Тьюринга

Работа машины Тьюринга осуществляется посредством команд, которые выполняет механическое устройство. Команда имеет один из следующих трёх возможных видов:

1. $q_s a_i \to q_t a_j$,
2. $q_s a_i \to q_t a_j L$,
3. $q_s a_i \to q_t a_j R$,

Команды машины Тьюринга

Работа машины Тьюринга осуществляется посредством команд, которые выполняет механическое устройство. Команда имеет один из следующих трёх возможных видов:

1. $q_s a_i \to q_t a_j$,
2. $q_s a_i \to q_t a_j L$,
3. $q_s a_i \to q_t a_j R$,

где L — это движение головки влево на одну ячейку, а R — вправо, $a_i, a_j \in \mathscr{A}$, $q_s, q_t \in \mathscr{Q}$, $q_s \neq q_0$.

Команды машины Тьюринга

Работа машины Тьюринга осуществляется посредством команд, которые выполняет механическое устройство. Команда имеет один из следующих трёх возможных видов:

1. $q_s a_i \to q_t a_j$,
2. $q_s a_i \to q_t a_j L$,
3. $q_s a_i \to q_t a_j R$,

где L — это движение головки влево на одну ячейку,
а R — вправо, $a_i, a_j \in \mathscr{A}$, $q_s, q_t \in \mathscr{Q}$, $q_s \neq q_0$.

Если головка в состоянии q_s обозревает ячейку в состоянии a_i, то внутреннее состояние меняется на q_t, а ячейки — на a_j.

Команды машины Тьюринга

Работа машины Тьюринга осуществляется посредством команд, которые выполняет механическое устройство. Команда имеет один из следующих трёх возможных видов:

1. $q_s a_i \to q_t a_j$,
2. $q_s a_i \to q_t a_j L$,
3. $q_s a_i \to q_t a_j R$,

где L — это движение головки влево на одну ячейку, а R — вправо, $a_i, a_j \in \mathscr{A}$, $q_s, q_t \in \mathscr{Q}$, $q_s \neq q_0$.

Если головка в состоянии q_s обозревает ячейку в состоянии a_i, то внутреннее состояние меняется на q_t, а ячейки — на a_j.

В команде 1 головка остаётся на месте, в 2 — смещается на одну ячейку влево, в 3 — смещается на одну ячейку вправо.

Команды машины Тьюринга

Работа машины Тьюринга осуществляется посредством команд, которые выполняет механическое устройство. Команда имеет один из следующих трёх возможных видов:

1. $q_s a_i \to q_t a_j$,
2. $q_s a_i \to q_t a_j L$,
3. $q_s a_i \to q_t a_j R$,

где L — это движение головки влево на одну ячейку, а R — вправо, $a_i, a_j \in \mathscr{A}$, $q_s, q_t \in \mathscr{Q}$, $q_s \neq q_0$.

Если головка в состоянии q_s обозревает ячейку в состоянии a_i, то внутреннее состояние меняется на q_t, а ячейки — на a_j.

В команде 1 головка остаётся на месте, в 2 — смещается на одну ячейку влево, в 3 — смещается на одну ячейку вправо.

Конечный набор команд образует программу.

Состояние машины Тьюринга

Состояние машины Тьюринга — это последовательность состояний $a_{i_1}, \ldots, a_{i_r}$ ячеек ленты, состояние внутренней памяти q_s и номер k читаемой ячейки в состоянии a_{i_k}.

Состояние машины Тьюринга

Состояние машины Тьюринга — это последовательность состояний $a_{i_1}, \ldots, a_{i_r}$ ячеек ленты, состояние внутренней памяти q_s и номер k читаемой ячейки в состоянии a_{i_k}. Состояние машины записывается в виде $a_{i_1} \ldots a_{i_{k-1}} q_s a_{i_k} \ldots a_{i_r}$ и называется машинным словом $\mathfrak{m}$ в алфавите $\mathscr{A} \cup \mathscr{Q}$.

Состояние машины Тьюринга

Состояние машины Тьюринга — это последовательность состояний $a_{i_1}, \ldots, a_{i_r}$ ячеек ленты, состояние внутренней памяти q_s и номер k читаемой ячейки в состоянии a_{i_k}.

Состояние машины записывается в виде $a_{i_1} \ldots a_{i_{k-1}} q_s a_{i_k} \ldots a_{i_r}$ и называется машинным словом $\mathfrak{m}$ в алфавите $\mathscr{A} \cup \mathscr{Q}$.

Символ q_s может быть самым левым, но не может быть самым правым в машинном слове, так как справа от него должно быть считываемое состояние ячейки.

Состояние машины Тьюринга

Состояние машины Тьюринга — это последовательность состояний $a_{i_1}, \ldots, a_{i_r}$ ячеек ленты, состояние внутренней памяти q_s и номер k читаемой ячейки в состоянии a_{i_k}.
Состояние машины записывается в виде $a_{i_1} \ldots a_{i_{k-1}} q_s a_{i_k} \ldots a_{i_r}$ и называется машинным словом $\mathfrak{m}$ в алфавите $\mathscr{A} \cup \mathscr{Q}$.
Символ q_s может быть самым левым, но не может быть самым правым в машинном слове, так как справа от него должно быть считываемое состояние ячейки.
Под воздействием программы происходит изменение состояния машины, сопровождающееся переделкой исходного машинного слова $\mathfrak{m} \to \mathfrak{m}^{(1)} \to \ldots \to \mathfrak{m}^{(p)}$.

Состояние машины Тьюринга

Состояние машины Тьюринга — это последовательность состояний $a_{i_1}, \ldots, a_{i_r}$ ячеек ленты, состояние внутренней памяти q_s и номер k читаемой ячейки в состоянии a_{i_k}.
Состояние машины записывается в виде $a_{i_1} \ldots a_{i_{k-1}} q_s a_{i_k} \ldots a_{i_r}$ и называется машинным словом $\mathfrak{m}$ в алфавите $\mathscr{A} \cup \mathscr{Q}$.
Символ q_s может быть самым левым, но не может быть самым правым в машинном слове, так как справа от него должно быть считываемое состояние ячейки.
Под воздействием программы происходит изменение состояния машины, сопровождающееся переделкой исходного машинного слова $\mathfrak{m} \to \mathfrak{m}^{(1)} \to \ldots \to \mathfrak{m}^{(p)}$.
В этом случае говорят, что машина $\mathfrak{T}$ перерабатывает слово $\mathfrak{m}$ в слово $\mathfrak{m}^{(p)}$, и обозначать этот факт $\mathfrak{m}^{(p)} = \mathfrak{T}(\mathfrak{m})$.

Состояние машины Тьюринга

Состояние машины Тьюринга — это последовательность состояний $a_{i_1}, \ldots, a_{i_r}$ ячеек ленты, состояние внутренней памяти q_s и номер k читаемой ячейки в состоянии a_{i_k}.
Состояние машины записывается в виде $a_{i_1} \ldots a_{i_{k-1}} q_s a_{i_k} \ldots a_{i_r}$ и называется машинным словом $\mathfrak{m}$ в алфавите $\mathscr{A} \cup \mathscr{Q}$.
Символ q_s может быть самым левым, но не может быть самым правым в машинном слове, так как справа от него должно быть считываемое состояние ячейки.
Под воздействием программы происходит изменение состояния машины, сопровождающееся переделкой исходного машинного слова $\mathfrak{m} \to \mathfrak{m}^{(1)} \to \ldots \to \mathfrak{m}^{(p)}$.
В этом случае говорят, что машина $\mathfrak{T}$ перерабатывает слово $\mathfrak{m}$ в слово $\mathfrak{m}^{(p)}$, и обозначать этот факт $\mathfrak{m}^{(p)} = \mathfrak{T}(\mathfrak{m})$.
Запись $\mathfrak{T}(\mathfrak{m})$ можно употреблять и в другом смысле — просто как обозначение машины $\mathfrak{T}$ с исходными данными $\mathfrak{m}$.

Пример

Нахождение суммы чисел a и b, представленных в унарной системе счисления. Между числами записан разделитель «$*$».

Пример

Нахождение суммы чисел a и b, представленных в унарной системе счисления. Между числами записан разделитель «$*$».

Решение

Пример

Нахождение суммы чисел a и b, представленных в унарной системе счисления. Между числами записан разделитель «$*$».

Решение

Пусть запись 1^n означает блок из n символов «1».

Пример

Нахождение суммы чисел a и b, представленных в унарной системе счисления. Между числами записан разделитель «$*$».

Решение

Пусть запись 1^n означает блок из n символов «1». Необходимо преобразовать слово $1^a{*}1^b$ в слово 1^{a+b}, т. е. удалить разделитель «$*$» и сдвинуть одно из слагаемых к другому.

Пример

Нахождение суммы чисел a и b, представленных в унарной системе счисления. Между числами записан разделитель «$*$».

Решение

Пусть запись 1^n означает блок из n символов «1».
Необходимо преобразовать слово $1^a{*}1^b$ в слово 1^{a+b}, т. е. удалить разделитель «$*$» и сдвинуть одно из слагаемых к другому.
Это преобразование осуществляет машина Тьюринга с четырьмя состояниями и следующей системой команд:

Пример

Нахождение суммы чисел a и b, представленных в унарной системе счисления. Между числами записан разделитель «$*$».

Решение

Пусть запись 1^n означает блок из n символов «1».
Необходимо преобразовать слово $1^a{*}1^b$ в слово 1^{a+b}, т. е. удалить разделитель «$*$» и сдвинуть одно из слагаемых к другому.
Это преобразование осуществляет машина Тьюринга с четырьмя состояниями и следующей системой команд:

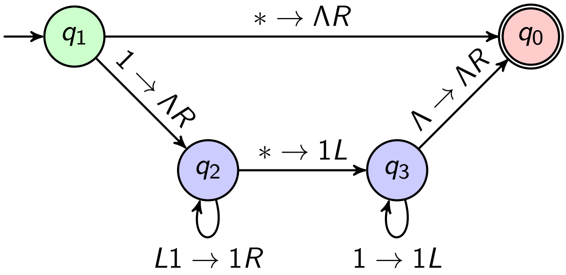
$$q_1* \to q_0\Lambda R;$$
$$q_1 1 \to q_2\Lambda R;$$
$$q_2 1 \to q_2 1 R;$$
$$q_2* \to q_3 1 L;$$
$$q_3 1 \to q_3 1 L;$$
$$q_3\Lambda \to q_0\Lambda R.$$

Граф переходов

Команды удобно представлять в виде графа переходов:

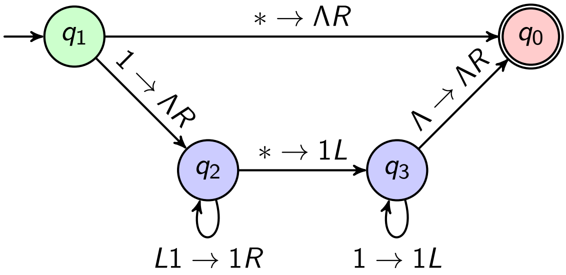
Граф переходов

Команды удобно представлять в виде графа переходов:



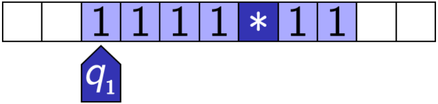
Граф переходов

Команды удобно представлять в виде графа переходов:



Заключительное состояние обозначено двойной окружностью.

Шаг 1



$q_1 * \to q_0 \Lambda R;$

$q_1 1 \to q_2 \Lambda R;$

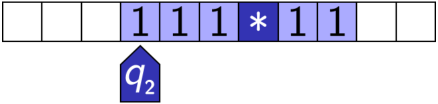
$q_2 1 \to q_2 1 R;$

$q_2 * \to q_3 1 L;$

$q_3 1 \to q_3 1 L;$

$q_3 \Lambda \to q_0 \Lambda R.$

Шаг 2



$q_1 * \to q_0 \Lambda R;$

$q_1 1 \to q_2 \Lambda R;$

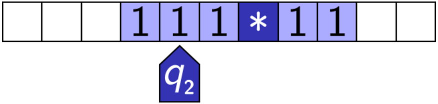
$q_2 1 \to q_2 1 R;$

$q_2 * \to q_3 1 L;$

$q_3 1 \to q_3 1 L;$

$q_3 \Lambda \to q_0 \Lambda R.$

Шаг 3



$q_1{*} \to q_0\Lambda R;$

$q_1 1 \to q_2\Lambda R;$

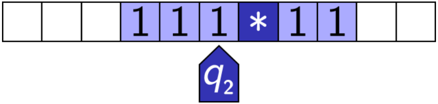
$q_2 1 \to q_2 1 R;$

$q_2{*} \to q_3 1 L;$

$q_3 1 \to q_3 1 L;$

$q_3\Lambda \to q_0\Lambda R.$

Шаг 4



$q_1 * \to q_0 \Lambda R;$

$q_1 1 \to q_2 \Lambda R;$

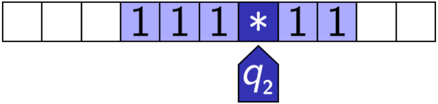
$q_2 1 \to q_2 1 R;$

$q_2 * \to q_3 1 L;$

$q_3 1 \to q_3 1 L;$

$q_3 \Lambda \to q_0 \Lambda R.$

Шаг 5



$q_1 * \to q_0 \Lambda R;$

$q_1 1 \to q_2 \Lambda R;$

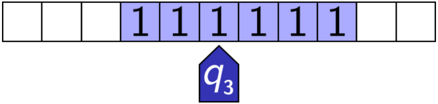
$q_2 1 \to q_2 1 R;$

$q_2 * \to q_3 1 L;$

$q_3 1 \to q_3 1 L;$

$q_3 \Lambda \to q_0 \Lambda R.$

Шаг 6



$q_1 * \to q_0 \Lambda R;$

$q_1 1 \to q_2 \Lambda R;$

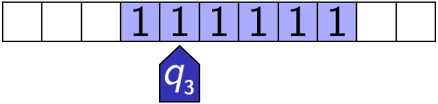
$q_2 1 \to q_2 1 R;$

$q_2 * \to q_3 1 L;$

$q_3 1 \to q_3 1 L;$

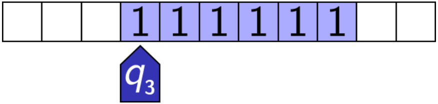
$q_3 \Lambda \to q_0 \Lambda R.$

Шаг 7



$$q_1 * \to q_0 \Lambda R;$$
$$q_1 1 \to q_2 \Lambda R;$$
$$q_2 1 \to q_2 1 R;$$
$$q_2 * \to q_3 1 L;$$
$$q_3 1 \to q_3 1 L;$$
$$q_3 \Lambda \to q_0 \Lambda R.$$

Шаг 8



$q_1 * \to q_0 \Lambda R;$

$q_1 1 \to q_2 \Lambda R;$

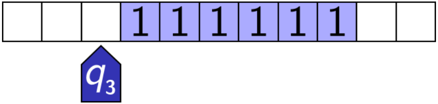
$q_2 1 \to q_2 1 R;$

$q_2 * \to q_3 1 L;$

$q_3 1 \to q_3 1 L;$

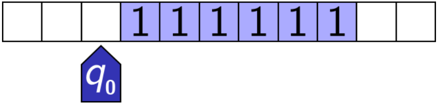
$q_3 \Lambda \to q_0 \Lambda R.$

Шаг 9



$q_1 * \to q_0 \Lambda R;$
$q_1 1 \to q_2 \Lambda R;$
$q_2 1 \to q_2 1 R;$
$q_2 * \to q_3 1 L;$
$q_3 1 \to q_3 1 L;$
$q_3 \Lambda \to q_0 \Lambda R.$

Шаг 10



$$q_1 * \to q_0 \Lambda R;$$
$$q_1 1 \to q_2 \Lambda R;$$
$$q_2 1 \to q_2 1 R;$$
$$q_2 * \to q_3 1 L;$$
$$q_3 1 \to q_3 1 L;$$
$$q_3 \Lambda \to q_0 \Lambda R.$$

Тезис Тьюринга

Тезис Тьюринга

Все вычислимые функции вычисляются на машинах Тьюринга.

Универсальная машина Тьюринга

Машина Тьюринга называется универсальной, если она может при определённых начальных входных данных вычислить любую функцию, которая вычислима на какой-либо машине Тьюринга.

Универсальная машина Тьюринга

Машина Тьюринга называется универсальной, если она может при определённых начальных входных данных вычислить любую функцию, которая вычислима на какой-либо машине Тьюринга.
Иначе говоря, с учётом тезиса Тьюринга можно сказать, что универсальная машина Тьюринга способна вычислить любую вычислимую функцию.

Универсальная машина Тьюринга

Машина Тьюринга называется универсальной, если она может при определённых начальных входных данных вычислить любую функцию, которая вычислима на какой-либо машине Тьюринга.
Иначе говоря, с учётом тезиса Тьюринга можно сказать, что универсальная машина Тьюринга способна вычислить любую вычислимую функцию.
Доказано, что универсальная машина Тьюринга существует.

Универсальная машина Тьюринга

Машина Тьюринга называется универсальной, если она может при определённых начальных входных данных вычислить любую функцию, которая вычислима на какой-либо машине Тьюринга.
Иначе говоря, с учётом тезиса Тьюринга можно сказать, что универсальная машина Тьюринга способна вычислить любую вычислимую функцию.
Доказано, что универсальная машина Тьюринга существует.
Существование универсальной машины Тьюринга означает, что систему команд любой машины $\mathfrak{T}$ можно интерпретировать двояким образом: либо как описание работы конкретного устройства машины $\mathfrak{T}$, либо как программу для универсальной машины $\mathfrak{U}$.

Универсальная машина Тьюринга

Машина Тьюринга называется универсальной, если она может при определённых начальных входных данных вычислить любую функцию, которая вычислима на какой-либо машине Тьюринга.
Иначе говоря, с учётом тезиса Тьюринга можно сказать, что универсальная машина Тьюринга способна вычислить любую вычислимую функцию.
Доказано, что универсальная машина Тьюринга существует.
Существование универсальной машины Тьюринга означает, что систему команд любой машины $\mathfrak{T}$ можно интерпретировать двояким образом: либо как описание работы конкретного устройства машины $\mathfrak{T}$, либо как программу для универсальной машины $\mathfrak{U}$.
При этом универсальная машина $\mathfrak{U}$ имитирует работу машины $\mathfrak{T}$, тратя несколько шагов своей работы на имитацию каждого шага машины $\mathfrak{T}$.

Машина Поста

Машина Поста — абстрактная вычислительная машина, предложенная Эмилем Леоном Постом, которая эквивалентна машине Тьюринга и отличается от неё большей простотой.

Машина Поста

Машина Поста — абстрактная вычислительная машина, предложенная Эмилем Леоном Постом, которая эквивалентна машине Тьюринга и отличается от неё большей простотой. Машина Поста состоит из следующих компонент:

Машина Поста

Машина Поста — абстрактная вычислительная машина, предложенная Эмилем Леоном Постом, которая эквивалентна машине Тьюринга и отличается от неё большей простотой. Машина Поста состоит из следующих компонент:

1. Разбитая на ячейки бесконечная в обе стороны лента. Ячейки ленты пронумерованы $(\ldots, -2, -1, 0, 1, 2, \ldots)$.

Машина Поста

Машина Поста — абстрактная вычислительная машина, предложенная Эмилем Леоном Постом, которая эквивалентна машине Тьюринга и отличается от неё большей простотой. Машина Поста состоит из следующих компонент:

1. Разбитая на ячейки бесконечная в обе стороны лента. Ячейки ленты пронумерованы $(\ldots, -2, -1, 0, 1, 2, \ldots)$. Каждая ячейка ленты может быть либо пустой (0), либо отмеченной меткой (1).

Машина Поста

Машина Поста — абстрактная вычислительная машина, предложенная Эмилем Леоном Постом, которая эквивалентна машине Тьюринга и отличается от неё большей простотой. Машина Поста состоит из следующих компонент:

1. Разбитая на ячейки бесконечная в обе стороны лента. Ячейки ленты пронумерованы $(\ldots, -2, -1, 0, 1, 2, \ldots)$. Каждая ячейка ленты может быть либо пустой (0), либо отмеченной меткой (1).
2. головка чтения-записи — за один шаг может сдвинуться на одну позицию влево или вправо, считать, поставить или удалить метку в том месте, где она стоит.

Команды машины Поста

Работа машины Поста определяется программой, состоящей из конечного числа команд. Каждая команда имеет уникальный номер N, который записывается перед ней.

Команды машины Поста

Работа машины Поста определяется программой, состоящей из конечного числа команд. Каждая команда имеет уникальный номер N, который записывается перед ней.

Команды машины Поста

1. $N. \to J$ — сдвинуть головку вправо и перейти на команду J;

Команды машины Поста

Работа машины Поста определяется программой, состоящей из конечного числа команд. Каждая команда имеет уникальный номер N, который записывается перед ней.

Команды машины Поста

1. $N. \to J$ — сдвинуть головку вправо и перейти на команду J;
2. $N. \leftarrow J$ — сдвинуть головку влево и перейти на команду J;

Команды машины Поста

Работа машины Поста определяется программой, состоящей из конечного числа команд. Каждая команда имеет уникальный номер N, который записывается перед ней.

Команды машины Поста

1. $N. \rightarrow J$ — сдвинуть головку вправо и перейти на команду J;
2. $N. \leftarrow J$ — сдвинуть головку влево и перейти на команду J;
3. $N.\ 1\ J$ — поставить метку в текущей ячейке и перейти на команду J;

Команды машины Поста

Работа машины Поста определяется программой, состоящей из конечного числа команд. Каждая команда имеет уникальный номер N, который записывается перед ней.

Команды машины Поста

1. $N.\ \rightarrow\ J$ — сдвинуть головку вправо и перейти на команду J;
2. $N.\ \leftarrow\ J$ — сдвинуть головку влево и перейти на команду J;
3. $N.\ 1\ J$ — поставить метку в текущей ячейке и перейти на команду J;
4. $N.\ 0\ J$ — удалить метку в текущей ячейке и перейти на команду J;

Команды машины Поста

Работа машины Поста определяется программой, состоящей из конечного числа команд. Каждая команда имеет уникальный номер N, который записывается перед ней.

Команды машины Поста

1. $N.\ \to\ J$ — сдвинуть головку вправо и перейти на команду J;
2. $N.\ \leftarrow\ J$ — сдвинуть головку влево и перейти на команду J;
3. $N.\ 1\ J$ — поставить метку в текущей ячейке и перейти на команду J;
4. $N.\ 0\ J$ — удалить метку в текущей ячейке и перейти на команду J;
5. $N.\ ?\ J_1 : J_2$ — считывание и условный переход: если в текущей ячейке есть метка, то перейти на команду J_1, если метки нет — на команду J_2;

Команды машины Поста

Работа машины Поста определяется программой, состоящей из конечного числа команд. Каждая команда имеет уникальный номер N, который записывается перед ней.

Команды машины Поста

1. $N. \rightarrow J$ — сдвинуть головку вправо и перейти на команду J;
2. $N. \leftarrow J$ — сдвинуть головку влево и перейти на команду J;
3. $N.\ 1\ J$ — поставить метку в текущей ячейке и перейти на команду J;
4. $N.\ 0\ J$ — удалить метку в текущей ячейке и перейти на команду J;
5. $N.\ ?\ J_1 : J_2$ — считывание и условный переход: если в текущей ячейке есть метка, то перейти на команду J_1, если метки нет — на команду J_2;
6. $N.$ Stop — остановить машину (результат — на ленте).

Для работы машины нужно задать программу
и начальное состояние — состояние ленты и позицию каретки.

Для работы машины нужно задать программу
и начальное состояние — состояние ленты и позицию каретки.
Первой должна выполняться команда № 1.

Для работы машины нужно задать программу
и начальное состояние — состояние ленты и позицию каретки.
Первой должна выполняться команда № 1.
После запуска возможны варианты:

Для работы машины нужно задать программу
и начальное состояние — состояние ленты и позицию каретки.
Первой должна выполняться команда № 1.
После запуска возможны варианты:

1. работа может закончиться командой Stop;

Для работы машины нужно задать программу и начальное состояние — состояние ленты и позицию каретки. Первой должна выполняться команда № 1.

После запуска возможны варианты:

1. работа может закончиться командой Stop;
2. работа может закончиться невыполнимой командой (стирание несуществующей метки, запись метки в помеченную ячейку, переход на команду с несуществующим номером);

Для работы машины нужно задать программу
и начальное состояние — состояние ленты и позицию каретки.
Первой должна выполняться команда № 1.
После запуска возможны варианты:

1. работа может закончиться командой Stop;
2. работа может закончиться невыполнимой командой (стирание несуществующей метки, запись метки в помеченную ячейку, переход на команду с несуществующим номером);
3. работа никогда не закончится.

Пример

Вычитание двух чисел $(a - b)$, записанных в унарной системе и разделённых пустой ячейкой.
К каждому числу добавляется единица, т. о. $1111 = 3$.

Пример

Вычитание двух чисел $(a - b)$, записанных в унарной системе и разделённых пустой ячейкой.

К каждому числу добавляется единица, т. о. $1111 = 3$.

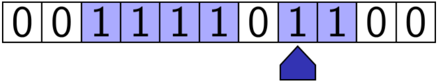
Пример начального состояния $(3 - 1)$:

Пример

Вычитание двух чисел ($a - b$), записанных в унарной системе и разделённых пустой ячейкой.

К каждому числу добавляется единица, т. о. $1111 = 3$.

Пример начального состояния ($3 - 1$):



Для реализации вычитания подойдёт следующая программа:

Пример

Вычитание двух чисел $(a - b)$, записанных в унарной системе и разделённых пустой ячейкой.

К каждому числу добавляется единица, т. о. $1111 = 3$.

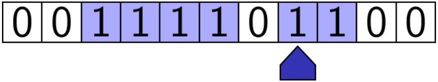
Пример начального состояния $(3 - 1)$:

0	0	1	1	1	1	0	1	1	0	0

Для реализации вычитания подойдёт следующая программа:

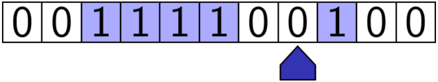
1. 0 2 — уменьшить второе число на 1;
2. $\rightarrow$ 3
3. ? 5 : 4 — второе число не равно 0?
4. Stop — конец, разность на ленте.
5. $\leftarrow$ 6 — сдвигаться влево. . .
6. ? 7 : 5 — . . . пока не найдено первое число;
7. 0 8 — уменьшить первое число на 1;
8. $\rightarrow$ 9 — сдвигаться вправо. . .
9. ? 1 : 8 — . . . пока не найдено второе число;

Начальное состояние



1. 0 2
2. → 3
3. ? 5 : 4
4. Stop
5. ← 6
6. ? 7 : 5
7. 0 8
8. → 9
9. ? 1 : 8

Шаг 1



1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 2

0	0	1	1	1	1	0	0	1	0	0

1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 3

0	0	1	1	1	1	0	0	1	0	0

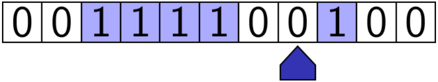
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 4

0	0	1	1	1	1	0	0	1	0	0

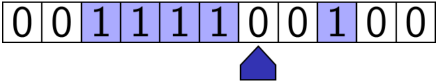
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 5



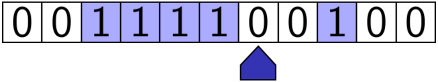
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 6



1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 7



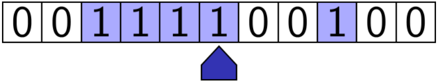
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 8

0	0	1	1	1	1	0	0	1	0	0

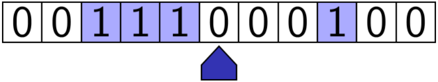
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 9



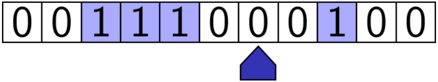
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 10



1. 0 2
2. → 3
3. ? 5 : 4
4. Stop
5. ← 6
6. ? 7 : 5
7. 0 8
8. → 9
9. ? 1 : 8

Шаг 11



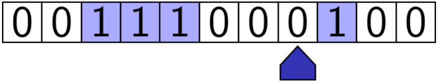
1. 0 2
2. $\to$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\to$ 9
9. ? 1 : 8

Шаг 12

0	0	1	1	1	0	0	0	1	0	0

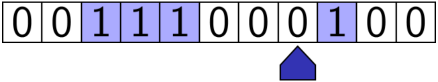
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 13



1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 14



1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 15

0	0	1	1	1	0	0	0	1	0	0

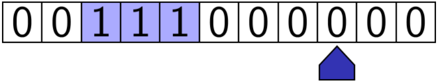
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 16

0	0	1	1	1	0	0	0	1	0	0

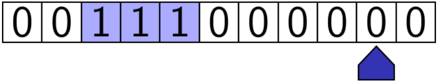
1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Шаг 17



1. 0 2
2. → 3
3. ? 5 : 4
4. Stop
5. ← 6
6. ? 7 : 5
7. 0 8
8. → 9
9. ? 1 : 8

Шаг 18



1. 0 2
2. → 3
3. ? 5 : 4
4. Stop
5. ← 6
6. ? 7 : 5
7. 0 8
8. → 9
9. ? 1 : 8

Шаг 19

0	0	1	1	1	0	0	0	0	0	0

1. 0 2
2. $\rightarrow$ 3
3. ? 5 : 4
4. Stop
5. $\leftarrow$ 6
6. ? 7 : 5
7. 0 8
8. $\rightarrow$ 9
9. ? 1 : 8

Язык Brainfuck

Brainfuck — один из известнейших эзотерических языков программирования, придуманный У. Мюллером в 1993 г. для забавы.

Язык Brainfuck

Brainfuck — один из известнейших эзотерических языков программирования, придуманный У. Мюллером в 1993 г. для забавы.

Урбан Доминик Мю́ллер — швейцарский программист.

Язык Brainfuck

Brainfuck — один из известнейших эзотерических языков программирования, придуманный У. Мюллером в 1993 г. для забавы.

Урбан Доминик Мю́ллер — швейцарский программист.

Язык имеет восемь команд, каждая из которых записывается одним символом.

Язык Brainfuck

Brainfuck — один из известнейших эзотерических языков программирования, придуманный У. Мюллером в 1993 г. для забавы.

Урбан Доминик Мю́ллер — швейцарский программист.

Язык имеет восемь команд, каждая из которых записывается одним символом.
Исходный код программы на языке Brainfuck представляет собой последовательность этих символов без какого-либо дополнительного синтаксиса.

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.

Команды языка Brainfuck:

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.

Команды языка Brainfuck:

1. > — перейти к следующей ячейке;

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.

Команды языка Brainfuck:

1. > — перейти к следующей ячейке;
2. < — перейти к предыдущей ячейке;

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.
Команды языка Brainfuck:

1. > — перейти к следующей ячейке;
2. < — перейти к предыдущей ячейке;
3. + — увеличить значение текущей ячейки на 1;

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.
Команды языка Brainfuck:

1. > — перейти к следующей ячейке;
2. < — перейти к предыдущей ячейке;
3. + — увеличить значение текущей ячейки на 1;
4. - — уменьшить значение текущей ячейки на 1;

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.

Команды языка Brainfuck:

1. `>` — перейти к следующей ячейке;
2. `<` — перейти к предыдущей ячейке;
3. `+` — увеличить значение текущей ячейки на 1;
4. `-` — уменьшить значение текущей ячейки на 1;
5. `.` — напечатать значение текущей ячейки;

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.
Команды языка Brainfuck:

1. > — перейти к следующей ячейке;
2. < — перейти к предыдущей ячейке;
3. + — увеличить значение текущей ячейки на 1;
4. - — уменьшить значение текущей ячейки на 1;
5. . — напечатать значение текущей ячейки;
6. , — ввести извне значение и сохранить его в текущей ячейке;

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.
Команды языка Brainfuck:

1. > — перейти к следующей ячейке;
2. < — перейти к предыдущей ячейке;
3. + — увеличить значение текущей ячейки на 1;
4. - — уменьшить значение текущей ячейки на 1;
5. . — напечатать значение текущей ячейки;
6. , — ввести извне значение и сохранить его в текущей ячейке;
7. [— если значение текущей ячейки равно 0, перейти вперёд на команду, следующую за соответствующей командой «]» (с учётом вложенности);

Среда и команды языка Brainfuck

Машина, которой управляют команды Brainfuck, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминающих ленту и головку машины Тьюринга.
Команды языка Brainfuck:

1. `>` — перейти к следующей ячейке;
2. `<` — перейти к предыдущей ячейке;
3. `+` — увеличить значение текущей ячейки на 1;
4. `-` — уменьшить значение текущей ячейки на 1;
5. `.` — напечатать значение текущей ячейки;
6. `,` — ввести извне значение и сохранить его в текущей ячейке;
7. `[` — если значение текущей ячейки равно 0, перейти вперёд на команду, следующую за соответствующей командой «`]`» (с учётом вложенности);
8. `]` — если значение текущей ячейки не равно 0, перейти назад на команду, следующую за соответствующей командой «`[`» (с учётом вложенности).

Классический Brainfuck

В «классическом» Brainfuck, описанном Мюллером, размер ячейки — один байт, количество ячеек — 30 000.

Классический Brainfuck

В «классическом» Brainfuck, описанном Мюллером, размер ячейки — один байт, количество ячеек — 30 000.
В начальном состоянии указатель находится в крайней левой позиции, а все ячейки заполнены нулями.

Классический Brainfuck

В «классическом» Brainfuck, описанном Мюллером, размер ячейки — один байт, количество ячеек — 30 000.
В начальном состоянии указатель находится в крайней левой позиции, а все ячейки заполнены нулями.
Увеличение и уменьшение значений ячеек происходят по модулю 256.

Классический Brainfuck

В «классическом» Brainfuck, описанном Мюллером, размер ячейки — один байт, количество ячеек — 30 000.

В начальном состоянии указатель находится в крайней левой позиции, а все ячейки заполнены нулями.

Увеличение и уменьшение значений ячеек происходят по модулю 256.

Ввод и вывод также происходят побайтно, с учётом кодировки ASCII (т. е. в результате операции ввода «,» символ «1» будет записан в текущую ячейку как число 49, а операция вывода «.», совершённая над ячейкой, содержащей число 65, напечатает латинскую букву «A»).

Классический Brainfuck

В «классическом» Brainfuck, описанном Мюллером, размер ячейки — один байт, количество ячеек — 30 000.
В начальном состоянии указатель находится в крайней левой позиции, а все ячейки заполнены нулями.
Увеличение и уменьшение значений ячеек происходят по модулю 256.
Ввод и вывод также происходят побайтно, с учётом кодировки ASCII (т. е. в результате операции ввода « , » символ «1» будет записан в текущую ячейку как число 49, а операция вывода « . », совершённая над ячейкой, содержащей число 65, напечатает латинскую букву «A»).
В других вариантах языка размер и количество ячеек может быть бо́льшими.

Классический Brainfuck

В «классическом» Brainfuck, описанном Мюллером, размер ячейки — один байт, количество ячеек — 30 000.
В начальном состоянии указатель находится в крайней левой позиции, а все ячейки заполнены нулями.
Увеличение и уменьшение значений ячеек происходят по модулю 256.
Ввод и вывод также происходят побайтно, с учётом кодировки ASCII (т. е. в результате операции ввода «,» символ «1» будет записан в текущую ячейку как число 49, а операция вывода «.», совершённая над ячейкой, содержащей число 65, напечатает латинскую букву «A»).
В других вариантах языка размер и количество ячеек может быть бо́льшими.
Есть версии, где значение ячеек не является целочисленным (ячейки содержат числа с плавающей точкой).

Соответствие языку C

Язык Brainfuck можно описать с помощью эквивалентов языка C:

Соответствие языку C

Язык Brainfuck можно описать с помощью эквивалентов языка C:

Brainfuck	C
>	`++ptr;`
<	`--ptr;`
+	`++*ptr;`
-	`--*ptr;`
.	`putchar(*ptr);`
,	`*ptr = getchar();`
[	`while (*ptr) {`
]	`}`

Соответствие языку C

Язык Brainfuck можно описать с помощью эквивалентов языка C:

Brainfuck	C
`>`	`++ptr;`
`<`	`--ptr;`
`+`	`++*ptr;`
`-`	`--*ptr;`
`.`	`putchar(*ptr);`
`,`	`*ptr = getchar();`
`[`	`while (*ptr) {`
`]`	`}`

Переменная `ptr` объявлена как указатель на байт.

Пример

Программа на языке Brainfuck, печатающая фразу «Hello World!»:

Пример

Программа на языке Brainfuck, печатающая фразу «Hello World!»:

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.
>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.
------.--------.>+.>.
```

Полнота по Тьюрингу языка Brainfuck

Несмотря на внешнюю примитивность, Brainfuck
с бесконечным набором ячеек имеет тьюринговскую полноту.

Полнота по Тьюрингу языка Brainfuck

Несмотря на внешнюю примитивность, Brainfuck с бесконечным набором ячеек имеет тьюринговскую полноту. Следовательно, по потенциальным возможностям он не уступает «настоящим» языкам, подобным C, Pascal или Java.

Полнота по Тьюрингу языка Brainfuck

Несмотря на внешнюю примитивность, Brainfuck
с бесконечным набором ячеек имеет тьюринговскую полноту.
Следовательно, по потенциальным возможностям он
не уступает «настоящим» языкам, подобным C, Pascal или
Java.
Brainfuck подходит для экспериментов по генетическому
программированию из-за простоты синтаксиса и лёгкости
генерации исходного кода.

Расширенный Brainfuck

Существует расширение классического языка Brainfuck, позволяющее использовать процедуры.

Расширенный Brainfuck

Существует расширение классического языка Brainfuck, позволяющее использовать процедуры.
Для поддержки этого расширения дополнительно вводятся три новых команды:

Расширенный Brainfuck

Существует расширение классического языка Brainfuck, позволяющее использовать процедуры.

Для поддержки этого расширения дополнительно вводятся три новых команды:

9. (— начало объявления процедуры (идентификатором процедуры служит число, хранящееся в текущей ячейке);

Расширенный Brainfuck

Существует расширение классического языка Brainfuck, позволяющее использовать процедуры.

Для поддержки этого расширения дополнительно вводятся три новых команды:

9. (— начало объявления процедуры (идентификатором процедуры служит число, хранящееся в текущей ячейке);
10.) — конец объявления процедуры;

Расширенный Brainfuck

Существует расширение классического языка Brainfuck, позволяющее использовать процедуры.

Для поддержки этого расширения дополнительно вводятся три новых команды:

9. (— начало объявления процедуры (идентификатором процедуры служит число, хранящееся в текущей ячейке);
10.) — конец объявления процедуры;
11. : — вызов процедуры с идентификатором, равным числу, хранящемуся в текущей ячейке.

Ссылка для скачивания этой лекции:

http://zalil.ru/32129818