Интуитивное и математическое понятие алгоритма. Алгоритмы как частично-рекурсивные функции

1. Интуитивное понятие алгоритма

Алгоритмами являются способы решения, описанные с помощью предписаний по обработке, которые удовлетворяют определенным требованиям.

Определение 1. Алгоритм есть способ с точным (т.е. выраженным в точно определенном языке) конечным описанием применения практически выполнимых элементарных шагов переработки информации.

Это интуитивное понятие алгоритма.

Алгоритмы типичным образом решают не только частные задачи, но и классы задач. Подлежащие решению частные задачи, выделяемые по мере необходимости из рассматриваемого класса, определяются с помощью параметров. Параметры играют роль исходных данных для алгоритма. Алгоритмы, как правило, по этим исходным данным доставляют результаты, которые в случае задач информационной обработки могут быть информацией (вернее, представлением информации) или последовательностью указаний (управляющих сигналов), по которым осуществляются определенные преобразования.

Независимо от формы описания для алгоритмов важно различать следующие аспекты:

- постановку задачи, которая должна быть решена с помощью алгоритма;
- специфичный способ, каким решается задача, при этом для алгоритма различают элементарные шаги обработки, которые имеются в распоряжении, и описание выбора отдельных подлежащих выполнению шагов.

2. Свойства алгоритма

Для алгоритмов справедливы некоторые общие закономерности, называемые свойствами.

Массовость. Алгоритм разрабатывается для целого класса задач.

Дискретность. Алгоритм представляется в виде конечной последовательности шагов.

Конечность. Алгоритм должен закончить работу за конечное число шагов.

Определенность. Каждый шаг должен быть точно и недвусмысленно определен. Перед началом каждого шага должны быть определены все необходимые ему данные.

Эффективность. Если задачу можно решить несколькими способами следует выбрать тот, который потребует меньше ресурсов.

Пример (алгоритм Евклида нахождения наибольшего общего делителя). На входе алгоритма два натуральных числа.

1. Если числа равны, выдать любое из них в качестве ответа и закончить алгоритм.
2. Заменить большее число на разность большего и меньшего.
3. Перейти к шагу 1.

Формальное понятие алгоритмов тесно связано с понятиями рекурсивных функций, машин Тьюринга, нормальных алгоритмов Маркова или систем текстовых замен (СТЗ).

Определение 2. Алгоритм называется *терминистическим*, если он завершается за конечное число шагов, *детерминированным*, если нет свободы в выборе очередного шага алгоритма.

3. Алгоритм как рекурсивная функция

Существование или не существование алгоритма может быть установлено, если найти такой математический объект, который будет существовать в точности тогда, когда и алгоритм. Таким математическим объектом может быть рекурсивная функция (РФ). Функция определяется однозначно, если известен закон, по которому каждому набору $x_1, \ldots, x_n$ ставится в соответствие одно значение функции y. Закон может быть произвольным, в том числе это может быть алгоритм вычисления значения функции. Тогда функцию называют вычислимой, а алгоритм, по которому вычисляется функция, называется алгоритмом, сопутствующим вычислимой функции. Рекурсивные функции являются частным классом вычислимых функций.

РФ строится здесь на множестве целых неотрицательных чисел следующим образом: задаются 3 базовые РФ, для которых сопутствующие алгоритмы одношаговые. Затем используются 3 приема, называемые операторами подстановки, рекурсии и минимизации, с помощью которых на основе базовых функций строятся более сложные РФ. По существу приведенные операторы — алгоритмы, комбинируя которые получают более сложные алгоритмы.

Перечислим простейшие базовые функции.

1. Функция произвольного количества аргументов тождественно равная нулю.

   Знак функции φ_n, где n — количество аргументов.

   Сопутствующий алгоритм: если знаком функции является φ_n, то значение функции равно нулю.

   Например: $\varphi_1(3) = 0$, $\varphi_3(4, 56, 78) = 0$.
2. Тождественная функция.

   Знак функции $\psi_{n,i}$, $0 < i \leq n$, где n — количество аргументов, i — номер аргумента.

   Сопутствующий алгоритм: если знак функции $\psi_{n,i}$, то значением функции является значение i-го аргумента, считая слева направо.

   Например: $\psi_{3,2}(3, 22, 54) = 22$.
3. Функция получения последователя. Функция одного независимого аргумента.

   Знак функции λ.

   Сопутствующий алгоритм: если знак функции λ, то значение функции — число, следующее за значением аргумента.

   Например: $\lambda(5) = 6$ или 5'=6.

Перечислим три приема построения сложных РФ.

1. Оператор подстановки $F(f_1, \ldots, f_n)$. Сопутствующий алгоритм: вычисляются значения функций $f_1, \ldots, f_n$ и используются как аргументы при вычислении F.

Например $\tau(y) = \lambda(\lambda(y)) = \lambda(y') = y''$.

2. Оператор рекурсии R. $f ::= R[f_1, f_2, x(y)]$.

Здесь f — функция n аргументов, f_1 — $(n-1)$ аргумента, f_2 — $(n+1)$ аргумента, причем $n-1$ аргументов функций совпадают, а 2 следующих аргумента называются дополнительными. Один из них, x — называется главным дополнительным аргументом (ГДА). Он войдет в определяющую функцию. Другой, y — вспомогательный дополнительный аргумент (ВДА), использующийся при построении новой функции.

Сопутствующий алгоритм: оператор рекурсии строит новую функцию по двум условиям:

$f(0) = f_1$

$f(i') = f_2(i, f(i))$.

Значением искомой функции при нулевом значении ГДА считать значение функции f_1, а значением новой функции для каждого последующего значения ГДА считать значение функции f_2 для предыдущего значения ГДА и для значения ВДА, совпадающего со значением искомой функции на предыдущем шаге.

Например:

$PR(x) ::= R[\varphi_0, \psi_{2,1}(x, y), x(y)]$;

$PR(0) = \varphi_0 = 0$;

$PR(1) = \psi_{2,1}(0, PR(0)) = 0;$
$PR(2) = \psi_{2,1}(1, PR(1)) = 1;$
$\dots$
$PR(x) = x - 1.$

3. Оператор минимизации или построение по первому нулю.

$f ::= \mu[f_1, (x)]$ или $f(x_1, \dots, x_n) = \mu[f_1(x_1, \dots, x_n, y), (y)]$ строит новую функцию f с помощью функции f_1, имеющей $n + 1$ аргументов, и дополнительного исчезающего аргумента. Сопутствующий алгоритм: придавать последнему аргументу значения, начиная с нуля, вычисляя при этом значение функции f_1. Как только значение функции f_1 станет равным нулю, значение дополнительного аргумента принимаем за значение искомой функции для тех главных аргументов, для которых вычислялось значение функции f_1.

Например:
$r(x) ::= \mu[\psi_{2,1}(x, y), (y)];$
$r(0) = 0;$
$r(i) = \psi_{2,1}(i, y)$ не определено для $i <> 0$.

Все базовые функции и функции, построенные без оператора минимизации, определены для всех значений дополнительных аргументов. Функции, построенные с помощью оператора минимизации, могут быть определены не для всех значений исходных данных. Большинство известных математических функций — рекурсивные.

Например:

Функция $y = x + 1$ совпадает с базовой.

Построим функцию $w = x + y$.

Получим функцию $f^*(x, y, z)$ подстановкой $(z + 1)$ вместо z в функцию $\psi_{3,3}(x, y, z)$, т.е. $f^*(x, y, z) = \psi_{3,3}(x, y, \lambda(z)) = z + 1$.

Затем воспользуемся следующим:
$S(x, y) ::= R[\psi_{1,1}(x), f^*(x, y, z), y(z)];$
$S(x, 0) = \psi_{1,1}(x) = x;$
$S(x, 1) = f^*(x, 0, S(x, 0)) = S(x, 0) + 1 = x + 1;$
$\dots$
$S(x, y) = x + y.$

Класс вычислимых функций исчерпывается классом РФ. Каков бы ни был алгоритм, перерабатывающий последовательность целых неотрицательных чисел в целые неотрицательные числа, найдется сопутствующий некоторой РФ алгоритм, эквивалентный данному и наоборот. Если нельзя построить РФ, то нельзя разработать алгоритм решения задачи.