

Теория алгоритмов

Лекции

Владимир Анатольевич Судаков

2026 г.

Материалы

- Судаков В.А. <https://github.com/sudakov/algo-lab> - будет дополняться и модифицироваться.
- Клейнберг Дж., Тардос Е. Алгоритмы. Разработка и применение. - Санкт-Петербург: Питер, 2016. - 800 с.: ил. - (Классика Computers Science).
- Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2016. – 528 с
- Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. - М.: Мир, 1982.
- Пенроуз Р. Новый ум короля. — М.: Еditorial УРСС, 2003.

Алгоритм

- Алгоритм — это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными характеристиками:
 - конечность,
 - определённость,
 - ввод
 - вывод,
 - эффективность.

Другие определения

- Алгоритм — это всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи». (А.Н. Колмогоров)
- Алгоритм — это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату (А.А. Марков)

Все ли задачи можно решить с помощью компьютера?

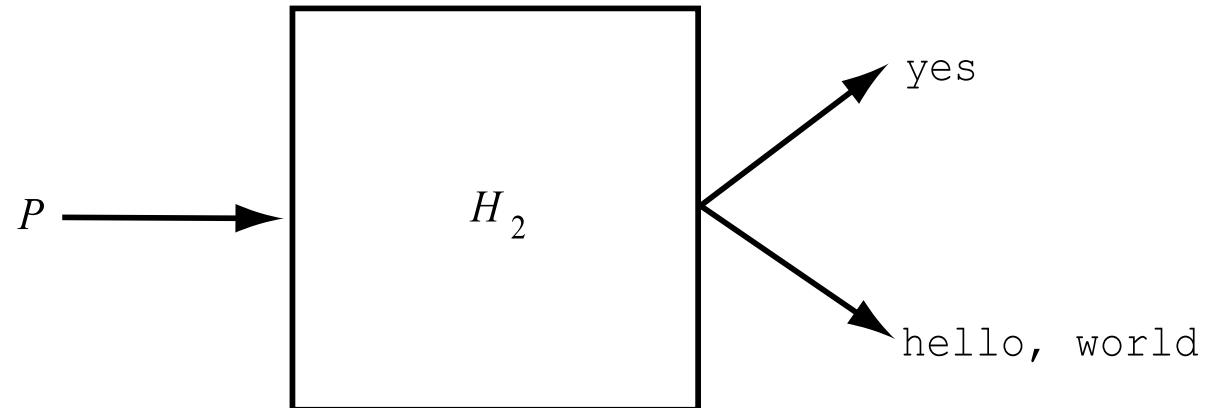
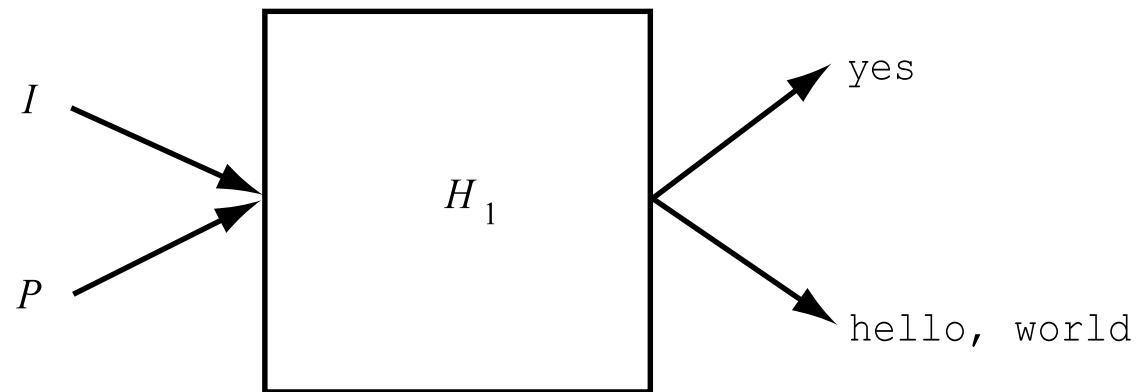
```
n=gets.to_i
t = 3
loop do
    for x in (1..t-2)
        for y in (1..t-x-1)
            z = t-x-y
            if x**n + y**n == z**n
                puts 'hello, world'
                exit
            end
        end
    end
    t += 1
end
```

- Напечатает ли данная программа hello, word?

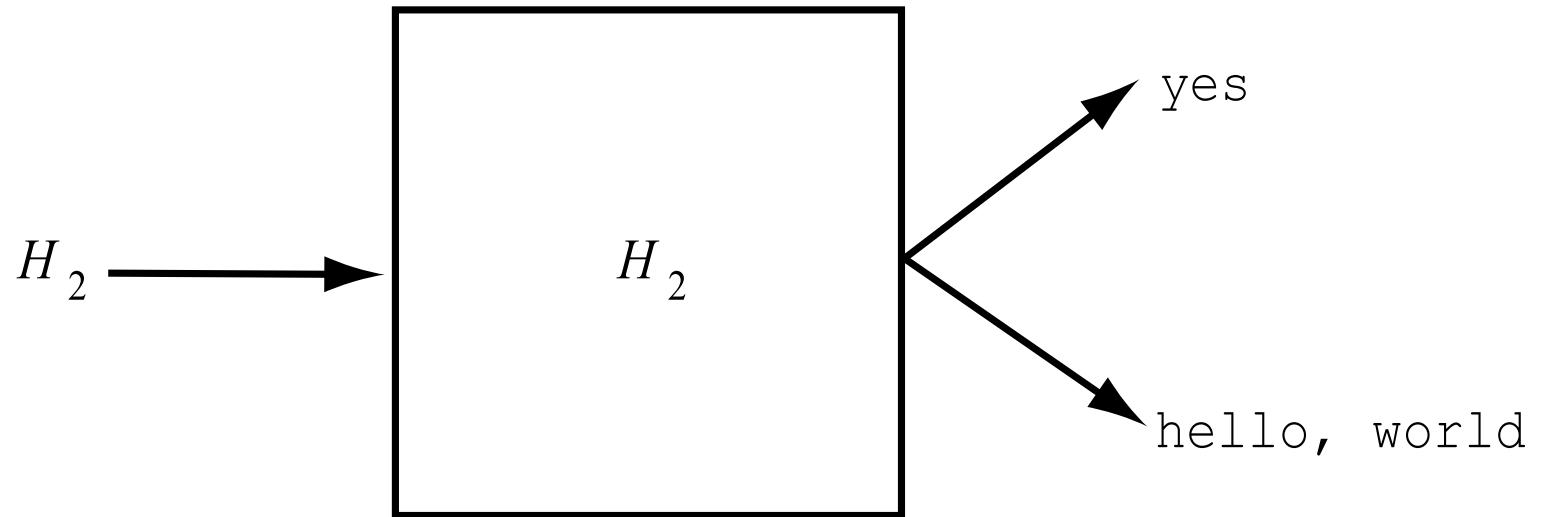
Доказательство Hello, world



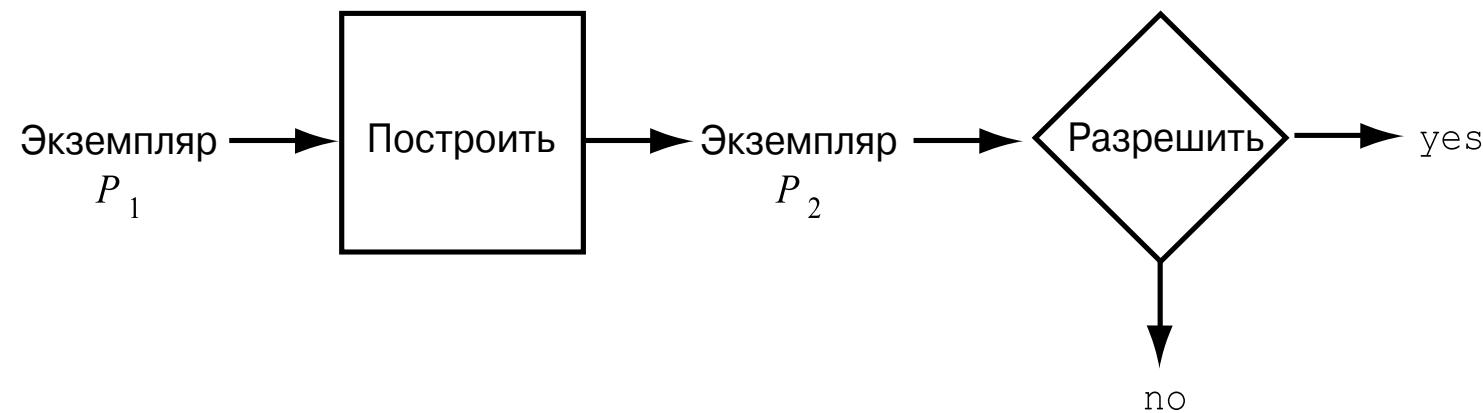
Hello, world (2)



Hello, world (3)



Сведение программы



Программа вызова функции `foo()`

Дана программа Q и ее вход u , необходимо построить программу R и ее вход z так, чтобы R со входом z вызывала foo тогда и только тогда, когда Q со входом u печатает `hello, world`:

1. Если Q содержит `foo()`, переименовать ее и все ее вызовы. Очевидно, новая программа Q_1 выполняет то же самое, что и Q .
2. Добавить в Q_1 функцию `foo()`. Эта функция не делает ничего и не вызывается. Полученная программа называется Q_2 .
3. Изменить Q_2 так, чтобы первые 12 символов ее печати запоминались в глобальном массиве A . Пусть полученная программа называется Q_3 .
4. Изменить Q_3 так, чтобы после каждого выполнения инструкции печати с использованием массива A проверялось, не образуют ли первые 12 напечатанных символов текст `hello, world`. В этом случае вызвать новую функцию `foo`, добавленную в п. 2. Полученная программа есть R , а ее вход z совпадает с u .

Проблема математического доказательства

- В начале XX века Д. Гильберт поставил вопрос о поисках алгоритма, который позволял бы определить истинность или ложность любого математического утверждения.
- В 1931 г. К. Гедель опубликовал свою знаменитую теорему о неполноте. Он доказал, что существует истинная формула первого порядка с целыми, которую нельзя ни доказать, ни опровергнуть в исчислении предикатов первого порядка над целыми.

Теорема о неполноте

- Перенумеруем все утверждения арифметики в лексикографическом порядке
- Пусть n -я (из пронумерованных выбранным способом строк символов) такая функция от аргумента ω обозначается:

$$P_n(\omega)$$

- Запишем предикат:

$$\neg \exists x [\Pi_x \text{ доказывает } P_\omega(\omega)] = P_k(\omega)$$

- Подставим $\omega=k$

$$\neg \exists x [\Pi_x \text{ доказывает } P_k(k)] = P_k(k)$$

Парадокс Рассела

- Назовем множество «обычным», если оно не является своим собственным элементом.
- Примером «необычного» множества является множество всех множеств, так как оно само является множеством, а следовательно, само является собственным элементом.
- Рассмотрим множество, состоящее только из всех «обычных» множеств – **«расселовское множество»**.

Содержит ли расселовское множество себя?

- Пусть в некой деревне живёт брадобрей, который бреет всех жителей деревни, которые не бреются сами, и только их. Бреет ли брадобрей сам себя?

Неразрешимые проблемы

- Проблема останова
- Не существует алгоритма, который узнавал бы по произвольному диофантову уравнению, имеет ли оно решения в целых числах или нет.
- Проблема соответствий Поста (ПСП):

Дано 2 списка

$$A = w_1, w_2, \dots, w_k \quad \text{и} \quad B = x_1, x_2, \dots, x_k$$

Существует ли последовательность индексов

i_1, i_2, \dots, i_m такая, что

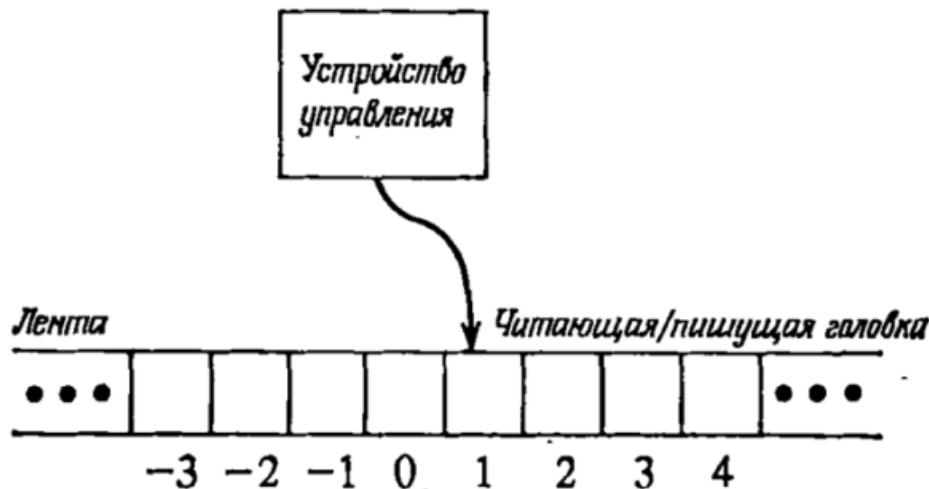
$$w_{i_1} w_{i_2} \dots w_{i_m} = x_{i_1}, x_{i_2}, \dots, x_{i_m}$$

Пример ПСП

	Список <i>A</i>	Список <i>B</i>
<i>i</i>	w_i	x_i
1	1	111
2	10111	10
3	10	0

$$w_2 w_1 w_1 w_3 = x_2 x_1 x_1 x_3 = 10111110$$

Машина Тьюринга (ДМТ)



- Γ – множество ленточных символов,
- $\Sigma \subset \Gamma$ – входные символы,
- $b \in \Gamma \setminus \Sigma$ – пустой символ,
- Q – множество состояний. q_0 – начальное состояние, $\{q_y, q_n\}$ – конечные состояния
- Функция перехода:

$$\delta: (Q \setminus \{q_y, q_n\}) \times \Gamma \rightarrow Q \times \Gamma \times \{-1, 1\}$$

Пример машины Тьюринга

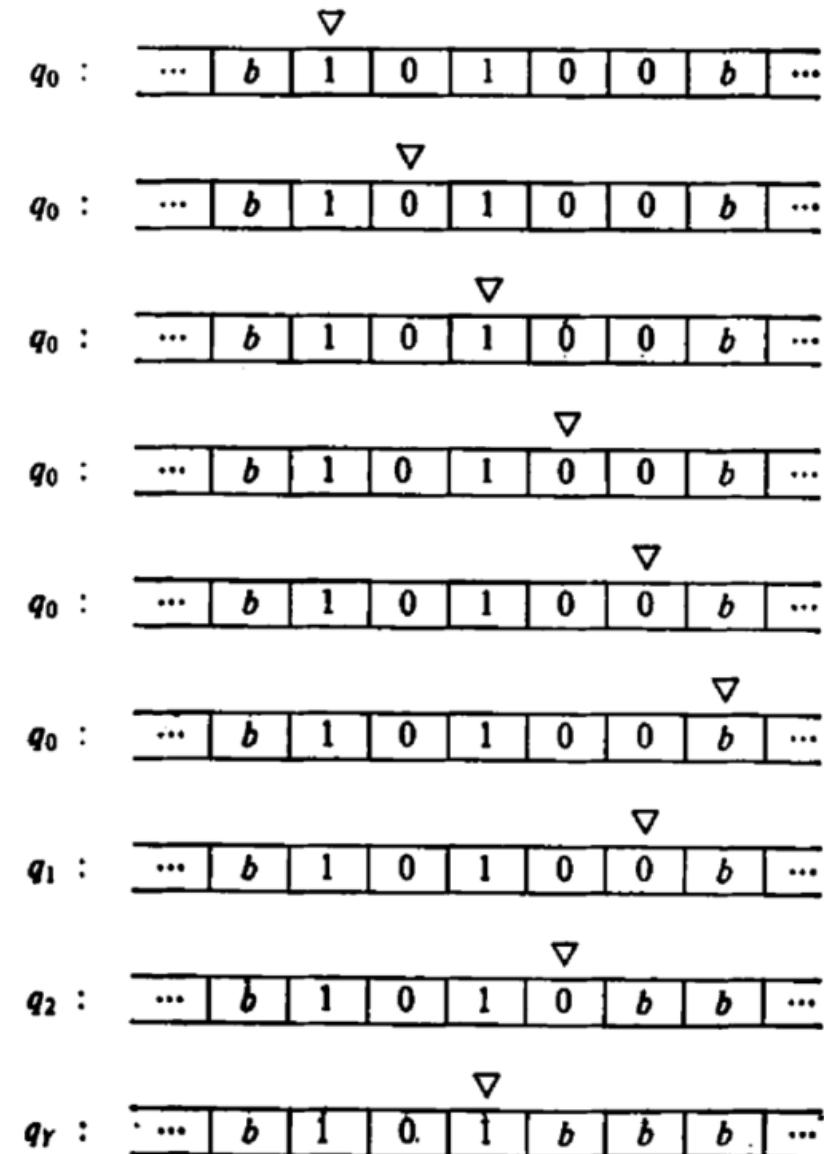
Делимость на 4

$$\Gamma = \{0, 1, b\}, \Sigma = \{0, 1\}$$

$$Q = \{q_0, q_1, q_2, q_3, q_r, q_N\}$$

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_r, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$$\delta(q, s)$$



Рекурсивные функции

К числу базовых примитивно рекурсивных функций относятся функции следующих трёх видов:

- *Нулевая функция* O — функция без аргументов, всегда возвращающая $\underline{0}$.
- *Функция следования* S одного переменного, сопоставляющая любому натуральному числу x непосредственно следующее за ним натуральное число $x + 1$.
- Функции I_n^m , где $0 < m \leq n$, от n переменных, сопоставляющие любому упорядоченному набору x_1, \dots, x_n натуральных чисел число x_m из этого набора.

Операторы подстановки и примитивной рекурсии определяются следующим образом:

- **Оператор суперпозиции** (иногда — *оператор подстановки*). Пусть f — функция от m переменных, а g_1, \dots, g_m — упорядоченный набор функций от n переменных каждая. Тогда результатом суперпозиции функций g_k в функцию f называется функция h от n переменных, сопоставляющая любому упорядоченному набору x_1, \dots, x_n натуральных чисел число

$$h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n)) .$$

- **Оператор примитивной рекурсии.** Пусть f — функция от n переменных, а g — функция от $n + 2$ переменных. Тогда результатом применения оператора примитивной рекурсии к паре функций f и g называется функция h от $n + 1$ переменной вида

$$\begin{aligned} h(x_1, \dots, x_n, 0) &= f(x_1, \dots, x_n) ; \\ h(x_1, \dots, x_n, y + 1) &= g(x_1, \dots, x_n, y, h(x_1, \dots, x_n, y)) . \end{aligned}$$

Рекурсивные функции (2)

$$Sum(x, 0) = I_1^1(x);$$

$$Sum(x, y + 1) = F(x, y, Sum(x, y));$$

$$F(x, y, z) = S(I_3^3(x, y, z)).$$

$$Mul(x, 0) = O(x);$$

$$Mul(x, y + 1) = G(x, y, Mul(x, y));$$

$$G(x, y, z) = Sum(I_3^1(x, y, z), I_3^3(x, y, z)).$$

Рекурсивные функции (3)

Частично рекурсивная функция определяется аналогично примитивно рекурсивной, только к двум операторам суперпозиции и примитивной рекурсии добавляется ещё третий оператор — минимизации аргумента.

- **Оператор минимизации аргумента.** Пусть f — функция от n натуральных переменных. Тогда результатом применения оператора минимума аргумента к функции f называется функция h от $n - 1$ переменной, задаваемая следующим определением:

$$h(x_1, \dots, x_{n-1}) = \min y, \text{ при условии } f(x_1, \dots, x_{n-1}, y) = 0$$

То есть функция h возвращает минимальное значение последнего аргумента функции f , при котором её значение равно 0.

Частично рекурсивные функции для некоторых значений аргумента могут быть не определены, так как оператор минимизации аргумента не всегда корректно определён, поскольку функция f может быть не равной нулю ни при каких значениях аргументов. С точки зрения императивного программирования, результатом частично рекурсивной функции может быть не только число, но и исключение или уход в бесконечный цикл, соответствующие неопределённому значению.

Общерекурсивная функция

Общерекурсивная функция — частично рекурсивная функция, определённая для всех значений аргументов. Задача определения того, является ли частично рекурсивная функция с данным описанием общерекурсивной или нет, алгоритмически неразрешима.

Тезис Черча- Тьюринга

Эвристическое утверждение:

- интуитивное понятие алгоритмической вычислимости

и строго формализованные понятия:

- частично рекурсивной функции,
- функции, вычислимой на машине Тьюринга

эквивалентны.

- Любая функция, которая может быть вычислена физическим устройством, может быть вычислена машиной Тьюринга
- Функции, вычислимые посредством алгоритмов, являются частично рекурсивными

Временная сложность алгоритма

- n - объем входных данных. Размер цепочки символов в заданной схеме кодирования.
- $f(n)$ – временная сложность – затраты времени. Максимальное (во всем индивидуальным задачам) время, затрачиваемое алгоритмом на решение индивидуальной задачи длины n .
- Функция не будет определена полностью, пока не будет выбрана схема кодирования и вычислительное устройство. Но эти детали оказывают несущественное влияние на класс труднорешаемых задач.
- Будем считать число шагов машины Тьюринга выполненных до момента остановки

Полиномиальная и экспоненциальная сложность

- $f(n)$ есть $O(g(n))$, если $\exists c : |f(n)| \leq c|g(n)|$
- Полиномиальный алгоритм (алгоритм полиномиальной временной сложности) – это алгоритм для которого временная сложность есть:

$O(p(n))$, где $p(n)$ – некоторая полиномиальная функция, а n – входная длина.

- Алгоритмы не поддающиеся такой оценки будем называть экспоненциальными (трудноразрешимыми).

Сравнение сложности

Время выполнения алгоритма с
определенной сложностью в зависимости от
размера входных данных при скорости
 10^6 операций в секунду:

размер сложность	10	20	30	40	50	60
n	0,00001 сек.	0,00002 сек.	0,00003 сек.	0,00004 сек.	0,00005 сек.	0,00005 сек.
n^2	0,0001 сек.	0,0004 сек.	0,0009 сек.	0,0016 сек.	0,0025 сек.	0,0036 сек.
n^3	0,001 сек.	0,008 сек.	0,027 сек.	0,064 сек.	0,125 сек.	0,216 сек.
n^5	0,1 сек.	3,2 сек.	24,3 сек.	1,7 минут	5,2 минут	13 минут
2^n	0,0001 сек.	1 сек.	17,9 минут	12,7 дней	35,7 веков	366 веков
3^n	0,059 сек.	58 минут	6,5 лет	3855 веков	2x10 ⁸ веков	1,3x10 ¹³ веков

Влияние быстродействия ЭВМ

*Размеры наибольшей задачи,
разрешимой за один час*

<i>Функция временной сложности</i>	<i>На современных ЭВМ</i>	<i>На ЭВМ, в 100 раз более быстрых</i>	<i>На ЭВМ, в 1000 раз более быстрых</i>
n	N_1	$100 N_1$	$1000 N_1$
n^2	N_2	$10 N_2$	$31,6 N_2$
n^3	N_3	$4,64 N_3$	$10 N_3$
n^5	N_4	$2,5 N_4$	$3,98 N_4$
2^n	N_5	$N_5 + 6,64$	$N_5 + 9,97$
3^n	N_6	$N_6 + 4,19$	$N_6 + 6,29$

Экспоненциальная сложность на практике

- Некоторые задачи имеют экспоненциальную сложность, но это сложность для *наихудшего* случая
- Во многих практических задачах эта сложность не проявляется
- Проблема – нет способа спрогнозировать как поведет себя экспоненциальный алгоритм на тех или иных данных

Теория NP-полных задач

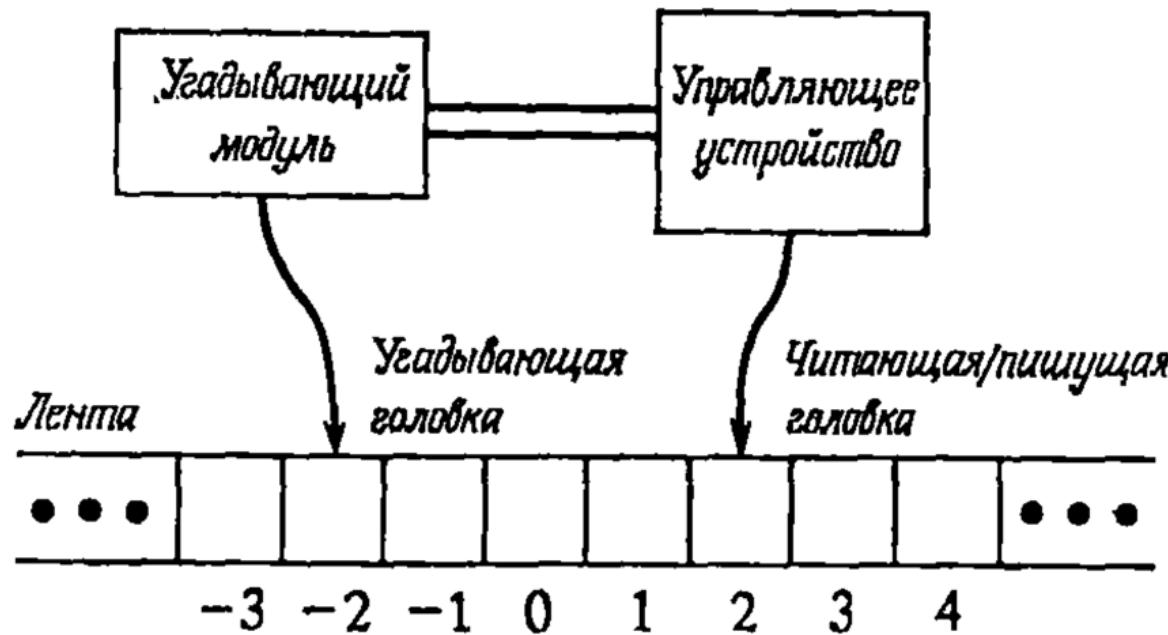
- Рассматриваются задачи распознавания:
 - Принадлежит ли цепочка символов заданному языку?
- Примеры задач распознавания:
 - Изоморфизм подграфу: Верно ли что граф G_1 содержит подграф изоморфный графу G_2 ?
 - Коммивояжер: Существует ли в графе путь через все города длина которого не превосходит B ?
- Задачи оптимизации не проще соответствующих задач распознавания.
- Многие задачи распознавания не проще соответствующих задач оптимизации

Задачи и языки

- Задача (массовая задача) Π – общий вопрос, на который следует дать ответ. Задача содержит:
 - список параметров;
 - формулировка тех свойств, которым должен удовлетворять ответ.
- Индивидуальная задача I – получается из Π путем приписывания всем параметрам конкретных значений.
- Задача распознавания свойств – Π состоит из индивидуальных задач D_Π и $Y_\Pi \subseteq D_\Pi$.
- Σ – алфавит. Σ^* – множество конкретных цепочек составленных из символа алфавита.
- Язык в алфавите Σ – это любое подмножество $L \subseteq \Sigma^*$.
- Задача Π и схема кодирования e разбивают Σ^* на 3 класса:
 - слова не являющиеся кодами индивидуальных задач из Π ;
 - слова являющиеся кодами индивидуальных задач из Π с отрицательным ответом на вопрос;
 - слова являющиеся кодами индивидуальных задач из Π с положительным ответом на вопрос (это язык L):

$$L[\Pi, e] = \left\{ x \in \Sigma^*: \begin{array}{l} \text{есть алфавит схемы кодирования } e, \\ a \ x - \text{код индивидуальной задачи} \\ I \in Y_\Pi \text{ при схеме кодирования } e \end{array} \right\}.$$

Недетерминированная машина Тьюринга (НДМТ)

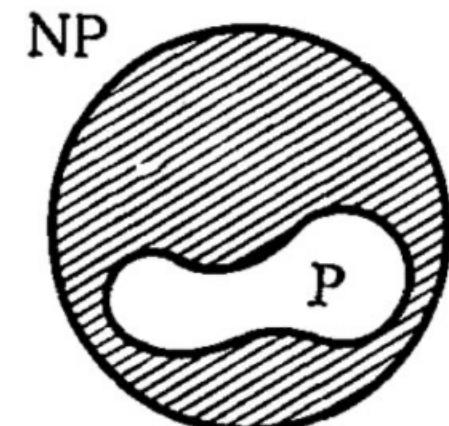


2 стадии:

- Угадывание. Угадывающий модуль записывает догадку
- Проверка. Проверяется догадка так же как и для детерминированной машины Тьюринга

Взаимоотношения классов P и NP

- Класс P – задачи, которые могут быть решены ДМТ за полиномиальное время
- Класс NP - задачи, которые могут быть решены НДМТ за полиномиальное время
- $P \subseteq NP$. Любая задача класса P может быть решена за полиномиальное время на НДМТ.
- $P \neq NP ?$



Сводимость задач

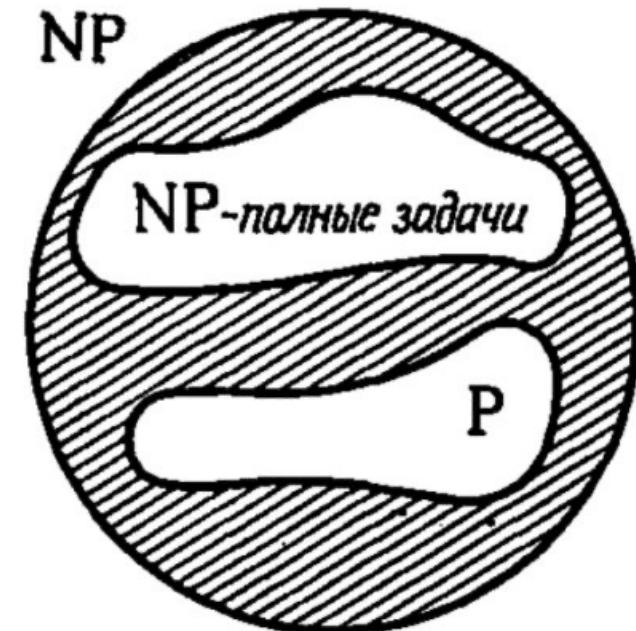
- Имеет место полиномиальная сводимость языка $L_1 \subseteq \Sigma^*_1$ к языку $L_2 \subseteq \Sigma^*_2$, если существует функция $f: \Sigma^*_1 \rightarrow \Sigma^*_2$ удовлетворяющая условиям:
 - существует ДМТ вычисляющая f с полиномиальной временной сложностью;
 - $(\forall x \in \Sigma^*_1) x \in L_1 \leftrightarrow f(x) \in L_2$
- $L_1 \propto L_2$
- Пример: Задача поиска гамильтонова цикла (ГЦ) сводится к задаче Коммивояжёр (КМ):
 - $G=(V,E)$, $|V|=m$,
 - $d(v_i, v_j)=1$ если $(v_i, v_j) \in E$, иначе $d(v_i, v_j)=2$
 - Существует ли путь длиной $\leq m$?
- ГЦ \propto КМ

NP-полные задачи

- Язык L называется NP-полным, если $L \in NP$ и любой другой язык $L' \in NP$ сводится к L :

$$L' \leq L$$

- Если хотя бы одна NP-полная задача может быть решена за полиномиальное время, то и все NP задачи могут быть решены за полиномиальное время.
- Если L_1 NP-полный язык, $L_2 \in NP$ и $L_1 \leq L_2$, то L_2 NP-полный язык
- Чтобы доказать, что Π является NP-полной задачей достаточно доказать:
 - $\Pi \in NP$
 - Какая-то известная NP-полная задача $\Pi' \leq \Pi$



Теорема Кука

- Задача выполнимость (ВЫП):
- Дано множество булевых переменных:

$$U = \{u_1, u_2, \dots, u_m\}$$

- Дано ФАЛ $f(u_1, u_2, \dots, u_m)$ в форме КНФ
- Вопрос: выполнима ли f ?
- Теорема Кука: ВЫП есть NP-полная задача

Доказательство теоремы Кука

- ВЫП – принадлежит классу NP. Так как НДМТ достаточно указать набор значений u_1, u_2, \dots, u_m и далее проверить $f(u_1, u_2, \dots, u_m) = 1$ за полиномиальное время
- Далее нужно показать $\forall L \in NP \propto L_{\text{вып}}$:
- НДМТ М: $\{\Gamma, \Sigma, b, Q, q_0, q_y, q_n, \delta\}$
- n – размер входа
- $p(n)$ полином ограничивающий временную сложность $T_M(n)$
- Нужно найти ФАЛ f_L , реализующую полиномиальную сводимость

Переменные f_L

Переменная	Пределы изменения индексов	Придаваемый смысл
$Q[i, k]$	$0 \leq i \leq p(n)$ $0 \leq k \leq r$	В момент времени i программа M находится в состоянии q_k
$H[i, j]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$	В момент времени i читающая/пишущая головка просматривает ячейку с номером j
$S[i, j, k]$	$0 \leq i \leq p(n)$ $-p(n) \leq j \leq p(n) + 1$ $0 \leq k \leq v$	В момент времени i в j -й ячейке записан символ s_k

Состав f_L

Группа
дизъюнкций

Налагаемое ограничение

-
- | | |
|-------|---|
| G_1 | В любой момент времени i программа M находится ровно в одном состоянии. |
| G_2 | В любой момент времени i читающая/пишущая головка просматривает ровно одну ячейку. |
| G_3 | В любой момент времени i каждая ячейка содержит ровно один символ из Γ . |
| G_4 | В момент времени 0 вычисление находится в исходной конфигурации стадии проверки при входе x . |
| G_5 | Не позднее чем через $p(n)$ шагов M переходит в состояние q_y и, следовательно, принимает x . |
| G_6 | Для любого момента времени i , $0 \leq i \leq p(n)$, конфигурация программы M в момент времени $i + 1$ получается из конфигурации в момент времени i одноразовым применением функции перехода δ . |
-

ДИЗЬЮНКЦИИ ДЛЯ G_1 - G_5

- G_1 $\{Q[i, 0], Q[i, 1], \dots, Q[i, r]\}, \quad 0 \leq i \leq p(n),$
 $\{\overline{Q[i, j]}, \overline{Q[i, j']}\}, \quad 0 \leq i \leq p(n), \quad 0 \leq j < j' \leq r,$
- G_2 $\{H[i, -p(n)], H[i, -p(n)+1], \dots, H[i, p(n)+1]\}, \quad 0 \leq i \leq p(n),$
 $\{\overline{H[i, j]}, \overline{H[i, j']}\}, \quad 0 \leq i \leq p(n), \quad -p(n) \leq j < j' \leq p(n)+1$
- G_3 $\{S[i, j, 0], S[i, j, 1], \dots, S[i, j, v]\}, \quad 0 \leq i \leq p(n),$
 $\quad \quad \quad -p(n) \leq j \leq p(n)+1,$
 $\{\overline{S[i, j, k]}, \overline{S[i, j, k']}\}, \quad 0 \leq i \leq p(n), \quad -p(n) \leq j \leq p(n)+1,$
 $\quad \quad \quad 0 \leq k < k' \leq v,$
- G_4 $\{Q[0, 0]\}, \{H[0, 1]\}, \{S[0, 0, 0]\},$
 $\{S[0, 1, k_1]\}, \{S[0, 2, k_2]\}, \dots, \{S[0, n, k_n]\},$
 $\{S[0, n+1, 0]\}, \{S[0, n+2, 0]\}, \dots, \{S[0, p(n)+1, 0]\},$
 $\text{где } x = s_{k_1} s_{k_2} \dots s_{k_n},$
- G_5 $\{Q[p(n), 1]\}.$

G₆

- Если головка не просматривает ячейку j в момент i, то значение в ячейке j не изменится к моменту i+1

$$\{\overline{S[i, \ j, \ l]}, \ H[i, \ j], \ S[i + 1, \ j, \ l]\}, \ 0 \leq i \leq p(n), \\ -p(n) \leq j \leq p(n) + 1, \ 0 \leq l \leq v.$$

- Перестройка конфигурации происходит согласно функции δ:

$$(i, \ j, \ k, \ l), \ 0 \leq i < p(n), \ -p(n) \leq j \leq p(n) + 1, \\ 0 \leq k \leq r \text{ и } 0 \leq l \leq v,$$

$$\{H[i, \ j], \ Q[i, \ k], \ \overline{S[i, \ j, \ l]}, \ H[i + 1, \ j + \Delta]\},$$

$$\{H[i, \ j], \ \overline{Q[i, \ k]}, \ \overline{S[i, \ j, \ l]}, \ Q[i + 1, \ k']\},$$

$$\{H[i, \ j], \ \overline{Q[i, \ k]}, \ \overline{S[i, \ j, \ l]}, \ S[i + 1, \ j, \ l']\},$$

если $q_k \in Q \setminus \{q_Y, q_N\}$, то $\delta(q_k, s_l) = (q_{k'}, s_{l'}, \Delta)$,

а если $q_k \in \{q_Y, q_N\}$, то $\Delta = 0$, $k' = k$ и $l' = l$.