Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Ярославский государственный университет им. П. Г. Демидова
Кафедра теоретической информатики

В. С. Рублев

Алгоритмы.
Машины Тьюринга, проверка истинности булевых функций, эффективная реализация множеств на компьютере

(*индивидуальная работа № 10 по дисциплине «Основы дискретной математики»*)

Методические указания

Рекомендовано
Научно-методическим советом университета
для студентов, обучающихся по специальности
Информационные технологии

Ярославль 2010

УДК 519.2
ББК В127я73
Р82

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2009/10 года

Рецензент
кафедра теоретической информатики Ярославского государственного университета им. П. Г. Демидова

Р82 **Рублев, В. С.** Алгоритмы. Машины Тьюринга, проверка истинности булевых функций, эффективная реализация множеств на компьютере: метод. указания / В. С. Рублев; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль: ЯрГУ, 2010. – 44 с.

Методические указания содержат варианты индивидуальных заданий по теме “Алгоритмы”, а также необходимый материал для ее самостоятельного изучения и выполнения индивидуальных заданий. Для качественного усвоения курса в издании даны подробные определения, примеры, иллюстрации и обоснования.

Предназначены для студентов, обучающихся по специальности 010400.62 Информационные технологии (дисциплина “Основы дискретной математики”, блок ОП), очной формы обучения.

УДК 519.2
ББК В174я73

Содержание

1 Проблема определения алгоритма

Дискретное преобразование одного дискретного множества в другое может быть выражено графом и, в частности, сетью из функциональных элементов. Но в этих случаях дискретные множества фиксированы: с их изменением необходимо изменять и граф. Другим способом выражения дискретного преобразования является алгоритм.

Под *алгоритмом* решения задачи принято понимать описание вычислительного процесса, приводящего к ее решению. Этот термин обязан имени арабского математика начала IX века Мухамеда бен Мусы по прозвищу ал-Хорезми (из Хорезма), который в своем трактате «Хисаб ал-джебр вал-мукабала» в словесной форме дал правила решения алгебраических уравнений 1-й и 2-й степени[1].

[1] Термин *алгебра* происходит от второго слова в названии трактата.

С этих пор алгоритм описывается как последовательность вычислительных шагов, каждый из которых определяет элементарные действия над исходными данными задачи и промежуточными величинами, вводимыми в описание алгоритма, а также определяет, какой шаг будет выполняться следующим. Такое определение алгоритма принято называть *неформальным.* До начала XX века казалось, что такого определения вполне достаточно. Но в 1900 году на математическом конгрессе в Париже великий немецкий математик Давид Гильберт в своем докладе о будущем развитии математики определил ряд проблем, которые XIX век оставил XX веку (эти 23 проблемы получили название *проблем Гильберта*). Некоторые из них формулировались как проблемы разработки алгоритмов. Например, десятая проблема Гильберта заключалась в разработке алгоритма решения диофантовых уравнений (алгебраические уравнения или системы уравнений с рациональными коэффициентами, решения которых ищутся в рациональных числах). Долгое время многие из этих проблем не поддавались решению, и к началу 30-х годов XX века возникло сомнение в том, можно ли построить алгоритм для их решения. Но получение математического доказательства невозможности построения алгоритма решения некоторой проблемы требует формализации понятия «алгоритм». Чтобы разобраться в трудностях формализации понятия «алгоритм», прежде всего рассмотрим свойства этого понятия, которые справедливы для указанного неформального определения и должны обязательно быть приняты во внимание при формальном определении.

В первую очередь следует отметить, что каждый алгоритм является способом решения некоторой потенциально *бесконечной совокупности* задач. Действительно, если рассматривать конечную совокупность задач, то, перенумеровав задачи, получив каким-либо образом (не обязательно алгоритмическим) решение каждой из них и перенумеровав эти решения, можно построить способ, который каждой задаче из этой совокупности по ее номеру определяет решение с тем же номером. Такой способ выбора решения не следует понимать как алгоритм. Бесконечная совокупность задач, для которой определяется алгоритм, характеризуется выбором исходных данных каждой ее задачи из некоторого бесконечного набора данных. Отмеченное первое свойство назовем *массовостью* алгоритма.

Второе свойство понятия «алгоритм» характеризует его как совокупность отдельных шагов и называется *дискретностью* алгоритма.

Каждый шаг алгоритма связан с входными данными шага, которыми являются как исходные данные алгоритма, так и выходные данные предыдущих выполненных шагов. Каждый шаг алгоритма связан также с выходными данными шага, которые однозначно образуются из его входных данных элементарным действием. Это характеризует третье и четвертое свойства алгоритма как *детерминированность* и *элементарность* шагов алгоритма.

Результатом выполнения шага алгоритма являются не только выходные данные шага, но и номер следующего выполняемого шага. Во многих случаях следующим при выполнении является шаг, номер которого на 1 больше, чем номер выполняемого шага. Но в некоторых случаях единственное действие алгоритма – определение следующего шага для выполнения. Это пятое свойство алгоритма называется *направленностью* алгоритма.

Число шагов алгоритма при его выполнении должно быть конечным, что составляет его шестое свойство – *конечность* числа шагов[2]. Это не означает, что при выполнении алгоритма каждый шаг должен выполняться только 1 раз. Некоторые шаги могут выполняться лишь при выполнении условия, которое проверяется на предыдущем шаге. Это дополнительное свойство называется *ветвлением* алгоритма. Некоторые шаги алгоритма могут выполняться многократно, если следующим выполняемым шагом становится один из предыдущих шагов. Такое дополнительное свойство называется *цикличностью* алгоритма. Среди шагов алгоритма обязательно должен быть шаг, *прекращающий* выполнение алгоритма. Выходные данные этого шага и являются результатом выполнения алгоритма. Если исходные данные алгоритма таковы, что при его выполнении никогда не выполняется шаг, прекращающий вычисления, то говорят, что алгоритм *зациклился* на этих исходных данных и что алгоритм *недопустим* для этих исходных данных (некорректные данные) [3].

[2] Это свойство иногда называют *результативностью* алгоритма.

[3] Корректный алгоритм должен отвергать некорректные данные.