Тест на простоту

Задача-пример. Дано натуральное число a. Определить, является ли число a простым.

Тест на простоту

Задача-пример. Дано натуральное число a. Определить, является ли число a простым.

Алгоритм. Для каждого натурального числа $2 \leq b \leq \sqrt{a}$ проверить делимость b на a. Если хотя бы для одного b проверка даст положительный результат, то a — составное. В противном случае a — составное.

Тест на простоту

Задача-пример. Дано натуральное число a. Определить, является ли число a простым.

Алгоритм. Для каждого натурального числа $2 \leq b \leq \sqrt{a}$ проверить делимость b на a. Если хотя бы для одного b проверка даст положительный результат, то a — составное. В противном случае a — составное.

$a = 139$ не делится на числа $b = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$, значит 139 —простое.

Тест на простоту

Задача-пример. Дано натуральное число a. Определить, является ли число a простым.

Алгоритм. Для каждого натурального числа $2 \leq b \leq \sqrt{a}$ проверить делимость b на a. Если хотя бы для одного b проверка даст положительный результат, то a — составное. В противном случае a — составное.

$a = 139$ не делится на числа $b = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$, значит 139 —простое.
$a = 143$ не делится на числа $b = 2, 3, 4, 5, 6, 7, 8, 9, 10$, но делится на $b = 11$, значит $143 = 11 \cdot 13$ — составное.

Нахождение наибольшего общего делителя

Задача-пример. Найти наибольший общий делитель двух натуральных чисел a и b.

Нахождение наибольшего общего делителя

Задача-пример. Найти наибольший общий делитель двух натуральных чисел a и b.

Алгоритм Евклида (рекурсивный). Если $a < b$, то $\text{НОД}(a, b) = \text{НОД}(a, c)$, где c — остаток от деления b на a. При этом $\text{НОД}(a, 0) = a$.

Нахождение наибольшего общего делителя

Задача-пример. Найти наибольший общий делитель двух натуральных чисел a и b.

Алгоритм Евклида (рекурсивный). Если $a < b$, то $\text{НОД}(a, b) = \text{НОД}(a, c)$, где c — остаток от деления b на a. При этом $\text{НОД}(a, 0) = a$.

$\text{НОД}(91, 119)$

Нахождение наибольшего общего делителя

Задача-пример. Найти наибольший общий делитель двух натуральных чисел a и b.

Алгоритм Евклида (рекурсивный). Если $a < b$, то $\text{НОД}(a, b) = \text{НОД}(a, c)$, где c — остаток от деления b на a. При этом $\text{НОД}(a, 0) = a$.

$\text{НОД}(91, 119) = \text{НОД}(91, 28)$

Нахождение наибольшего общего делителя

Задача-пример. Найти наибольший общий делитель двух натуральных чисел a и b.

Алгоритм Евклида (рекурсивный). Если $a < b$, то $\text{НОД}(a, b) = \text{НОД}(a, c)$, где c — остаток от деления b на a. При этом $\text{НОД}(a, 0) = a$.

$\text{НОД}(91, 119) = \text{НОД}(91, 28) = \text{НОД}(7, 28)$

Нахождение наибольшего общего делителя

Задача-пример. Найти наибольший общий делитель двух натуральных чисел a и b.

Алгоритм Евклида (рекурсивный). Если $a < b$, то $\text{НОД}(a, b) = \text{НОД}(a, c)$, где c — остаток от деления b на a. При этом $\text{НОД}(a, 0) = a$.

$$\text{НОД}(91, 119) = \text{НОД}(91, 28) = \text{НОД}(7, 28) = \text{НОД}(7, 0) = 7.$$

Алгоритмическая разрешимость

Вопрос. Для каждой ли (по крайней мере арифметической) задачи имеется разрешающий алгоритм?

Алгоритмическая разрешимость

Вопрос. Для каждой ли (по крайней мере арифметической) задачи имеется разрешающий алгоритм?

Проблема. Каждое ли (по крайней мере арифметическое) истинное утверждение можно доказать?

Алгоритмическая разрешимость

Вопрос. Для каждой ли (по крайней мере арифметической) задачи имеется разрешающий алгоритм?

Проблема. Каждое ли (по крайней мере арифметическое) истинное утверждение можно доказать?

Например, чтобы вычислить $f(n)$ можем искать доказательство утверждения вида "$f(n) = m$"для некоторого натурального m.

Давид Гильберт, 1862-1943

Исчисление предикатов (Гильберт)

Аксиомы:

$A \to (B \to A)$; $(A \to B) \to (A \to (B \to C) \to (A \to C))$;
$A \,\&\, B \to A$; $A \,\&\, B \to B$; $A \to (B \to A \,\&\, B)$; $A \to A \vee B$;
$B \to A \vee B$; $(A \to C) \to ((B \to C) \to (A \vee B \to C))$;
$(A \to B) \to ((A \to \neg B) \to \neg A)$; $\neg\neg A \to A$; $\forall x A(x) \to A(t)$;
$A(t) \to \exists x A(x)$.

Исчисление предикатов (Гильберт)

Аксиомы:

$A \to (B \to A)$; $(A \to B) \to (A \to (B \to C) \to (A \to C))$;
$A \,\&\, B \to A$; $A \,\&\, B \to B$; $A \to (B \to A \,\&\, B)$; $A \to A \vee B$;
$B \to A \vee B$; $(A \to C) \to ((B \to C) \to (A \vee B \to C))$;
$(A \to B) \to ((A \to \neg B) \to \neg A)$; $\neg\neg A \to A$; $\forall x A(x) \to A(t)$;
$A(t) \to \exists x A(x)$.

Правила вывода:

$\frac{A, A \to B}{B}$; $\frac{B \to A(x)}{B \to \forall x A(x)}$; $\frac{A(x) \to B}{\exists x A(x) \to B}$;

Исчисление предикатов (Гильберт)

Аксиомы:

$A \to (B \to A)$; $(A \to B) \to (A \to (B \to C) \to (A \to C))$;
$A \,\&\, B \to A$; $A \,\&\, B \to B$; $A \to (B \to A \,\&\, B)$; $A \to A \vee B$;
$B \to A \vee B$; $(A \to C) \to ((B \to C) \to (A \vee B \to C))$;
$(A \to B) \to ((A \to \neg B) \to \neg A)$; $\neg\neg A \to A$; $\forall x A(x) \to A(t)$;
$A(t) \to \exists x A(x)$.

Исчисление предикатов (Гильберт)

Аксиомы:

$A \to (B \to A)$; $(A \to B) \to (A \to (B \to C) \to (A \to C))$;
$A \,\&\, B \to A$; $A \,\&\, B \to B$; $A \to (B \to A \,\&\, B)$; $A \to A \vee B$;
$B \to A \vee B$; $(A \to C) \to ((B \to C) \to (A \vee B \to C))$;
$(A \to B) \to ((A \to \neg B) \to \neg A)$; $\neg\neg A \to A$; $\forall x A(x) \to A(t)$;
$A(t) \to \exists x A(x)$.

Правила вывода:

$\frac{A, A \to B}{B}$; $\frac{B \to A(x)}{B \to \forall x A(x)}$; $\frac{A(x) \to B}{\exists x A(x) \to B}$;

Джузеппе Пеано, 1858 — 1932

Аксиоматика арифметики (Пеано)

1. $s(x) \neq 0$;
2. $s(x) = s(y) \to x = y$;
3. $x + 0 = x$;
4. $x + s(y) = s(x + y)$;
5. $x \cdot 0 = 0$;
6. $x \cdot s(y) = x \cdot y + x$;
7. $A(0)\ \&\ \forall x(A(x) \to A(s(x)) \to \forall x A(x)$.

Курт Гёдель, 1906 — 1978

Полнота формальной математики (Гёдель)

1. Теорема о существовании модели. Если утверждение не доказуемо, то его отрицание реализуемо.

Полнота формальной математики (Гёдель)

1. Теорема о существовании модели. Если утверждение не доказуемо, то его отрицание реализуемо. (Как, например, отрицание пятого постулата Евклида реализуемо в геометрии Лобачевского).

Полнота формальной математики (Гёдель)

1. Теорема о существовании модели. Если утверждение не доказуемо, то его отрицание реализуемо. (Как, например, отрицание пятого постулата Евклида реализуемо в геометрии Лобачевского).
2. Полнота исчисления предикатов. Всякое всюду истинное утверждение доказуемо в исчислении предикатов.

Неполнота формальной арифметики (Гёдель)

Теорема о Гёделя о неполноте. Существуют арифметические утверждения, которые нельзя доказать или опровергнуть.

Неполнота формальной арифметики (Гёдель)

Теорема о Гёделя о неполноте. Существуют арифметические утверждения, которые нельзя доказать или опровергнуть. Таким образом, в арифметике имеются истинные, но недоказуемые утверждения.

Неформальное объяснение неполноты

Парадокс лжеца. Данное утверждение — ложно.

Неформальное объяснение неполноты

Парадокс лжеца. Данное утверждение — ложно.

Рассмотрим похожее утверждение.
Данное утверждение — не доказуемо.

Неформальное объяснение неполноты

Парадокс лжеца. Данное утверждение — ложно.

Рассмотрим похожее утверждение.
Данное утверждение — не доказуемо.

Если последнее утверждение ложно, то оно доказуемо и, значит, истинно, что невозможно.

Неформальное объяснение неполноты

Парадокс лжеца. Данное утверждение — ложно.

Рассмотрим похожее утверждение.
Данное утверждение — не доказуемо.

Если последнее утверждение ложно, то оно доказуемо и, значит, истинно, что невозможно. Поэтому оно истинно и, следовательно, не доказуемо.

Формализация понятия алгоритма

- Общерекурсивные функции (Гёдель);
- Машины Тьюринга;
- λ-исчисление (Черч);
- Машины Поста;
- Нормальные алгорифмы Маркова;

Формализация понятия алгоритма

- Общерекурсивные функции (Гёдель);
- Машины Тьюринга;
- λ-исчисление (Черч);
- Машины Поста;
- Нормальные алгорифмы Маркова;

  ...
- Языки программирования (Паскаль, Си, Java, ...).

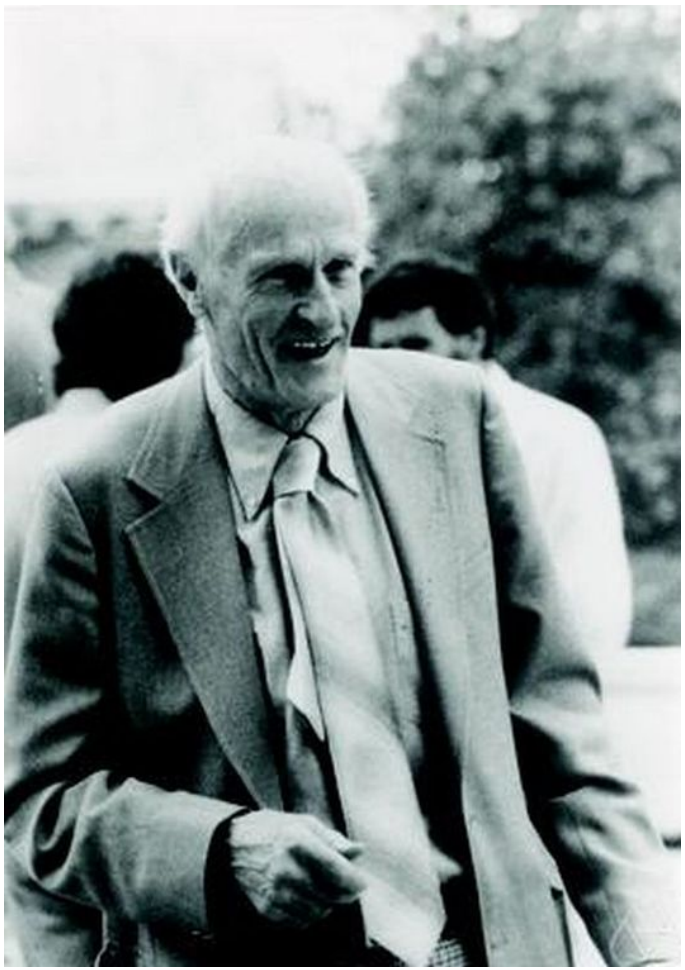
Формализация понятия алгоритма

- Общерекурсивные функции (Гёдель);
- Машины Тьюринга;
- λ-исчисление (Черч);
- Машины Поста;
- Нормальные алгорифмы Маркова;

  ...
- Языки программирования (Паскаль, Си, Java, ...).

Лемма о β-функции Гёделя. Любой алгоритм и результат его работы может быть выражен в виде арифметического утверждения.

Стивен Соул Клини, 1909 — 1994



Теорема о рекурсии (Клини)

Куайны (quine) — программы, печатающие свой текст

Теорема о рекурсии (Клини)

Куайны (quine) — программы, печатающие свой текст

Для текста программы P с параметром X обозначим через $f(P)$ программу P, в которой вместо X подставлено значение P.

Теорема о рекурсии (Клини)

Куайны (quine) — программы, печатающие свой текст

Для текста программы P с параметром X обозначим через $f(P)$ программу P, в которой вместо X подставлено значение P.

Представим алгоритм, который текст X преобразует в текст программы, печатающий текст $f(X)$.

Теорема о рекурсии (Клини)

Куайны (quine) — программы, печатающие свой текст

Для текста программы P с параметром X обозначим через $f(P)$ программу P, в которой вместо X подставлено значение P.

Представим алгоритм, который текст X преобразует в текст программы, печатающий текст $f(X)$.

Пусть P_0 — текст программы этого алгоритма.

Теорема о рекурсии (Клини)

Куайны (quine) — программы, печатающие свой текст

Для текста программы P с параметром X обозначим через $f(P)$ программу P, в которой вместо X подставлено значение P.

Представим алгоритм, который текст X преобразует в текст программы, печатающий текст $f(X)$.

Пусть P_0 — текст программы этого алгоритма. Тогда программа $f(P_0)$ печатает свой текст.

Пример куайна

Пример на Паскале
(https://www.nyx.net/ gthompso/quine.htm)

```
program s;const p='program s;const
p=';a='a';aa='';';aaa='a='';aaaa='';aaaaa='begin
write(p,aaaa,p,aa, aaa,a,aa,a,aaa,aaaa, aa, aa, a, a,aaa, aaa,
aaaa,aa ,a,a,a, aaa, aaaa,aaaa,aa,a,a,a,a,aaa,aaaaa,aa,aaaaa)
end.';begin write(p, aaaa, p ,aa, aaa, a, aa, a, aaa, aaaa, aa, aa,
a, a, aaa, aaa, aaaa, aa, a, a, a, aaa, aaaa, aaaa, aa, a, a, a,
a,aaa,aaaaa,aa,aaaaa) end.
```

Значение теоремы о рекурсии

С практической точки зрения существование куайнов дает программисту возможность использовать в вычисляющем алгоритме текст этого алгоритма.

Значение теоремы о рекурсии

С практической точки зрения существование куайнов дает программисту возможность использовать в вычисляющем алгоритме текст этого алгоритма.

А с математической точки зрения — возможность использовать при построении арифметического утверждения его текст (или соответствующий ему код).

Неполнота формальной арифметики (Гедель)

Теорема о Геделя о неполноте. Для всякого алгоритмически заданного расширения аксиоматики Пеано существуют арифметические утверждения, которые нельзя доказать или опровергнуть.

Неполнота формальной арифметики (Гедель)

Теорема о Геделя о неполноте. Для всякого алгоритмически заданного расширения аксиоматики Пеано существуют арифметические утверждения, которые нельзя доказать или опровергнуть.

Следствие (Теорема Черча). Не существует алгоритма, определяющего является ли данное арифметическое утверждение доказуемым или не доказуемым.

Алонзо Чёрч, 1903 — 1995

Алан Тьюринг, 1912 — 1954

Разрешимость и полу-разрешимость

Другими словами, проблема доказуемости утверждения неразрешима

Разрешимость и полу-разрешимость

Другими словами, проблема доказуемости утверждения неразрешима, но является полу-разрешимой (перечислимой) в том смысле, что если утверждение доказуемо, то существует сертификат (доказательство), проверка которого может быть осуществлена алгоритмом.

Разрешимость и полу-разрешимость

Другими словами, проблема доказуемости утверждения неразрешима, но является полу-разрешимой (перечислимой) в том смысле, что если утверждение доказуемо, то существует сертификат (доказательство), проверка которого может быть осуществлена алгоритмом.
Другие подобные проблемы:

- проблема остановки (заканчивает ли программа работу за конечное число шагов);
- проблема равенства слов (Марков);
- проблема существования решений арифметических уравнений в целых числах (Матиясевич).

Реализуемая разрешимость

Вопрос. Пусть некоторая математическая проблема алгоритмически разрешима. Возможно ли реализовать данный алгоритм на практике?

Реализуемая разрешимость

Вопрос. Пусть некоторая математическая проблема алгоритмически разрешима. Возможно ли реализовать данный алгоритм на практике?

Одной из основных ограничений на реализуемые алгоритмы является алгоритмы, работающий не более чем $p(n)$ шагов, где n — длина входных данных алгоритма, а p — многочлен. Класс проблем, разрешимых такими алгоритмами, образует класс сложности **P**.

Сложность алгоритма Евиклида

Теорема Ламё (1844). Количество делений, которое нужно выполнить для нахождения наибольшего общего делителя двух целых чисел, не превосходит количества цифр в меньшем числе, умноженного на пять.

Сложность алгоритма Евиклида

Теорема Ламё (1844). Количество делений, которое нужно выполнить для нахождения наибольшего общего делителя двух целых чисел, не превосходит количества цифр в меньшем числе, умноженного на пять.

Таким образом, проблема нахождения наибольшего общего делителя принадлежит **P**.

Сложность проверки на простоту

Теорема (Агравал, Каял, Саксен, 2002). Существует алгоритм (тест) проверки простоты числа a, время работы которого ограничено многочленом 11-й степени от количества цифр в a.

Сложность проверки на простоту

Теорема (Агравал, Каял, Саксен, 2002). Существует алгоритм (тест) проверки простоты числа a, время работы которого ограничено многочленом 11-й степени от количества цифр в a.

Таким образом, проблема определения простоты числа также принадлежит **P**.

Класс NP

Разрешимая проблема принадлежит классу **NP**, если она полу-разрешима алгоритмом, у которого время проверки сертификата и и его длина ограничены многочленом.

Класс NP

Разрешимая проблема принадлежит классу **NP**, если она полу-разрешима алгоритмом, у которого время проверки сертификата и и его длина ограничены многочленом.

В случае "наивного теста"на не-простоту числа a сертификатом будет любой нетривальный делитель числа a.

Класс NP

Другие примеры **NP**-задач:

1. Проблема нулевой суммы подмножества. По данному множеству целых чисел проверить существование подмножества, сумма элемента которого равна нулю.

Класс NP

Другие примеры **NP**-задач:

1. Проблема нулевой суммы подмножества. По данному множеству целых чисел проверить существование подмножества, сумма элемента которого равна нулю.
2. По данной булевой формуле (логическая формула, не содержащая $\forall$ и $\exists$) проверить выполнимость этой формулы.

Класс NP

Другие примеры **NP**-задач:

1. Проблема нулевой суммы подмножества. По данному множеству целых чисел проверить существование подмножества, сумма элемента которого равна нулю.
2. По данной булевой формуле (логическая формула, не содержащая $\forall$ и $\exists$) проверить выполнимость этой формулы.
3. Проблема раскраски графа.

Класс NP

Другие примеры **NP**-задач:

1. Проблема нулевой суммы подмножества. По данному множеству целых чисел проверить существование подмножества, сумма элемента которого равна нулю.
2. По данной булевой формуле (логическая формула, не содержащая $\forall$ и $\exists$) проверить выполнимость этой формулы.
3. Проблема раскраски графа.
4. Проблема обхода всех вершин связного графа без повторений.

   ...

P=NP?

Открытый (и очень важный) вопрос. (Кук, 1971) Совпадают ли классы **NP** и **P**?