Лекция 2. Основные типы алгоритмических моделей.

п.1. Что в интуитивном понятии алгоритма нуждается в уточнении

Попытки сформулировать точное понятие "любой алгоритм" привели в 30-х гг. 20-го века к возникновению теории алгоритмов. Она оказалась тесно связанной с математической логикой, а вместе они образовали новую науку - **метаматематику**, которая изучает основные средства математики (методы доказательств, способы построения аксиоматических теорий, свойства математических процедур и т.п.) математическими методами. Когда же началось бурное развитие вычислительной техники и наук, связанных с ее использованием, то выяснилось, что в основе этих наук должна лежать теория алгоритмов, поскольку ЭВМ может реализовать только такие процедуры, которые представлены в виде алгоритмов.

Рассмотрим неформально, что именно в интуитивном понятии алгоритма нуждается в уточнении.

1. Каждый алгоритм имеет дело с данными - входными, промежуточными, выходными. Для того чтобы уточнить понятие данных, фиксируется конечный алфавит исходных символов (цифры, буквы и т.п.) и указываются правила построения алгоритмических объектов. Типичным используемым средством является индуктивное построение. Например, определение идентификатора в языке программирования может выглядеть следующим образом: идентификатор - это либо буква, либо идентификатор, к которому приписана справа либо буква, либо цифра. Слова конечной длины в конечных алфавитах - наиболее обычный тип алгоритмических данных, а число символов в слове - естественная мера объема данных. Другой случай алгоритмических объектов - формулы. Примером могут служить формулы алгебры предикатов и алгебры высказываний. В этом случае не каждое слово в алфавите будет формулой.

2. Алгоритм для размещения данных требует памяти. Память обычно считается однородной и дискретной, т.е. она состоит из одинаковых ячеек, причем каждая ячейка может содержать один символ данных, что позволяет согласовать единицы измерения объема данных и памяти.

3. Алгоритм состоит из отдельных элементарных шагов, причем множество различных шагов, из которых составлен алгоритм, конечно. Типичный пример множества элементарных шагов система команд ЭВМ.

4. Последовательность шагов алгоритма *детерминирована*, т.е. после каждого шага указывается, какой шаг следует выполнять дальше, либо указывается, когда следует работу алгоритма считать законченной.

5. Алгоритм должен обладать *результативностью*, т.е. останавливаться после конечного числа шагов (зависящего от исходных данных) с выдачей результата. Данное свойство иногда называют сходимостью алгоритма.

6. Алгоритм предполагает наличие механизма реализации, который по описанию алгоритма порождает процесс вычисления на основе исходных данных. Предполагается, что описание алгоритма и механизм его реализации конечны.

Можно заметить аналогию с вычислительными машинами. Требование 1 соответствует цифровой природе ЭВМ, требование 2 -памяти ЭВМ, требование 3 - программе машины, требование 4 - ее логической природе, требования 5,6 - вычислительному устройству и его возможностям.

Имеются также некоторые черты неформального понятия алгоритма, Относительно которых не достигнуто окончательного соглашения. Эти черты сформулируем в виде вопросов и ответов.

7. Следует ли фиксировать конечную границу для размера входных данных?

8. Следует ли фиксировать конечную границу для числа элементарных шагов?

9. Следует ли фиксировать конечную границу для размера памяти?

10. Следует ли ограничить число шагов вычисления?

На все эти вопросы далее принимается ответ «НЕТ», хотя возможны и другие варианты ответов, поскольку у физически существующих ЭВМ соответствующие размеры ограничены.

Однако теория, изучающая алгоритмические вычисления, осуществимые в принципе, не должна считаться с такого рода ограничениями, поскольку они преодолимы, по крайней мере, в принципе (например, вообще говоря, любой фиксированный размер памяти всегда можно увеличить на одну ячейку).

Таким образом, уточнение понятия алгоритма связано с уточнением алфавита данных и формы их представления, памяти и размещения в ней данных, элементарных шагов алгоритма и механизма реализации алгоритма. Однако эти понятия сами нуждаются в уточнении. Ясно, что их словесные определения потребуют введения новых понятий, для которых в свою очередь, снова потребуются уточнения и т.д.

В теории алгоритмов принят другой подход, основанный на построении конкретной алгоритмической модели, в которой все сформулированные требования выполняются очевидным образом. При этом используемые алгоритмические модели *универсальны,* т.е. моделируют любые другие разумные алгоритмические модели, что позволяет снять возможное возражение против такого подхода: не приводит ли жесткая фиксация алгоритмической модели к потере общности формализации алгоритма? Поэтому данные алгоритмические модели отождествляются с формальным понятием алгоритма.

Для уточнения понятия алгоритм в теории алгоритмов используется идея построения конкретных алгоритмических моделей, каждая из которых содержит конкретный набор элементарных шагов, способов определения следующего шага и т.д. С теоретической точки зрения наибольший интерес представляют модели, которые были бы одновременно универсальными (т.е. позволяющими описать любой алгоритм) и простыми, содержащими минимум необходимых средств. Требование простоты важно для того, чтобы выделить действительно необходимые элементы и свойства алгоритма и облегчить доказательства общих утверждений об этих свойствах. (В прикладных моделях гораздо важнее удобство программирования и эффективность вычислений, поэтому их средства: набор элементарных шагов и т.п. - намного богаче.)

п.2. Основные типы алгоритмических моделей

Поиск теоретических моделей алгоритмов происходил в трех направлениях, которые и определили три основных класса таких моделей.

Первое направление основано на арифметизации алгоритмов. Ясно, что любые данные можно закодировать числами, и тогда всякое их преобразование станет арифметическим вычислением. Всякий алгоритм в таких моделях вычисляет значение некоторой числовой функции, а его элементарные шаги - это арифметические операции. Последовательности шагов определяются с помощью двух основных способов. Первый способ - **суперпозиция,** т.е. подстановка функций в функции, порождающая математические формулы, в которых порядок действий определяется расстановкой скобок. Например, формула a-b+c/d получается подстановкой в функцию X+Y функций a-b вместо X и c/d вместо Y. Другой способ менее известен, хотя встречается уже в элементарной математике. Это **рекурсия,** т.е. определение очередного значения функции через ранее вычисленные значения этой же функции.

Рассмотрим, например, функцию факториал:

$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$.

С помощью суперпозиции факториал через арифметические операции определить нельзя: дело в том, что "обычные" суперпозиционные формулы содержат фиксированное число операций для любых значений переменных, а при вычислении факториала число умножений растет с ростом *n*. Рекурсивное же определение факториала выглядит так:

$0!=1; \quad (n+1)!= (n+1)\cdot n!$

Структура такого определения функций *(называемого примитивно-рекурсивным)* состоит из двух частей:

- определение значения функции *f* для начального значения аргумента (в нашем примере - для 0) – граничное или якорное условие;
- и определения $f(n+1)$ через $f(n)$ и другую функцию, которая считается определенной ранее (в примере роль этой функции играет умножение).

Для вычисления $f(n+1)$ надо до этого вычислить *f* в предыдущих точках: 1, 2, 3, ..., *n*.

Пример. Вычислить $f(4)$, если $f(x) = \begin{cases} -1 & при \ \ x \le 1 \\ x^2 - (x+1)\cdot f(x-1) & при \ \ x > 1 \end{cases}$

Ответ. $f(4) = 111$.

Существуют и другие виды рекурсии, в которых понятие предыдущих точек определяется более сложно.

Функции, которые можно построить из целых чисел и арифметических операций с помощью суперпозиций и рекурсивных определений, называются **рекурсивными функциями.** Они являются исторически первым уточнением понятия алгоритма, т.е. первой универсальной алгоритмической моделью. Используя эту модель, австрийский математик К.Гедель доказал свою знаменитую теорему о принципиальной неполноте формальных теорий *(Математическая логика)*.

Итак, первое направление связывает понятие алгоритма с традиционным представлением – процедурами вычисления значений числовых функций. Основной теоретической моделью этого типа являются рекурсивные функции - исторически первая формализация понятия алгоритма.

Второе направление основано на простой идее: для того, чтобы алгоритм понимался однозначно, а каждый его шаг можно было считать элементарным и выполнимым, он должен быть представлен так, чтобы его могла выполнять машина. Алгоритм трактуется как некоторое детерминированное устройство, способное выполнять в каждый момент фиксированное множество операций.

Чем проще структура машины и ее действия, тем убедительнее выглядит утверждение, что ее работа и есть выполнение некоторого алгоритма. При этом структура машины должна быть универсальной, такой, чтобы на ней можно было выполнить любой алгоритм.

Эта идея привела к концепциям абстрактной машины как универсальной алгоритмической модели. Действия абстрактной машины сводятся к тому, что она обозревает символы, записанные в памяти (например, на ленте), и в зависимости от своего состояния и того, каков обозреваемый символ, она стирает его или заменяет другим символом или какой-то последовательностью символов, после чего переходит в новое состояние и читает следующий символ.

Концепция абстрактной машины была выдвинута А. Тьюрингом и Э. Постом практически одновременно (в 1936-1937 гг.)

Основной теоретической моделью такого типа является машина Тьюринга, оказавшая существенное влияние на понимание логической природы разрабатываемых ЭВМ. Другой теоретической моделью данного типа является машина произвольного доступа (МПД) - введенная достаточно недавно (в 70-х годах) с целью моделирования реальных вычислительных машин и получения оценок сложности вычислений.

Третье направление, связанное с уточнением понятия "алгоритм", близко ко второму, но отвлекается от конкретных машинных механизмов. Если рассматривать формальные преобразования данных, то описание действий машины превращается в систему подстановок, которые указывают, какие замены символов в последовательности исходных данных надо производить и в каком порядке использовать сами подстановки.

Третий тип алгоритмических моделей - это преобразования слов в произвольных алфавитах, в которых операциями являются замены частей слов другим словом. Основной теоретической моделью этого типа являются нормальные алгоритмы Маркова.

Гипотетическое утверждение о том, что алгоритмическая модель универсальна, означает, что для любой вычислимой функции (и для любого разрешимого множества) существует алгоритм, описываемый средствами этой модели. Доказать это математически нельзя: доказательства имеют дело с формализованными, т.е. точно описанными объектами, а в данном утверждении объект “*любая вычислимая функция*” (“*любое вычислимое множество*”) не является формализованным. Это утверждение больше похоже на естественнонаучные гипотезы типа “*любое физическое явление можно описать средствами механики Ньютона*”, которые принимаются как основополагающий принцип (или тезис) и считаются справедливыми, пока имеется достаточное количество подтверждающих фактов и нет ни одного опровергающего. Аналогичные принципы декларируются и в теории алгоритмов.

Первым был сформулирован **тезис Черча**: "*всякую вычислимую функцию можно представить как рекурсивную*", а затем **тезис Тьюринга:** "*для всякой вычислимой функции можно построить машину Тьюринга, которая ее вычисляет*". Подтверждением справедливости таких тезисов стали строго доказанные теоремы о сводимости одной алгоритмической модели к другой, т.е. теоремы вида: "любую рекурсивную функцию можно вычислить на некоторой машине Тьюринга"; "для любой задачи, решаемой на машине Тьюринга, существует решающий ее нормальный алгоритм Маркова" и т.д.

п.3. Существование универсального алгоритма

В теории алгоритмов установлен важный факт: в универсальной алгоритмической модели всегда существует универсальный алгоритм, т.е. алгоритм, который способен моделировать работу любого другого алгоритма, описанного в этой модели.

Универсальный алгоритм устроен следующим образом: имеется метод кодирования S для любого алгоритма A в данной модели. Если на вход универсального алгоритма U подать код S(A) алгоритма A и исходные данные X, то результат работы U будет равен результату работы A над X: U (S(A),X) = A (X).

По существу, метод кодирования алгоритмов - это язык программирования, а код S(A) - это программа алгоритма A в языке S. Поэтому, концепция универсального алгоритма, существование которого было доказано в 30-х гг. нашего века (раньше, чем появились универсальные ЭВМ), свидетельствует о том, что в основе работы ЭВМ лежат не только физические, но и математические идеи (успехи электроники влияют лишь на быстродействие и размеры ЭВМ).

Теория алгоритмов оказала существенное влияние на развитие ЭВМ и практику программирования. В теории алгоритмов были предугаданы основные концепции, заложенные в аппаратуру и языки программирования ЭВМ. Упоминаемые выше главные алгоритмические модели эквивалентны; но на практике они существенно различаются сложностными эффектами, возникающими при реализации алгоритмов, и породили разные направления в программировании. Так микропрограммирование строится на идеях машин Тьюринга, структурное программирование заимствовало свои конструкции из теории рекурсивных функций, языки символьной обработки информации (РЕФАЛ, ПРОЛОГ) берут начало от нормальных алгоритмов Маркова и систем Поста.

Авторы обзора[*] основных достижений теории алгоритмов пишут: «*Алгоритмические концепции играют в процессе обучения и воспитания современного человека фундаментальную роль, сравнимую лишь с ролью письменности*».

[*] Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. М., 1987. С.230.