

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

**В. С. Рублев**

# **Алгоритмы и анализ сложности**

*(индивидуальная работа по дисциплине*

*«Алгоритмы и анализ сложности»)*

*Методические указания*

*Рекомендовано*

*Научно-методическим советом университета  
для студентов, обучающихся по направлениям*

*Информационные технологии,  
Фундаментальная информатика и информационные технологии*

Ярославль 2010

УДК 519.2  
ББК В127я73  
Р82

*Рекомендовано  
Редакционно-издательским советом университета  
в качестве учебного издания. План 2010/11 года*

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

**Рублев, В. С.** Алгоритмы и анализ сложности: метод.  
Р82 указания/ В. С. Рублев; Яросл. гос. ун-т им. П. Г. Демидова.  
– Ярославль: ЯрГУ, 2010. – 54 с.

Методические указания содержат варианты индивидуальных заданий № 1, 2, 3, а также необходимый материал для самостоятельного изучения и выполнения индивидуальных заданий. Для качественного усвоения курса в издании даны подробные определения, примеры, иллюстрации и обоснования.

Предназначены для студентов, обучающихся по направлению 010400.62 Информационные технологии и направлению 010300.62 Фундаментальная информатика и информационные технологии (дисциплина “Алгоритмы и анализ сложности”, блок ОПД), очной формы обучения.

УДК 519.2  
ББК В174я73

© Ярославский  
государственный  
университет  
им. П. Г. Демидова,  
2010

# Оглавление

<b>1</b>	<b>Определение алгоритма и машина Тьюринга</b>	<b>4</b>
1.1	Проблема определения алгоритма . . . . .	4
1.2	Описание машины Тьюринга . . . . .	9
1.3	Тезис Тьюринга . . . . .	13
<b>2</b>	<b>Сложность алгоритмов</b>	<b>19</b>
2.1	Характеристики сложности алгоритмов . . . . .	19
2.2	Определение трудоемкости алгоритма . . . . .	20
2.3	Оценка трудоемкости алгоритма . . . . .	21
2.4	Анализ алгоритмов и методика оценивания трудоемкости	24
2.5	Рекомендации . . . . .	31
2.6	Индивидуальное задание 1 . . . . .	33
2.6.1	Общее задание . . . . .	33
2.6.2	Варианты индивидуального задания 1 . . . . .	33
<b>3</b>	<b>Прогнозирование времени выполнения программы</b>	<b>41</b>
3.1	Проблема прогнозирования времени и подходы к ее решению . . . . .	41
3.2	Индивидуальное задание 2 (программа сортировки/поиска) . . . . .	43
3.2.1	Общее задание . . . . .	43
3.2.2	Варианты индивидуального задания 2 . . . . .	44
3.3	Характеристики временной сложности программы . . .	46
3.4	Индивидуальное задание 3 (прогнозирование времени выполнения процедуры задания 2) . . . . .	52
3.4.1	Общее задание . . . . .	52

# 1 Определение алгоритма и машина Тьюринга

## 1.1 Проблема определения алгоритма

Под *алгоритмом* решения задачи принято понимать описание вычислительного процесса, приводящего к ее решению. Этот термин обязан имени арабского математика начала IX века Мухаммеда 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<sup>1</sup>.

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 этой формализации, прежде всего рассмотрим свойства этого понятия, которые справедливы для указанного неформального определения и должны обязательно быть приняты во внимание при формальном определении.

---

<sup>1</sup> Термин *алгебра* происходит от второго слова в названии трактата.

В первую очередь следует отметить, что каждый алгоритм – способ 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, в качестве которых выступают 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.

Результат выполнения шага алгоритма – 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<sup>2</sup>. Это не означает, что при работе алгоритма каждый шаг должен выполняться ровно 1 раз. Некоторые шаги могут выполняться лишь при условии, которое проверяется на предыдущем шаге. Это дополнительное свойство называется *ветвлением* алгоритма. Некоторые шаги алгоритма могут выполняться многократно, если следующим шагом ста-

---

<sup>2</sup> Это свойство иногда называют *результативностью* алгоритма.