Дискретная математика. Модуль 3. Лекция 5

Лекторий ПМИ ФКН 2015-2016
Гринберг Вадим
Жижин Пётр
Пузырев Дмитрий

8 февраля 2016

Введение в теорию алгоритмов. Задача вычисления функций. Разрешимость диофантова уравнения.

В начале отметим, что в последущих задачах мы будем заниматься *возможностью* написания алгоритма, решающего данную задачу, но не его *вычислительной сложностью*.

Перед тем, как дать, собственно, определение *алгоритма*, приведем ряд показательных примеров, попутно вводя базовые свойства, которые помогут в написании определения в дальнейшем.

Для начала рассмотрим наиболее распространённую задачу вычисления функций. Имеется некоторая функция $f : x \mapsto f(x)$, которая переводит элементы множества входов в множество результатов с помощью некоторого алгоритма.

Мы предполагаем, что алгоритм действует на некотором *конструктивном множестве*. Чаще всего мы будем говорить о мн-ве $\mathbb{N}$.

Определение: Функция называется вычислимой, если для неё существует некоторый алгоритм вычисления.

Определение: Конструктивное множество — множество, для которого есть вычислимая биекция с множеством натуральных чисел.

Классическим примером задач на вычисление функций является задача на разрешимость диофантова уравнения. Ранее мы уже обсуждали решение *линейных диофантовых уравнений*, являющееся, вообще говоря, частным случаем. Однако, начиная с третьей степени, не представляется возможным привести формулу для решения. Остаётся только алгоритмическое построение.

Итак, существует многочлен $f(x_1 \dots x_n) = 0$ от произвольного количества переменных. Требуется привести такой алгоритм, у которого:

На вход подаются: последовательности натуральных чисел $\mathbb{N}^*$.

На выходе: {0,1}. 1 — если решение диофантова уравнения существует, 0 — если решений (в целых числах) нет. В качестве решения может быть предложен алгоритм на странице 2 (алгоритм 1).

В итоге получили:

$$A(f) = \begin{cases} 1, f(x_1 \dots x_n) \text{ разрешимо} \\ \text{иначе не определена} \end{cases}$$

Однако можно заметить, что получена не та функция, которая нужна. Почему так? Всё очень просто, она не возвращает ноль. Например, для уравнения $x^2+1=0$ она просто зациклится и ничего никогда не вернёт.

Здесь введём первое свойство алгоритмов.

Первое свойство алгоритмов. *Композиция вычислимых функций вычислима.*

Algorithm 1 Алгоритм проверки разрешимости диофантового уравнения

function SolveEquation
  for $i := 0 \ldots \infty$ **do**
    $x \leftarrow \pi(t)$
    if $f(x) = 0$ **then**
      return 1
    end if
  end for
end function

Доказательство. Пусть существуют вычислимые функции f, переводящая множество входов X в множество выходов Y и g, переводящая Y в Z.

$$\begin{cases} f : X \to Y \\ g : Y \to Z \end{cases}$$

Построить $g \circ f : X \to Z$ достаточно просто. На первом шаге нужно применить функцию f. Далее берём множество выходов f и подаём на вход в g. На выходе получим некоторое множество выходов Z. Вычислимая композиция построена.

Algorithm 2 Алгоритм получения композиции функций

function Composition(x)
  $t \leftarrow f(x)$
  return $g(t)$
end function

Q.E.D.

Итак, есть вычислимая биекция $\pi : \mathbb{N}^* \to \mathbb{N}$.

Построим композицию $\mathbb{N}^* \to \mathbb{N} \to \{0, 1\}$.

$$\begin{cases} \pi^{-1} : B \to A \\ f : A \to A \\ \pi : A \to B \end{cases}$$

Построить это можно применением композиции $\pi \circ f \circ \pi^{-1}$. Получим алгоритм из B в B. Поэтому нам, по сути, **без разницы какие множества на входе и выходе** так как можно получить легко из одного другое.

Заметим, что если некоторая биекция π вычислима, то обратная ей π^{-1} также будет вычислима. Способ вычисления схож с алгоритмом для диофантовых уравнений.

Рассматриваем все возможные значения входов и вычисляем $\pi^{-1}(x)$.

Algorithm 3 Алгоритм построения обратной функции для биекции

function RevBiection(x)
  for $n := 0 \ldots \infty$ **do**
    if $\pi(n) = x$ **then**
      return n
    end if
  end for
end function

Стоит отметить, что данный алгоритм всегда завершается так как (по определению) биекция сюрьективна и найдется номер n для которого $\pi(n) = x$.

Задачи разрешения. Разрешимые множества

Вспомним что такое характеристическая функция множества S:

$$\chi_S(x) = \begin{cases} 1,\ x \in S \\ 0,\ x \notin S \end{cases}$$

Определение: Множество называется разрешимым, если его характеристическая функция вычислима.

Понятно, что в таком виде очень удобно исследовать вопросы о свойствах некоторых объектов (разрешимости диофантовых уравнений, связность графа и так далее). Поэтому разрешимые множества очень важный объект.

Утверждение: Конечное множество $S \subset \mathbb{N}$ всегда разрешимо.

Доказательство. Следующий алгоритм вычисляет характеристическую функцию.

Algorithm 4 Алгоритм разрешения конечного множества

function SolveSet(s, x)
    if $x = s_1$ **then**
        return 1
    end if
    $\ldots$
    if $x = s_n$ **then**
        return 1
    end if
    return 0
end function

Стоит отметить, что функция останавливает свою работу после возвращения значения (как в языке C, например, а не как в Pascal).

По сути мы просто по очереди сравниваем элементы множества на равенство входному. Их у нас конечное число, а значит и алгоритм всегда работает конечное время. Поэтому функция вычислима. **Q.E.D.**

Утверждение: Если множества A, B разрешимы, то и множества $A \cap B$, $A \cup B$, $A \setminus B$, $\mathbb{N} \setminus A$ тоже разрешимы.

Доказательство. Алгоритмы для определения характеристических функций очень похожи, поэтому будет приведена сначала общая часть всех алгоритмов. На месте точек

Algorithm 5 Алгоритм разрешения операций над множествами

function SolveSet(A, B, x)
    $a \leftarrow \chi_A(x)$
    $b \leftarrow \chi_B(x)$
    return $\ldots$
end function

должны стоять:

1. $a \wedge b$ для $A \cap B$
2. $a \vee b$ для $A \cup B$
3. $a \wedge \neg b$ для $A \setminus B$
4. $\neg a$ для $\mathbb{N} \setminus A$

Фактически, мы пользуемся свойствами характеристических функций. **Q.E.D.**

Теорема. *Существует неразрешимое множество.*

Доказательство. Алгоритмов всего счётное количество, а множество подмножеств натуральных чисел континуально, значит такое множество действительно существует. **Q.E.D.**

Алгоритмы перечисления. Перечислимые множества. Теорема Поста

Определение: Алгоритм перечисления – такой алгоритм, у которого нет входа, он работает и может выводить некоторые числа, причём все напечатанные числа составляют счетное множество.

Определение 1: Множество S называется перечислимым, если есть алгоритм перечисления всех его элементов.

Теорема. *Существует неперечислимое множество.*

Доказательство. Алгоритмов перечисления – счётное множество, а бесконечных подмножеств множества натуральных чисел несчётное количество (это разность несчётного и счётного множества).

Q.E.D.

Определение 2: Множество S – перечислимо, если существует такая вычислимая функция:

$$f : \mathbb{N} \to \mathbb{N} \begin{cases} f(\mathbb{N}) = S \\ \text{Область определения } f \text{ равна либо } \mathbb{N}, \text{либо } [n]. \end{cases}$$

Теорема. *Определения* 1 *и* 2 *эквивалентны.*

Доказательство.

$\Longrightarrow$ Пусть A – алгоритм перечисления множества S. Тогда возьмём следующий алгоритм B: принимает на вход число n, запускает алгоритм A и считает, сколько чисел напечатано. Как только вывели $n + 1$ слово – алгоритм печатает результат.

Покажем, что соблюдаются свойства вычислимой функции:

1. $B(n) = S$, так как $\forall\ n\ \exists\ B(n) \Rightarrow \begin{cases} \forall\ x \in S\ \exists\ B(x) \\ \forall\ x \notin S \text{ никогда не выведет } B(x) \end{cases}$

2. Пусть S – бесконечное множество, тогда функция, задаваемая B, определена везде, значит $dom\ B = \mathbb{N}$. Если B работает на n числах, то алгоритм переберёт их и остановится.

$\Longleftarrow$ Возьмём следующий алгоритм перечисления B для множества S:

Algorithm 6 Алгоритм перечисления разрешимого множества

```
function PrintSet(S)
    for i := 0 ... ∞ do
        if f(i) = 1 then
            print i
        end if
    end for
end function
```

Если функция определена для некоторых n чисел, то ровно их он и напечатает. Если $dom\ f = \mathbb{N}$, то алгоритм никогда не остановится, то есть напечатает всю область определения f. Значит, существует алгоритм, перечисляющий S.

Q.E.D.

Теорема Поста. *Если множества A и $\overline{A}$ перечислимы, то множество A разрешимо.*

Доказательство. Алгоритм разрешения множества A устроен так. Он исполняет модифицированные алгоритмы перечисления множеств A и $\overline{A}$ параллельно: один шаг работы алгоритма перечисления множества A, затем один шаг работы алгоритма перечисления $\overline{A}$ и так далее.

Вместо того, чтобы печатать очередной элемент, модифицированный алгоритм перечисления запоминает его в списке элементов множества. (В любой момент исполнения алгоритма такой список конечен.)

Когда один из списков увеличивается, добавленный элемент сравнивается со входом x. Если обнаружено вхождение x в список элементов множества A, то алгоритм разрешения останавливается и выдаёт результат 1. Если обнаружено вхождение x в список элементов множества $\overline{A}$, то алгоритм разрешения останавливается и выдаёт результат 0. В остальных случаях продолжается работа алгоритмов перечисления.

Докажем корректность алгоритма. Пусть $x \in A$. Тогда x заведомо не входит в список элементов $\overline{A}$ и результат 0 невозможен. С другой стороны, рано или поздно x появится в списке элементов A, поэтому алгоритм выдаст результат 1.

Аналогично рассуждаем в случае $x \notin A$.

Q.E.D.