

Элементы теории алгоритмов

Уточнение понятия алгоритма

Зачем уточнять определение?

Алгоритм – точный **набор инструкций** для **исполнителя**.



Всегда ли существует алгоритм?

Конструктивное доказательство: построить алгоритм.



А если не удалось?

- задача о квадратуре круга
- задача о трисекции угла
- задача об удвоении куба
- вечный двигатель
- ...

нестрогие
понятия



Как доказать, что алгоритма не существует?

Зачем уточнять определение?

Задача: алгоритм как математический объект.

Теория алгоритмов (1930-е):

- доказательство алгоритмической неразрешимости задач
- анализ сложности алгоритмов
- сравнительная оценка качества алгоритмов



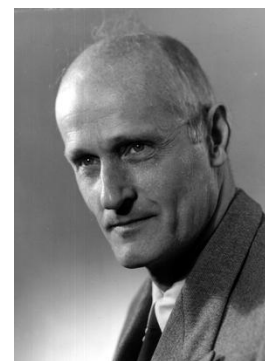
А. Тьюринг



Э. Пост



А. Чёрч



С. Клини



А. Марков

Что такое алгоритм?

Первые алгоритмы – правила арифметических действий:

- объекты – числа
- шаги – операции с однозначными числами



Что считать шагом?

Все объекты можно закодировать как символьные строки:



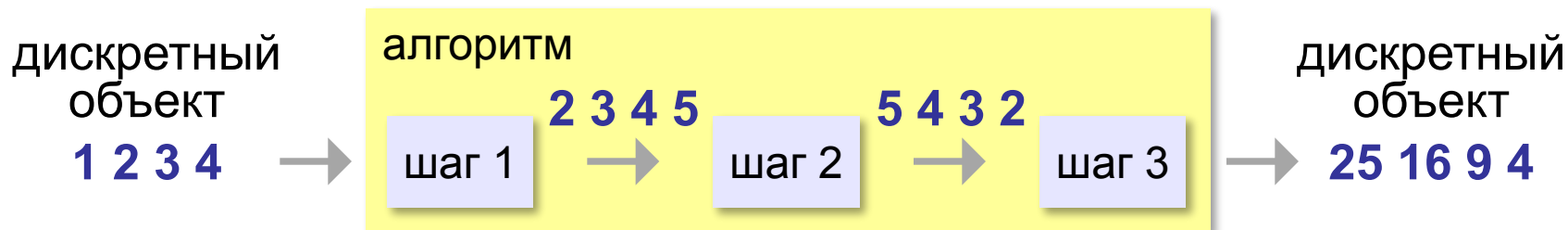
Можно рассматривать только алгоритмы обработки **строк**!

Из любого кода можно перевести в двоичный:



Можно рассматривать только алгоритмы обработки **битовых** строк!

Как работает алгоритм?



- получает на вход дискретный объект
- в результате строит другой дискретный объект (или выдаёт сообщение об ошибке)
- обрабатывает объект по шагам
- на каждом шаге получается новый дискретный объект

Как работает алгоритм?



Любой алгоритм определяет функцию!

т.е. правило преобразования входа в выход

Функция не определена $\Leftrightarrow$ алгоритм зацикливается или завершается аварийно.

ввод a, b
вывод $a * \text{sqrt}(b)$

→ $\times b < 0$

ввод a
нц пока да
кц

→ $\times$ для всех a

Эквивалентные алгоритмы

Задают одну и ту же функцию:

если $a < b$ то

$M := a$

иначе

$M := b$

все



$M := b$

если $a < b$ то

$M := a$

все

Универсальные исполнители

Алгоритм привязан к исполнителю $\Rightarrow$ идея: построить универсального исполнителя.

Для любого алгоритма для любого исполнителя можно построить эквивалентный алгоритм для **универсального исполнителя**.

- если есть алгоритм для универсального исполнителя, то задача разрешима
- если доказано, что нет алгоритма для универсального исполнителя, задача неразрешима



Любой алгоритм может быть представлен как программа для универсального исполнителя!

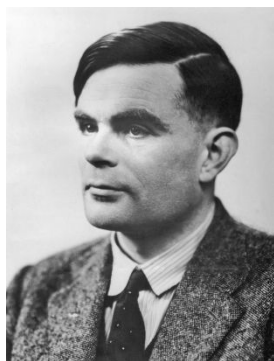
Универсальные исполнители

Алгоритм – это программа для универсального исполнителя.

Модель вычислений:

- *«процессор»* (система команд и способ их выполнения)
- *«память»* (способ хранения данных)
- *язык программирования* (способ записи программ);
- *способ ввода данных*
- *способ вывода результата*

Универсальные исполнители



А. Тьюринг

**машина
Тьюринга**



Э. Пост

**машина
Поста**



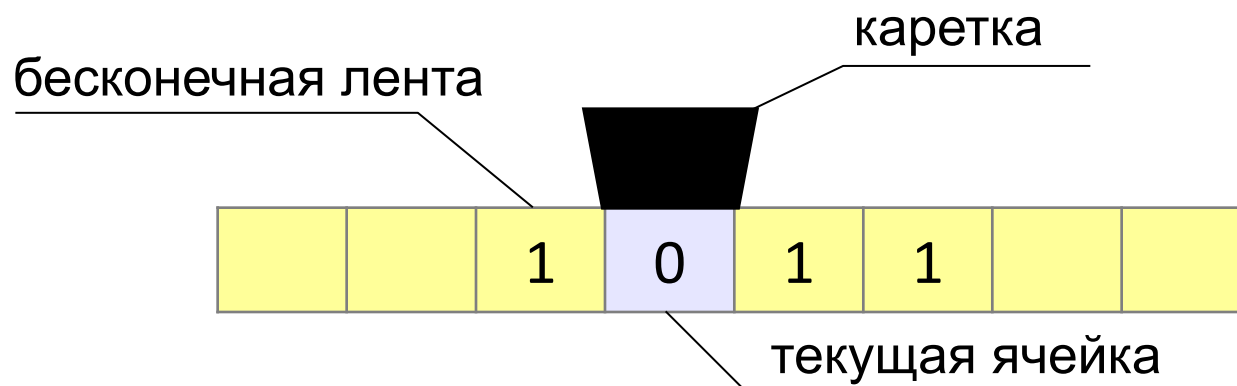
А. Марков

**нормальные
алгоритмы
Маркова**



Все универсальные исполнители эквивалентны!

Машина Тьюринга



А. Тьюринг

- бесконечная лента («**память**»)
- каретка (**запись и чтение**)
- программируемый автомат («**процессор**»)

алфавит: $A = \{a_1, a_2, \dots, a_N\}$ $A = \{0, 1, \square\}$

пробел

Что такое автомат?

Автомат – это устройство, работающее без участия человека.

Состояние – промежуточная задача, которую решает автомат.

$$Q = \{q_1, q_2, \dots, q_M\}$$

начальное
состояние

q_0 – остановка автомата

Программа для машины Тьюринга

Программа состоит из команд:

- записать символ a_i в текущую ячейку
- переместить каретку $\rightarrow \leftarrow$ • (не перемещать)
- перейти в состояние q_j

$$A = \{0, 1, \square\}$$

$$1 \rightarrow q_1$$

- записать 1
- переместиться вправо
- перейти в состояние q_1

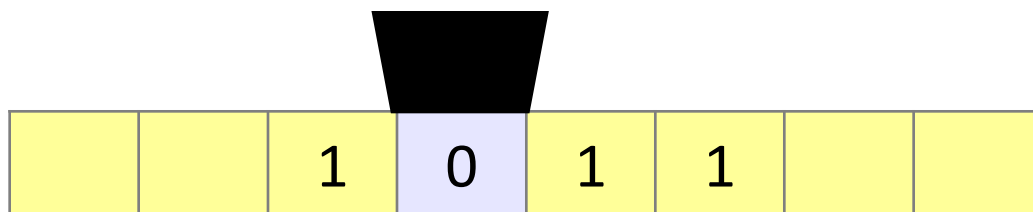
$$0 \cdot q_0$$

- записать 0
- не перемещать каретку
- останов (q_0)

Программа для машины Тьюринга

Задача. На ленте записано число в двоичной системе счисления. Каретка находится где-то над числом. Требуется увеличить число на единицу.

алфавит: $A = \{0, 1, \square\}$



СОСТОЯНИЯ: q_1 – поиск правого конца слова

подзадачи q_2 – увеличение числа на 1

Программа для машины Тьюринга

q_1 : поиск конца слова

- если 0, то $\rightarrow$
- если 1, то $\rightarrow$
- если $\square$, то $\leftarrow$ и **переход в q_2**

ТОЛЬКО
изменения!

	q_1
0	$\rightarrow$
1	$\rightarrow$
$\square$	$\leftarrow q_2$

q_2 : увеличение числа на 1

- если 0, то записать 1 и стоп (q_0)
- если 1, то записать 0 и $\leftarrow$
- если $\square$, то записать 1 и стоп (q_0)

	q_2
0	$1 \cdot q_0$
1	$0 \leftarrow$
$\square$	$1 \cdot q_0$



Как объединить две программы?

Программа для машины Тьюринга

	q_1
0	$\rightarrow$
1	$\rightarrow$
$\square$	$\leftarrow q_2$

	q_2
0	$1 \cdot q_1$
1	$0 \cdot q_1$
$\square$	$1 \cdot q_0$



Связь подзадач через ячейку ($\square, q_1$)!

Если алгоритмы А и Б можно запрограммировать на машине Тьюринга, то и любую их комбинацию тоже можно запрограммировать.

Тезис Чёрча-Тьюринга: Любой алгоритм (в интуитивном смысле этого слова) может быть представлен как программа для машины Тьюринга.

Программа для машины Тьюринга

начальное
состояние

новая
метка

переход

новое
состояние

(0 , q_1 , 0 , $\rightarrow$, q_1)

(1 , q_1 , 1 , $\rightarrow$, q_1)

($\square$, q_1 , $\square$, $\leftarrow$, q_2)

(0 , q_2 , 1 , $\bullet$, q_0)

(1 , q_2 , 0 , $\leftarrow$, q_2)

($\square$, q_2 , 1 , $\bullet$, q_0)

	q_1
0	$0 \rightarrow q_1$
1	
$\square$	

	q_2
0	$1 \bullet q_0$
1	$0 \leftarrow q_2$
$\square$	$1 \bullet q_0$

Программы для машины Тьюринга

	q_1
0	$\leftarrow$
1	$\leftarrow$
$\square$	$\rightarrow q_0$

	q_1
0	$\rightarrow q_0$
1	$\rightarrow q_0$
$\square$	$\leftarrow$

	q_1	q_2
0	q_2	$\square \leftarrow$
1	q_2	$\square \leftarrow$
$\square$	$\leftarrow$	q_0

? Что делает программа?

? Когда зацикливается?

Программы для машины Тьюринга

Задача 1. Уменьшить двоичное число на 1.

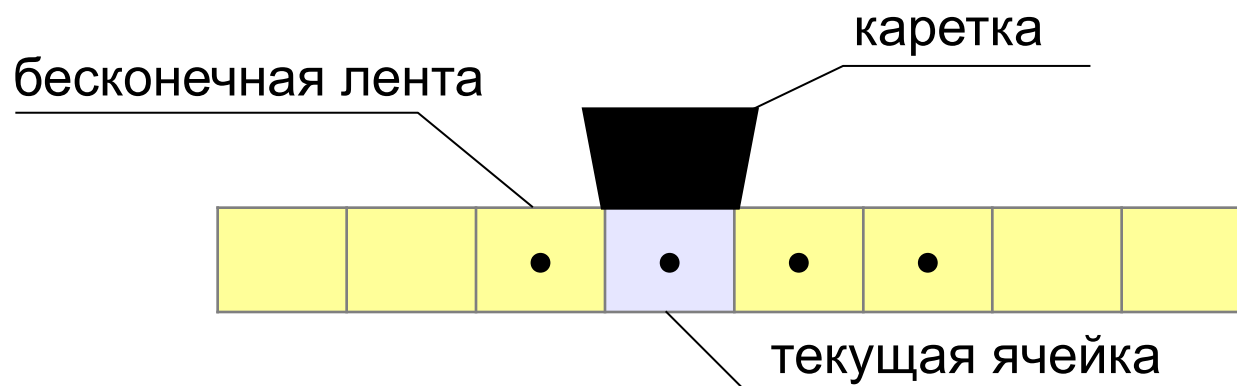
Задача 2. Увеличить на единицу число, записанное в десятичной системе счисления.

Задача 3. Уменьшить на единицу число, записанное в десятичной системе счисления.

Задача 4. Сложить два числа в двоичной системе, разделенные на ленте знаком «+».

Задача 5. Сложить два числа в десятичной системе, разделенные на ленте знаком «+».

Машина Поста



Э. Пост



Алфавит сокращён до двух символов!

- ← переместить каретку на 1 ячейку влево
- переместить каретку на 1 ячейку вправо
- 0 стереть метку в рабочей ячейке (записать 0)
- 1 поставить метку в рабочей ячейке (записать 1)
- ? n_0, n_1 если в рабочей ячейке нет метки, перейти к строке n_0 , иначе перейти к строке n_1
- стоп остановить машину

Программа для машины Поста

1. ←

2. ? 1, 3

3. стоп



Что делает?

Зацикливается?

команда с переходом

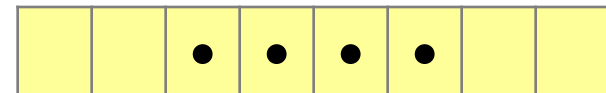
← 3

строки
нумеруются

сдвинуть каретку влево
и перейти на строку 3



Как записывать числа?



4 в унарной системе!



Машины Поста и Тьюринга эквивалентны!

Программы для машины Поста

1	1
2	→
3	→ 1

1	←
2	? 3, 4
3	1 1
4	стоп

1	? 2, 3
2	1 4
3	→ 1
4	стоп

? Что делает программа?

? При каких состояниях ленты?

Программы для машины Поста

Задача 1. Напишите программу для машины Поста, которая увеличивает (уменьшает) число в единичной системе счисления на единицу. Каретка расположена слева от числа.

Задача 2. Напишите программу для машины Поста, которая складывает два числа в единичной системе счисления. Каретка расположена над пробелом, разделяющим эти числа на ленте.

Нормальные алгоритмы Маркова (НАМ)

НАМ – правила обработки символьных строк с помощью подстановок.



А. Марков

а → **н**

ух → **ло**

м → **с**

муха → **мухн** → **млон** → **слон**

→ 0

добавить 0 в начало строки

а → **о**

заменить и стоп



Корова → ?

Карова

Нормальные алгоритмы Маркова (НАМ)

Задача. Удалить из строки, состоящей из букв a и b , первый символ. Например, строка $abba$ должна бы преобразована в bba .

«Очевидное» решение:

$a \rightarrow .$
 $b \rightarrow .$



Что плохо?

$b \rightarrow .$
 $a \rightarrow .$

Правильное решение:

маркер

$*a \rightarrow .$
 $*b \rightarrow .$
 $\rightarrow *$

$abba$

$*abba$

bba

$baab$

$*baab$

aab



НАМ, машины Поста и Тьюринга эквивалентны!

Нормальные алгоритмы Маркова

алфавит: $A = \{0, 1\}$

$0 \rightarrow 00$

$1 \rightarrow 11$

$*0 \rightarrow 0*$

$*1 \rightarrow 1*$

$* \rightarrow =.$

$\rightarrow *$

$*0 \rightarrow 00*$

$*1 \rightarrow 11*$

$* \rightarrow .$

$\rightarrow *$



Что делает НАМ?

Нормальные алгоритмы Маркова

Задача 1. Напишите НАМ, который «сортирует» цифры двоичного числа так, чтобы сначала стояли все нули, а потом – все единицы.

Задача 2. Напишите НАМ, который удаляет последний символ строки, состоящей из цифр 0 и 1. Какую операцию он выполняет, если рассматривать строку как двоичную запись числа?

Задача 3. Напишите НАМ, который умножает двоичное число на 2, добавляя 0 в конец записи числа.

Элементы теории алгоритмов

Сложность вычислений

Что такое сложность вычислений?

Задачи теории алгоритмов:

- существует ли алгоритм решения задачи?
- можно ли им воспользоваться?

Шахматы:

- алгоритм существует (конечное число позиций)
- полный перебор нереален

Требования к алгоритму:

- быстродействие
- минимальный расход
памяти

временная
сложность

пространственная
сложность



Обычно эти требования противоречивы!

Временная сложность

T – количество элементарных операций универсального исполнителя (компьютера)



T зависит от размера входных данных n !

Временная сложность алгоритма – функция $T(n)$.

Задача 1. Вычислить сумму первых трёх элементов массива (при $n \geq 3$).

```
Sum := A[1] + A[2] + A[3]
```

$$T(n) = 3$$

2 сложения
+ запись в
память

Задача 2. Вычислить сумму всех элементов массива.

```
Sum := 0
```

```
нц для i от 1 до n
```

```
    Sum := Sum + A[i]
```

```
кц
```

$$T(n) = 2n + 1$$

n сложений, $n+1$
операций записи

Временная сложность

Задача 3. Отсортировать все элементы массива по возрастанию методом выбора.

```
нц для i от 1 до n-1
  nMin := i;
  нц для j от i+1 до n
    если A[i] < A[nMin] то nMin := j все
  кц
  если nMin <> i то
    c := A[i]; A[i] := A[nMin]; A[nMin] := c
  все
кц
```

Число сравнений: $T_c(n) = (n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$

Число перестановок: $T_p(n) \leq n-1$

зависит от
данных

Сравнение алгоритмов по сложности

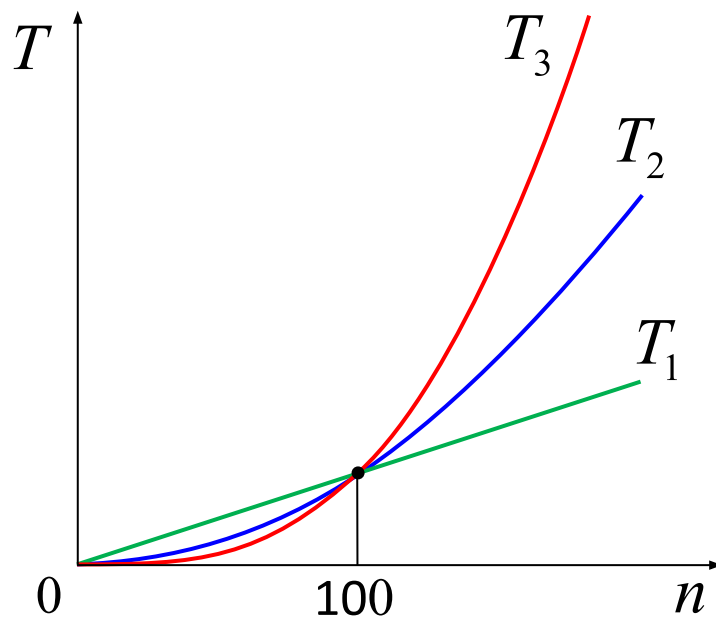
$$T_1(n) = 10000 \cdot n$$

$$T_2(n) = 100 \cdot n^2$$

$$T_3(n) = n^3$$



Какой алгоритм выбрать?



при $n < 100$:

$$T_3(n) < T_2(n) < T_1(n)$$

при $n > 100$:

$$T_3(n) > T_2(n) > T_1(n)$$



Нужно знать размер данных!

Асимптотическая сложность

Асимптотическая сложность – это скорость роста количества операций при больших значениях n .

линейная

постоянная

$$\text{сложность } O(n) \quad \Leftrightarrow \quad T(n) \leq c \cdot n \quad \text{для } n \geq n_0$$

сумма элементов массива:

$$T(n) = 2 \cdot n - 1 \leq 2 \cdot n \quad \text{для } n \geq 1 \quad \Rightarrow$$

квадратичная

$$\text{сложность } O(n^2) \quad \Leftrightarrow \quad T(n) \leq c \cdot n^2 \quad \text{для } n \geq n_0$$

сортировка методом выбора:

$$T_c(n) = \frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2 \quad \text{для } n \geq 0 \quad \Rightarrow \quad O(n^2)$$

Асимптотическая сложность

кубическая

сложность $O(n^3)$ $\Leftrightarrow T(n) \leq c \cdot n^3$ для $n \geq n_0$

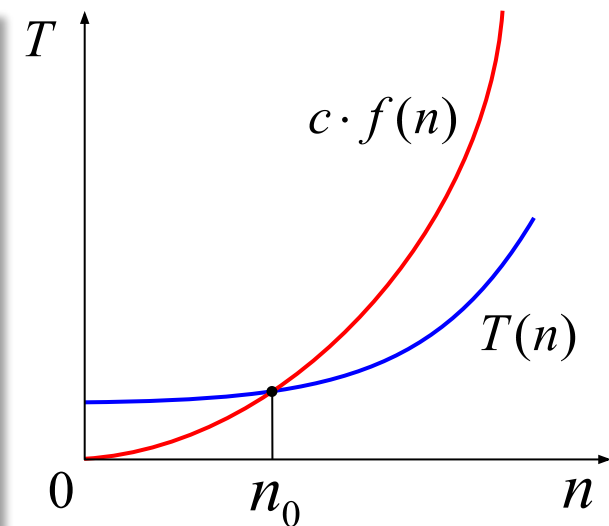
сложность $O(2^n)$

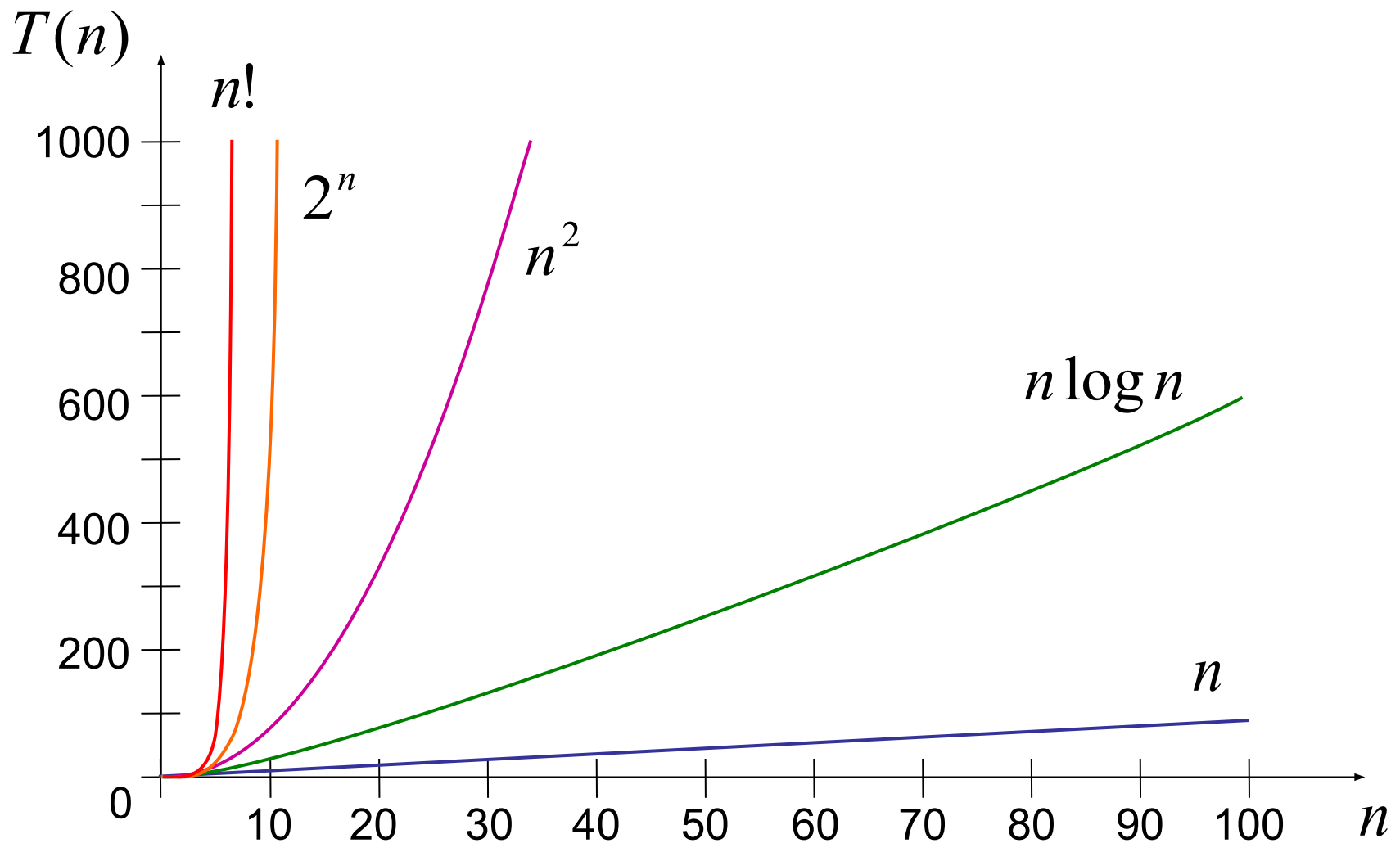
сложность $O(n!)$

задачи оптимизации,
полный перебор вариантов

Алгоритм имеет **асимптотическую сложность** $O(f(n))$, если найдется такая постоянная c , что начиная с некоторого $n = n_0$ выполняется условие

$$T(n) \leq c \cdot f(n)$$





Алгоритмы поиска

Линейный поиск

```
nX := 0
нц для i от 1 до n
    если A[i] = X то
        nX := i
    выход
все
кц
```



Сложность?

сложность $O(n)$

Алгоритмы поиска

Двоичный поиск

```
L := 1; R := n + 1
нц пока L < R - 1
  c := div(L + R, 2)
  если X < A[c] то
    R := c
  иначе
    L := c
все
кц
```

? Какой алгоритм поиска лучше?

$$\log_2 n$$

$$n = 2^m$$

? Сколько шагов?

$$T(n) = m + 1$$

$$T(n) = \log_2 n + 1$$

сложность $O(\log n)$

основание роли
не играет

$$\log_a n = \frac{1}{\log_b a} \cdot \log_b n$$

Алгоритмы сортировки

Метод «пузырька»

```
нц для i от 1 до n-1
  нц для j от n-1 до i шаг -1
    если A[j] > A[j+1] то
      c := A[j]; A[j] := A[j+1]; A[j+1] := c;
    все
  кц
кц
```

сравнений: $T_c(n) = (n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$

присваиваний при перестановках:

$$T_p(n) = 3 \cdot \frac{n(n-1)}{2} = \frac{3}{2}n^2 - \frac{3}{2}n$$

сложность $O(n^2)$

Алгоритмы сортировки

Сортировка подсчётом

```
цел С[1:MAX]
нц для i от 1 до MAX
    С[i] := 0
кц
```

```
нц для i от 1 до n
    С[A[i]] := С[A[i]] + 1
кц
```

```
k := 1
нц для i от 1 до MAX
    нц для j от 1 до С[i]
        A[k] := i
        k := k + 1
    кц
кц
```



Все значения [1, MAX]!

обнулить массив
счётчиков

подсчитать, сколько
каких чисел

заполнить массив
заново

сложность $O(n)$



За счёт чего?

Алгоритмы сортировки



При использовании операций «сравнить» и «переставить» сложность не может быть меньше $O(n \cdot \log n)$!

Сортировка слиянием (*Merge sort*)

$O(n \cdot \log$

Пирамидальная сортировка (*Heap sort*)

$O(n \cdot \log$

Быстрая сортировка (*Quick sort*)

$n)$

в среднем $O(n \cdot \log$

в худшем случае $O(n^2)$

Конец фильма

ПОЛЯКОВ Константин Юрьевич

д.т.н., учитель информатики

ГБОУ СОШ № 163, г. Санкт-Петербург

kpolyakov@mail.ru

ЕРЕМИН Евгений Александрович

к.ф.-м.н., доцент кафедры мультимедийной

дидактики и ИТО ПГГПУ, г. Пермь

eremin@pspu.ac.ru

Источники иллюстраций

1. en.wikipedia.org
2. ru.wikipedia.org
3. иллюстрации художников издательства «Бином»
4. авторские материалы