

АСА лекция

silvia.lesnaia

February 2025

11.02.25

1 Введение в теорию алгоритмов

2 Примеры интуитивного понятия алгоритма

Алгоритм - точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время

Алгоритм - это понятные и точные предписания исполнителю совершить конечное число шагов, направленных на решение поставленной задачи

Алгоритм - это конечный набор правил, который определяет последовательность операций

2.1 Основные свойства алгоритмов

Дискретность

Детерминированность

Понятность

Завершаемость

Массовость

Однозначность результата

2.2 Основные задачи теории алгоритмов

формализация понятия «алгоритм» и исследование формальных алгоритмических систем;

формальное доказательство алгоритмической неразрешимости ряда задач;

классификация задач, определение и исследование сложностных классов;

асимптотический анализ сложности алгоритмов;

исследование и анализ рекурсивных алгоритмов;

получение явных функций трудоемкости в целях сравнительного анализа алгоритмов;

разработка критериев сравнительной оценки качества алгоритмов.

2.3 Схема определения понятия «алгоритм»:

Понятие данных

Память

Элементарный шаг

Детерминированность

Результативность

2.4 Основные типы алгоритмических моделей

Алгоритм как некое детерминированное устройство - абстрактные машины. Машина Тьюринга и машина Поста.

Алгоритм как процедура вычисления некой числовой функции. Рекурсивные функции Черча.

Алгоритм как последовательность преобразований цепочек в каком-либо алфавите. (Комбинаторные операции над словами). Нормальные алгоритмы Маркова.

3 Машина Поста

Тезис Поста - "Всякий алгоритм представим в форме машины Поста".

Алгоритм (по Посту) — программа для машины Поста, приводящая к решению поставленной задачи.

Если задача имеет алгоритмическое решение, то она представима в форме команд для машины Поста.

3.1 Варианты окончания выполнения программы на машине Поста

останов по команде "стоп". Такой останов называется результативным и указывает на корректность алгоритма;

останов при выполнении недопустимой команды. Случай, когда указатель должен записать метку там, где она уже есть, или стереть метку там, где ее нет;

машина не останавливается никогда. Уход в бесконечность, закливание.

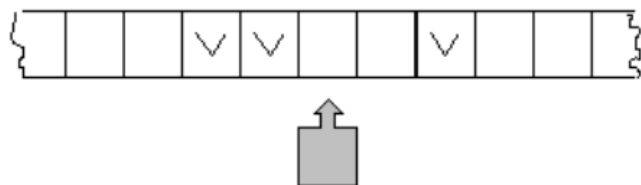
3.2 Примеры

Пример: покажем, как можно воспользоваться командой условного перехода для организации циклического процесса. Пусть на ленте имеется запись из нескольких меток подряд, и головка находится над самой крайней меткой справа. Требуется перевести головку влево до первой пустой позиции.

$1 \leftarrow 2$

2?3; 1

3!



Всего для машины Поста существует шесть типов команд:

→	Шаг вправо
←	Шаг влево
✓	Записать отметку
X	Стереть отметку
? <i>a; b</i>	Просмотреть ячейку; если в ячейке находится 0, то перейти на команду с номером <i>a</i> , иначе на команду с номером <i>b</i>
!	Останов

Пример: увеличить число 3 на единицу (изменить значение в памяти с 3 на 4). Допустим, точно известно, что каретка стоит где-то слева от меток и обозревает пустую ячейку. Тогда программа увеличения числа на единицу может выглядеть так:

```
1 -> 2
2 ? 1;3
3 <- 4
4 V 5
5 !
```

Пример: на ленте машины Поста расположен массив из n меток. Составить программу, действуя по которой машина выяснит, делится ли число n на 3. Если да, то после массива через одну пустую ячейку поставить метку.

```
1 → 2
2?3;4
3!
4 → 5
5?3;6
6 → 7
7&8;1
8 → 9
9V3
```

Пример: заикливание.

```
1β2
2,1
6 вариант
```

18.02.25

4 Машина Тьюрингита

4.1 Формальное описание машины Тьюрингита

4.2 Способы задания МТ

Граф переходов

4.3 Конфигурация МТ

Совокупность состояний ленты, указаний на ленте

Протоколы -

Выяснить, применимы ли программы к заданным состояниям машины Поста, указать результат работы машины Поста для каждого состояния. В начальный момент времени каретка оботрывает ячейку с самой левой меткой.

a)
 1 ? 5; 2 6 → 7 Начальное состояние ленты:
 2 → 1 7 ? 8, 9 1) $1^0 0^1 1^2$
 3 → 4 8 ? 2) $1^0 0^1 1^3$
 4 ? 6; 5 9 → 4 3) $10 [01]^2 1$
 5 ← 1

b)
 1 ? 4; 2 7 ∨ 8 Начальное состояние ленты:
 2 X 5 8 ← 9 1) $1^0 0^1$
 3 → 9 9 ? 11; 10 2) $1^0 0^1 1^2$
 4 ∨ 5 10 → 1 3) 1^4
 5 → 6 11 ?
 6 ? 7; 6

c)
 1 ? 4; 2 7 ← 8 Начальное состояние ленты:
 2 X 3 8 ? 9; 11 1) $10 1^2$
 3 → 6 9 ∨ 10 2) $1^0 0^1 1^3$
 4 ∨ 5 10 ← 1 3) $[10]^2 1$
 5 → 1 11 ?
 6 ? 4; 7

Ответ: 01010110

4.4 Приведений конфигураций к стандартному виду

4.5 Определение вычислимости по Тьюрингу

25.02.25

5 Принцип суперпозиции

6 Оператор примитивной рекурсии

окончание первой презентации тезис Черча
начало второй презентации

7 Алгоритм Маркова

8 Алгоритмически неразрешимые задачи

9 Основы задачи ...

25.03.25

10 NP

Задачи, которые нельзя отнести ни к классу P, ни к классу E.

Задачи, которые недетерминированная машина Тьюринга может решить за полиномиальное время, тогда как для детерминированной машины Тьюринга полиномиальный алгоритм неизвестен.

Для этих задач до сих пор не разработан эффективный (т.е. полиномиальный) алгоритм, но и не доказано, что таких алгоритмов не существует.

К классу NP относятся все задачи, решение которых можно проверить за полиномиальное время. Оракул предлагает решения, которые после проверки верификатором за полиномиальное время приобретают «юридическую» силу.

Пример

Дано n чисел a_1, \dots, a_n и число V .

Задача: Найти вектор (массив) $X = (x_1, \dots, x_n)$, $x_i \in \{0, 1\}$, такой, что $\sum a_i x_i = V$. Т.е. может ли быть представлено число V в виде суммы каких-либо элементов массива A .

Если какой-то алгоритм выдает результат – массив X , то проверка правильности этого результата может быть выполнена с полиномиальной сложностью: проверка

$\sum a_i x_i = V$ требует не более $O(N)$ операций.

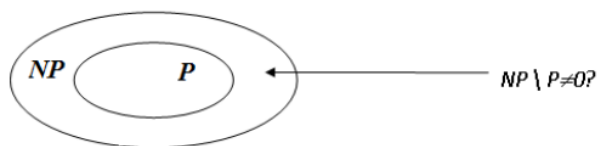
10.1 Проблема равенства классов P и NP

Поскольку детерминированная машина Тьюринга может рассматриваться как специальный случай недетерминированной машины Тьюринга, в которой отсутствует стадия угадывания, а стадия проверки совпадает с ДМТ, класс NP включает в себя класс P, а также некоторые проблемы, для решения которых известны лишь алгоритмы, экспоненциально зависящие от размера входа (то есть неэффективные для больших входов).

Вопрос о равенстве этих двух классов считается одной из самых сложных открытых проблем в области теоретической информатики.

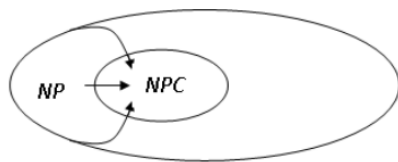
На сегодня отсутствуют теоретические доказательства как совпадения этих классов ($P=NP$), так и их несовпадения.

Предположение состоит в том, что класс P является собственным подмножеством класса NP, т.е. $NP \setminus P \neq \emptyset$.



10.2 Класс NPC (NP – полные задачи)

Определение класса NPC (NP-complete) или класса NP-полных задач требует выполнения следующих двух условий: во-первых, задача должна принадлежать классу NP, и, во-вторых, к ней полиномиально должны сводиться все задачи из класса NP.



Полиномиальная сводимость

Сведение по Карпу

....

Сведение по Куку

....

Для класса NPC доказана следующая теорема: если существует задача, принадлежащая классу NPC, для которой существует полиномиальный

алгоритм решения, то класс P совпадает с классом NP, т.е. $P=NP$. В настоящее время доказано существование сотен NP-полных задач, но ни для одной из них пока не удалось найти полиномиального алгоритма решения. В настоящее время исследователи предполагают следующее соотношение классов:

тут должна быть картинка

01.04.25

11 Вычисление времени выполнения нерекурсивных алгоритмов

11.1 Нахождение функции трудоемкости по фактическому количеству элементарных операций

В качестве «элементарных» операций алгоритма, представленного в формальной системе RAM будем использовать следующие: 1) простое присваивание: $a \leftarrow b$; 2) одномерная индексация $a[i]$; 3) арифметические операции: $(*, /, -, +)$; 4) операции сравнения; 5) логические операции.

Анализ трудоемкости основных алгоритмических конструкций

А) Конструкция «Следование»




Трудоемкость конструкции есть сумма трудоемкостей блоков, следующих друг за другом.


$T_{\text{«следование»}} = t_1 + \dots + t_k$, где k – количество блоков.

Б) Конструкция «Ветвление»

if (logical expression) then

 t_{then} с вероятностью p

else

 t_{else} с вероятностью $(1-p)$

Общая трудоемкость конструкции «Ветвление» требует анализа вероятности выполнения переходов на блоки «Then» и «Else» и определяется как:

$T_{\text{«ветвление»}} = t_{\text{logical expression}} + t_{\text{then}} * p + t_{\text{else}} * (1-p)$.