

Оглавление

От автора	11
Вступительное слово Алексея Малеева, основателя Moscow Workshops	11
Отзыв Дмитрия Гришина, основателя Mail.Ru Group	13
Благодарность от редакции	13
Отзыв Нияза Нигматуллина, двукратного чемпиона мира АСМ ICPC 2012 и 2013 годов	14
Предисловие ко второму изданию	15
Предисловие к первому изданию	16
Глава 1. Введение.....	18
1.1. Что такое олимпиадное программирование?	18
1.1.1. Соревнования по программированию	19
1.1.2. Рекомендации желающим поучаствовать	20
1.2. Об этой книге	20
1.3. Сборник задач CSES	22
1.4. Другие ресурсы	24
Глава 2. Техника программирования.....	26
2.1. Языковые средства	26
2.1.1. Ввод и вывод	27
2.1.2. Работа с числами	28
2.1.3. Сокращение кода	31
2.2. Рекурсивные алгоритмы	32
2.2.1. Порождение подмножеств	32
2.2.2. Порождение перестановок	33
2.2.3. Перебор с возвратом	34
2.3. Поразрядные операции	36
2.3.1. Поразрядные операции	38
2.3.2. Представление множеств	40
Глава 3. Эффективность	43
3.1. Временная сложность	43
3.1.1. Правила вычисления	43
3.1.2. Часто встречающиеся оценки временной сложности	46
3.1.3. Оценка эффективности	47
3.1.4. Формальные определения	48
3.2. Примеры проектирования алгоритмов	49
3.2.1. Максимальная сумма подмассивов	49
3.2.2. Задача о двух ферзях	51
3.3. Оптимизация кода	53
3.3.1. Результат работы компилятора	54

3.3.2. Особенности процессора.....	56
Глава 4. Сортировка и поиск.....	59
4.1. Алгоритмы сортировки.....	59
4.1.1. Пузырьковая сортировка	60
4.1.2. Сортировка слиянием.....	61
4.1.3. Нижняя граница временной сложности сортировки	62
4.1.4. Сортировка подсчетом.....	63
4.1.5. Сортировка на практике.....	63
4.2. Решение задач с применением сортировки	66
4.2.1. Алгоритмы заметающей прямой.....	66
4.2.2. Составление расписания.....	67
4.2.3. Работы и сроки исполнения	68
4.3. Двоичный поиск.....	69
4.3.1. Реализация поиска	69
4.3.2. Двоичный поиск по ответу.....	71
Глава 5. Структуры данных.....	74
5.1. Динамические массивы	74
5.1.1. Векторы	74
5.1.2. Итераторы и диапазоны.....	75
5.1.3. Другие структуры данных.....	77
5.2. Множества	78
5.2.1. Множества и мультимножества	78
5.2.2. Отображения.....	80
5.2.3. Очереди с приоритетом	81
5.2.4. Множества, основанные на политиках.....	82
5.3. Эксперименты	83
5.3.1. Сравнение множества и сортировки.....	83
5.3.2. Сравнение отображения и массива.....	84
5.3.3. Сравнение очереди с приоритетом и мультимножества.....	84
Глава 6. Динамическое программирование	86
6.1. Основные понятия	86
6.1.1. Когда жадный алгоритм не работает	86
6.1.2. Нахождение оптимального решения.....	87
6.1.3. Подсчет решений	91
6.2. Другие примеры.....	92
6.2.1. Наибольшая возрастающая подпоследовательность.....	92
6.2.2. Пути на сетке.....	93
6.2.3. Задачи о рюкзаке.....	95
6.2.4. От перестановок к подмножествам	97
6.2.5. Подсчет количества замощений	98
Глава 7. Алгоритмы на графах.....	101
7.1. Основы теории графов	101
7.1.1. Терминология	102

7.1.2. Представление графа.....	104
7.2. Обход графа.....	107
7.2.1. Поиск в глубину.....	107
7.2.2. Поиск в ширину.....	109
7.2.3. Применения.....	110
7.3. Кратчайшие пути.....	111
7.3.1. Алгоритм Беллмана–Форда.....	112
7.3.2. Алгоритм Дейкстры.....	114
7.3.3. Алгоритм Флойда–Уоршелла.....	116
7.4. Ориентированные ациклические графы.....	118
7.4.1. Топологическая сортировка.....	119
7.4.2. Динамическое программирование.....	120
7.5. Графы преемников.....	122
7.5.1. Нахождение преемников.....	123
7.5.2. Обнаружение циклов.....	124
7.6. Минимальные остовные деревья.....	125
7.6.1. Алгоритм Краскала.....	126
7.6.2. Система непересекающихся множеств.....	128
7.6.3. Алгоритм Прима.....	130
Глава 8. Избранные вопросы проектирования алгоритмов.....	132
8.1. Алгоритмы с параллельным просмотром разрезов.....	132
8.1.1. Расстояние Хэмминга.....	132
8.1.2. Подсчет подсеток.....	134
8.1.3. Достижимость в графах.....	135
8.2. Амортизационный анализ.....	136
8.2.1. Метод двух указателей.....	136
8.2.2. Ближайшие меньшие элементы.....	138
8.2.3. Минимум в скользящем окне.....	139
8.3. Нахождение минимальных значений.....	140
8.3.1. Тернарный поиск.....	141
8.3.2. Выпуклые функции.....	142
8.3.3. Минимизация сумм.....	142
Глава 9. Запросы по диапазону.....	144
9.1. Запросы к статическим массивам.....	144
9.1.1. Запросы о сумме.....	144
9.1.2. Запросы о минимуме.....	146
9.2. Древовидные структуры.....	147
9.2.1. Двоичные индексные деревья.....	147
9.2.2. Деревья отрезков.....	150
9.2.3. Дополнительные приемы.....	154
Глава 10. Алгоритмы на деревьях.....	157
10.1. Базовые понятия.....	157
10.1.1. Обход дерева.....	158

10.1.2. Вычисление диаметра	159
10.1.3. Все максимальные пути	161
10.2. Запросы к деревьям	162
10.2.1. Нахождение предков	162
10.2.2. Поддеревья и пути	163
10.2.3. Наименьшие общие предки	165
10.2.4. Объединение структур данных	168
10.3. Более сложные приемы	169
10.3.1. Центроидная декомпозиция	170
10.3.2. Heavy-light декомпозиция	170
Глава 11. Математика	172
11.1. Теория чисел	172
11.1.1. Простые числа и разложение на простые множители	172
11.1.2. Решето Эратосфена	175
11.1.3. Алгоритм Евклида	176
11.1.4. Возведение в степень по модулю	178
11.1.5. Теорема Эйлера	178
11.1.6. Решение уравнений в целых числах	180
11.2. Комбинаторика	181
11.2.1. Биномиальные коэффициенты	181
11.2.2. Числа Каталана	184
11.2.3. Включение-исключение	186
11.2.4. Лемма Бёрнсайда	188
11.2.5. Теорема Кэли	189
11.3. Матрицы	190
11.3.1. Операции над матрицами	190
11.3.2. Линейные рекуррентные соотношения	192
11.3.3. Графы и матрицы	194
11.3.4. Метод Гаусса	196
11.4. Вероятность	199
11.4.1. Операции с событиями	200
11.4.2. Случайные величины	201
11.4.3. Марковские цепи	203
11.4.4. Рандомизированные алгоритмы	205
11.5. Теория игр	207
11.5.1. Состояния игры	207
11.5.2. Игра ним	209
11.5.3. Теорема Шпрага–Гранди	210
11.6. Преобразование Фурье	213
11.6.1. Работа с полиномами	213
11.6.2. Алгоритм БПФ	214
11.6.3. Вычисление свертки	217
Глава 12. Дополнительные алгоритмы на графах	219
12.1. Сильная связность	219

12.1.1. Алгоритм Косарайю	220
12.1.2. Задача 2-выполнимости	221
12.2. Полные пути	223
12.2.1. Эйлеровы пути	223
12.2.2. Гамильтоновы пути	226
12.2.3. Применения	226
12.3. Максимальные потоки	228
12.3.1. Алгоритм Форда–Фалкерсона	229
12.3.2. Непересекающиеся пути	232
12.3.3. Максимальные паросочетания	233
12.3.4. Покрытие путями	235
12.4. Деревья поиска в глубину	237
12.4.1. Двусвязность	238
12.4.2. Эйлеровы подграфы	239
12.5. Потоки минимальной стоимости	240
12.5.1. Алгоритм путей минимальной стоимости	241
12.5.2. Паросочетания минимального веса	243
12.5.3. Улучшение алгоритма	244
Глава 13. Геометрия	247
13.1. Технические средства в геометрии	247
13.1.1. Комплексные числа	247
13.1.2. Точки и прямые	249
13.1.3. Площадь многоугольника	252
13.1.4. Метрики	254
13.2. Алгоритмы на основе заметающей прямой	256
13.2.1. Точки пересечения	256
13.2.2. Задача о ближайшей паре точек	257
13.2.3. Задача о выпуклой оболочке	258
Глава 14. Алгоритмы работы со строками	260
14.1. Базовые методы	260
14.1.1. Префиксное дерево	261
14.1.2. Динамическое программирование	261
14.2. Хеширование строк	263
14.2.1. Полиномиальное хеширование	263
14.2.2. Применения	263
14.2.3. Коллизии и параметры	264
14.3. Z-алгоритм	266
14.3.1. Построение Z-массива	266
14.3.2. Применения	268
14.4. Суффиксные массивы	269
14.4.1. Метод удвоения префикса	269
14.4.2. Поиск образцов	270
14.4.3. LCP-массивы	271

14.5. Строковые автоматы	272
14.5.1. Регулярные языки.....	273
14.5.2. Автоматы для сопоставления с образцом	274
14.5.3. Суффиксные автоматы	276
Глава 15. Дополнительные темы.....	279
15.1. Квадратный корень в алгоритмах.....	279
15.1.1. Структуры данных.....	279
15.1.2. Подалгоритмы.....	281
15.1.3. Целые разбиения.....	283
15.1.4. Алгоритм Мо	285
15.2. И снова о деревьях отрезков.....	286
15.2.1. Ленивое распространение.....	287
15.2.2. Динамические деревья	290
15.2.3. Структуры данных в вершинах	292
15.2.4. Двумерные деревья.....	293
15.3. Дучи	294
15.3.1. Разбиение и слияние	295
15.3.2. Реализация	296
15.3.3. Дополнительные методы.....	298
15.4. Оптимизация динамического программирования.....	298
15.4.1. Трюк с выпуклой оболочкой	299
15.4.2. Оптимизация методом «разделяй и властвуй»	301
15.4.3. Оптимизация Кнута.....	302
15.5. Методы перебора с возвратом	303
15.5.1. Отсечение ветвей дерева поиска	304
15.5.2. Эвристические функции.....	306
15.6. Разное	308
15.6.1. Встреча в середине.....	308
15.6.2. Подсчет подмножеств	309
15.6.3. Параллельный двоичный поиск	310
15.6.4. Динамическая связность.....	312
Приложение. Сведения из математики.....	315
Формулы сумм	315
Множества	317
Математическая логика	318
Функции.....	319
Логарифмы	319
Системы счисления.....	320
Библиография	321
Предметный указатель	323

От автора

I hope this Russian edition of my book will be useful to future competitive programmers in Russia. I appreciate the competitive programming culture in Russia, especially the international training camps which are known for challenging and high-quality problems. Indeed, competitive programming is a way to learn algorithms together with people from different countries. It does not matter where you come from – everybody is interested in creating efficient algorithms, which is the core of computer science.

Antti Laaksonen

Я надеюсь, что русское издание моей книги будет полезно будущим спортивным программистам России. Я ценю российскую культуру олимпиадного программирования, а в особенности международные учебные кэмпы, которые известны своими сложными и интересными задачами. Так спортивное программирование объединяет людей из разных стран в изучении алгоритмов. Не важно, откуда вы, важна ваша заинтересованность в создании эффективных алгоритмов, которые являются основой Computer Science.

Антти Лааксонен

Вступительное слово Алексея Малеева, основателя Moscow Workshops

Про спортивное программирование в России знают мало. А между тем именно российские команды ежегодно выигрывают самое престижное мировое соревнование по программированию для студентов – International Collegiate Programming Contest (ICPC). История участия студентов из российских вузов на старейшем чемпионате ICPC отсчитывается с далекого 1993 года. В 2000 году студенты Санкт-Петербургского государственного университета показали необычайный прогресс и впервые среди российских команд вырвались в чемпионы мира. С этого года российские команды завоевали 33 золотые медали. Для сравнения: студенты из Китая всего 13 раз удаивались золота за этот период, европейские участники – 12, США – всего 7. В мире наиболее сильную подготовку демонстрируют российские, китайские команды, студенты из Азии и Польши.

Секрет успеха наших команд во многом объясняется усиленными тренировками. Но важна система, в рамках которой происходит подготовка. И здесь именно России удалось выстроить сильную тренировочную струк-

туру для олимпиадных программистов. Чемпионов готовят ведущие вузы страны, технические и не только: Университет ИТМО, Московский физико-технический институт, Санкт-Петербургский государственный университет, Московский государственный университет им. М.В. Ломоносова, Уральский федеральный университет, Саратовский университет и др.

Объединившись вместе, консорциум из вышеперечисленных вузов в 2012 году запустил первый глобальный проект по подготовке студентов к чемпионату по спортивному программированию на кампусе МФТИ – **Moscow Workshops**. В формате учебно-тренировочных сборов студенты решают контексты и разбирают их с лучшими тренерами. Важная компонента **Moscow Workshops** – это интернациональность. Кэмпсы открывают свои двери для молодых участников всего мира, предлагая им не только учиться, но и путешествовать, расширять свой кругозор. На каждом воркшопе участники обогащают свои знания о других странах – посещают экскурсии, знакомятся с местной культурой и людьми.

Уровень подготовки воркшопов очень высокий. Можно сказать, что те, кто прошел обучение в **Moscow Workshops** и успешно прошел отбор начальных этапов ICPC, уже представляют элиту мира программирования. За этими ребятами охотятся крупнейшие IT-компании, они получают лучшие предложения о работе.

В 2016 году по образовательной франшизе мы запустили тренировочный лагерь в Гродно (Западная Белоруссия). С тех пор мы открыли регулярно действующие площадки в Барселоне (Испания) и Коллам (Индия), а в этом году запускаем воркшоп в одном из самых восточных городов России – во Владивостоке. На данный момент в сборах уже приняли участие команды 240 университетов из 60 стран Европы, Азии, Южной и Северной Америки, Африки и Австралии. В 2019 году 11 из 12 команд, завоевавших медали в финале чемпионата ICPC, принимали участие в подготовительных сборах **Moscow Workshops**.

Спортивное программирование – это самый перспективный интеллектуальный вид спорта, который можно назвать шахматами будущего. Уже сейчас им увлекаются лучшие умы планеты, и число участников растет год от года. Рост популярности олимпиадного программирования положительно влияет на другие сферы жизнедеятельности человека. Навыки быстрого решения сложнейших задач помогают сегодняшним студентам в будущем справляться с реальными проблемами человечества креативно и эффективно. В ходе соревнований ребята учатся справляться со сложными моральными и психологическими нагрузками. За это время у них вырабатывается навык управлять рисками, ведь, чтобы выйти в финал ICPC, нужно потратить около 5 лет. Не каждому это суждено. Стрессоустойчивость и нацеленность на результат – это те качества, которые необходимы технологическим предпринимателям для запуска собственных проектов.

Система **Moscow Workshops** также включает онлайн-курсы на платфор-

ме Coursera и онлайн-чемпионаты Opencup.ru, а также сборы для школьников по подготовке к IOI – Moscow Workshops Juniors, что в совокупности дает любому студенту из любого уголка мира, вне зависимости от принадлежности к ведущим мировым университетам, реализоваться в области алгоритмов и программирования.

Студентов, которые горят желанием развиваться в программировании и «кодят», не замечая хода времени, мы ждем на кэмпях **Moscow Workshops** и желаем всегда стремиться к результатам, которые кажутся невозможными.

*С уважением, Алексей Малеев,
проректор МФТИ,
основатель Moscow Workshops*

Отзыв Дмитрия Гришина, основателя Mail.Ru Group

В мои студенческие годы всегда было много программирования, и я часто участвовал в различных олимпиадах, но только теперь понимаю, насколько мне тогда не хватало подобной книги. Всем, кто стремится покорить олимп международных чемпионатов по программированию, она точно придется по вкусу и поможет на этом нелегком пути достижения цели. Мы в Mail.Ru Group тоже создали несколько чемпионатов по программированию, которые за последние 7 лет стали популярны не только в России, но и далеко за рубежом – уже более 20 стран принимает участие и больше 175 тысяч человек соревновались в Russian Code Cup, VK Cup, Russian AI Cup, ML BootCamp и др. Уверен, что эта книга поможет и тем, кто собирается принять в них участие тоже.

*Дмитрий Гришин,
основатель, соучредитель и председатель совета директоров Mail.Ru Group,
основатель венчурного фонда Grishin Robotics*

Благодарность от редакции

Редакция издательства «ДМК-Пресс» выражает огромную благодарность Центру развития ИТ-образования при Московском физико-техническом институте и лично его руководителю Алексею Малееву и Mail.Ru Group за помощь в подготовке выпуска этой книги и, конечно, за отличную подготовку наших чемпионов по программированию.

Отзыв Нияза Нигматуллина, двукратного чемпиона мира АСМ ICPC 2012 и 2013 годов

Эта книга помогает познакомиться с олимпиадным программированием. Она рассчитана больше на начинающих либо не очень опытных в этом деле читателей, чем на продвинутых. Здесь подробно описано, как проходят олимпиады, что требуется, в чем их цель. Рассказано, как нужно к ним готовиться, какие качества в себе вырабатывать и какие есть способы тренироваться. Подробно разобраны базовые темы, трюки и алгоритмы. Средние и сложные методы преподносятся без четких доказательств. Присутствует много полезных олимпиадных «трюков», которые редко встречаются в научной литературе. Большая часть популярных алгоритмов и методов, используемых в решении задач на олимпиадах, упомянута в этой книге.

К каждой теме автор приложил код на языке C++, что позволяет лучше понять описанное и увидеть способы реализации. Есть отдельная глава, в которой описываются те конструкции и библиотеки C++, которые часто используются на олимпиадах, и только те, которые необходимы: описываются они без подробностей и большой теории, а только на том уровне, чтобы читатель смог ими воспользоваться. Если читатель хочет разобраться в них подробнее, то следует почитать дополнительную информацию по языку из специальных источников. Кроме того, описанные инструменты языка C++ сравниваются на протяжении всей книги на эффективность и удобство использования на олимпиадах.

В книге раскрыт уникальный опыт участников и тренеров олимпиадного программирования.

По правилам этих соревнований в финалах нельзя участвовать более двух раз. За более чем сорокалетнюю историю этих чемпионатов всего шесть человек стали двукратными чемпионами мира, все из Санкт-Петербурга. Четверо – из Университета ИТМО: Геннадий Короткевич, Нияз Нигматуллин, Евгений Капун и Михаил Кевер, и двое – из СПбГУ: Николай Дуров и Андрей Лопатин.

Предисловие ко второму изданию

Во второе издание книги добавлено несколько новых разделов, в которых рассматриваются темы повышенного уровня: вычисление преобразования Фурье, нахождение потоков минимальной стоимости в графах и использование конечных автоматов в задачах о строках.

Я благодарен Олли Матилайнену, который прочитал большую часть нового материала и высказал много полезных замечаний и предложений.

Антти Лааксонен

*Хельсинки, Финляндия
Февраль 2020*

Предисловие к первому изданию

Эта книга задумана как содержательное введение в современное олимпиадное программирование. Предполагается, что читатель уже знаком с основами программирования, но опыт проектирования алгоритмов или участия в соревнованиях по программированию не обязателен. Поскольку в книге рассматривается широкий круг тем разной степени трудности, она будет полезна как начинающей, так и более искушенной аудитории.

Соревнования по программированию имеют довольно долгую историю. *Международные студенческие олимпиады по программированию (ICPC)* для студентов университетов проводились уже в 1970-х годах, а первая *Международная олимпиада по информатике* для учащихся старших классов состоялась в 1989 году. Ныне оба мероприятия стали регулярными и собирают множество участников со всего мира.

В наши дни олимпиадное программирование популярно как никогда. Важную роль в его распространении сыграл интернет. Существует активное сетевое сообщество увлеченных этим движением программистов, каждую неделю проводятся различные соревнования. Одновременно растет и трудность заданий. Технические приемы, которыми еще несколько лет назад владели лишь лучшие из лучших, стали стандартными инструментами, известными многим.

Своими корнями олимпиадное программирование уходит в научное исследование алгоритмов. Но если ученый приводит доказательство работоспособности своего алгоритма, то олимпиадный программист *реализует* алгоритм и подает его на вход системы оценивания результатов. Эта система прогоняет алгоритм через различные тесты, и если все они проходят, то решение принимается. Это важный элемент олимпиадного программирования, который позволяет *автоматически* получить убедительные аргументы в пользу правильности алгоритма. Вообще, олимпиадное программирование стало отличным способом изучения алгоритмов, т. к. от участника требуется спроектировать алгоритм, который действительно работает, а не ограничиваться наброском идей, может, правильных, а может, и нет.

У олимпиадного программирования есть еще одно достоинство – конкурсные задачи заставляют думать. В формулировках задач не бывает никакого жульничества, в отличие от многих курсов по алгоритмам, где вам предлагают решить красивую задачу, но только последнее предложение звучит, к примеру, так: «*Подсказка*: для решения задачи модифицируйте алгоритм Дейкстры». Ну, а дальше думать особенно нечего, поскольку по-

ход к решению уже известен. В олимпиадном программировании так не бывает. У вас имеется полный комплект инструментов, а уж какими из них воспользоваться, решайте сами.

Решение олимпиадных задач развивает навыки программирования и отладки. Обычно решение засчитывается, только если пройдены все тесты, поэтому программист, стремящийся к успеху, должен реализовывать алгоритм без ошибок. Это умение высоко ценится в программной инженерии, так что неудивительно, что ИТ-компании проявляют интерес к имеющим опыт олимпиадного программирования.

Чтобы стать хорошим олимпиадным программистом, нужно много времени, зато на этом пути можно и многому научиться. Можете быть уверены, что, потратив время на чтение этой книги, решение задач и участие в различных соревнованиях, вы будете лучше понимать, как устроены алгоритмы.

Буду рад вашим отзывам. Вы можете писать мне по адресу ahslaaks@cs.helsinki.fi.

Я благодарен многим людям, приславшим отзывы на черновые редакции этой книги. Они очень помогли сделать книгу лучше. Отдельное спасибо Микко Эрvasti (Mikko Ervasti), Яанне Юннила (Janne Junnila), Яанне Коккала (Janne Kokkala), Туукка Корхонену (Tuukka Korhonen), Патрику Остегярду (Patric Östergård) и Роопе Сальми (Roopo Salmi), написавшим подробные рецензии на рукопись. Я также признателен Саймону Рису (Simon Rees) и Уэйну Уилеру (Wayne Wheeler) из издательства Springer за плодотворное сотрудничество в ходе подготовки книги к печати.

Антти Лааксонен

*Хельсинки, Финляндия
Октябрь, 2017*

Глава 1

Введение

В этой главе рассказывается, что такое олимпиадное программирование, представлен план книги и обсуждаются дополнительные образовательные ресурсы.

В разделе 1.1 описаны элементы олимпиадного программирования, перечислены некоторые популярные соревнования по программированию и даны рекомендации о том, как готовиться к соревнованиям.

В разделе 1.2 обсуждаются цели этой книги и рассматриваемые в ней темы, а также кратко излагается содержание каждой главы.

В разделе 1.3 мы познакомимся со сборником задач CSES. Решение задач параллельно чтению этой книги – отличный способ научиться олимпиадному программированию.

В разделе 1.4 обсуждаются другие книги по олимпиадному программированию и проектированию алгоритмов.

1.1. Что такое олимпиадное программирование?

Олимпиадное программирование состоит из двух частей – проектирования алгоритмов и реализации алгоритмов.

Проектирование алгоритмов. По сути своей, олимпиадное программирование – это придумывание эффективных алгоритмов решения корректно поставленных вычислительных задач. Для проектирования алгоритмов необходимы навыки в решении задач и знание математики. Зачастую решение появляется в результате сочетания хорошо известных методов и новых идей.

Важную роль в олимпиадном программировании играет математика. На самом деле четких границ между проектированием алгоритмов и математикой не существует. Эта книга не требует глубокой математической подготовки. В приложении, которое можно использовать как справочник, описаны некоторые встречающиеся в книге математические понятия и методы, например множества, математическая логика и функции.

Реализация алгоритмов. В олимпиадном программировании решение задачи оценивается путем проверки реализованного алгоритма на ряде тестов. Поэтому придумать алгоритм недостаточно, его еще нуж-

но корректно реализовать, для чего требуется умение программировать. Олимпиадное программирование сильно отличается от традиционной программной инженерии: программы короткие (несколько сотен строк – уже редкость), писать их нужно быстро, а сопровождение после соревнования не требуется.

В настоящее время на соревнованиях по программированию популярнее всего языки C++, Python и Java. Например, среди 3000 лучших участников Google Code Jam 2017 79% писали на C++, 16% – на Python и 8% – на Java. Многие считают C++ самым подходящим выбором для олимпиадного программиста. К его достоинствам можно отнести очень высокую эффективность и тот факт, что в стандартной библиотеке много разнообразных структур данных и алгоритмов.

Все примеры в книге написаны на C++, в них часто используются структуры данных и алгоритмы из стандартной библиотеки. Код отвечает стандарту C++11, который разрешен в большинстве современных соревнований. Если вы еще не знаете C++, самое время начать его изучение.

1.1.1. Соревнования по программированию

Международная олимпиада по информатике (IOI) – ежегодное соревнование для старшеклассников. От каждой страны допускается команда из четырех человек. Обычно набирается около 300 участников из 80 стран.

IOI проводится в течение двух дней. В каждый день участникам предлагается решить три трудные задачи, на решение отводится пять часов. Задачи разбиты на подзадачи, за каждую из которых начисляются баллы. Хотя участники являются членами команды, соревнуются они самостоятельно.

Участники IOI отбираются на национальных олимпиадах. IOI предшествует множество региональных соревнований, например: Балтийская олимпиада по информатике (BOI), Центрально-Европейская олимпиада по информатике (CEOI) и Азиатско-Тихоокеанская олимпиада по информатике (APIO).

ICPC (Международная студенческая олимпиада по программированию) проводится ежегодно для студентов университетов. В каждой команде три участника; в отличие от IOI, студенты работают вместе, и каждой команде выделяется только один компьютер.

ICPC включает несколько этапов, лучшие команды приглашаются на финальный этап мирового первенства. Хотя в соревновании принимают участие тысячи студентов, количество участников финала ограничено¹, поэтому даже сам выход в финал считается большим достижением.

На соревновании ICPC у команды есть пять часов на решение примерно десяти алгоритмических задач. Решение засчитывается, только если прог-

¹ Точное количество участников финала от года к году меняется. В 2017 году их было 133.

рамма прошла все тесты и показала свою эффективность. В ходе соревнования участники могут видеть результаты соперников, но в начале пятого часа табло замораживается, и результаты последних прогонов не видны.

Онлайновые соревнования. Существует также много онлайн-овых соревнований, куда допускаются все желающие. В настоящий момент наиболее активен сайт Codeforces, который организует конкурсы почти каждую неделю. Отметим также сайты AtCoder, CodeChef, CS Academy, HackerRank и Topcoder.

Некоторые компании организуют онлайн-овые соревнования с очными финалами, например: Facebook Hacker Cup, Google Code Jam и Yandex.Algorithm. Разумеется, компании используют такие соревнования для подбора кадров: достойно выступить в соревновании – хороший способ доказать свои таланты в программировании.

1.1.2. Рекомендации желающим поучаствовать

Чтобы научиться олимпиадному программированию, нужно напряженно работать. Но способов приобрести практический опыт много, одни лучше, другие хуже.

Решая задачи, нужно иметь в виду, что не так важно *количество* решенных задач, как их *качество*. Возникает соблазн выбирать красивые и легкие задачи, пропуская те, что кажутся трудными и требующими кропотливого труда. Но чтобы отточить свои навыки, нужно отдавать предпочтение именно задачам второго типа.

Важно и то, что большинство олимпиадных задач решается с помощью простого и короткого алгоритма, самое трудное – придумать этот алгоритм. Смысл олимпиадного программирования – не в заучивании сложных и малопонятных алгоритмов, а в том, чтобы научиться решать трудные задачи, обходясь простыми средствами.

Наконец, есть люди, презирающие реализацию алгоритмов, им доставляет удовольствие проектировать алгоритмы, а реализовывать их скучно. Однако умение быстро и правильно реализовать алгоритм – важное преимущество, и этот навык поддается тренировке. Будет очень плохо, если вы потратите большую часть отведенного времени на написание кода и поиск ошибок, а не на обдумывание того, как решить задачу.

1.2. Об этой книге

Программа IOI [17] определяет, какие темы могут предлагаться на Международных олимпиадах по информатике. Именно эта программа стала отправной точкой при отборе материала. Но обсуждаются также более сложные темы, которые исключены из IOI (по состоянию на 2017 год), но могут встречаться в других соревнованиях. К ним относятся, например, максимальные потоки, игра ним и суффиксные массивы.

Многие темы олимпиадного программирования обсуждаются в стандартных учебниках по алгоритмам, но есть и отличия. Например, во многих книгах реализация алгоритмов сортировки и основных структур данных рассматривается с нуля, но такие знания на олимпиаде несущественны, потому что можно пользоваться средствами из стандартной библиотеки. С другой стороны, некоторые темы хорошо известны в олимпиадном сообществе, но редко встречаются в учебниках. Примером может служить дерево отрезков – структура данных, которая используется для решения многих задач без привлечения более хитроумных алгоритмов.

Одна из целей этой книги – *документировать* приемы олимпиадного программирования, которые обычно обсуждаются только на форумах и в блогах. Там, где возможно, даются ссылки на научную литературу по таким методам. Но сделать это можно не всегда, потому что многие методы ныне стали частью *фольклора*, и никто не знает имени первооткрывателя.

Книга организована следующим образом.

- В главе 2 рассматриваются средства языка программирования C++, а затем обсуждаются рекурсивные алгоритмы и поразрядные операции.
- Глава 3 посвящена эффективности: как создавать алгоритмы, способные быстро обрабатывать большие наборы данных.
- В главе 4 обсуждаются алгоритмы сортировки и двоичного поиска с упором на их применение в проектировании алгоритмов.
- В главе 5 дается обзор избранных структур данных в стандартной библиотеке C++: векторов, множеств и отображений.
- Глава 6 представляет собой введение в один из методов проектирования алгоритмов – динамическое программирование. Здесь же приводятся примеры задач, которые можно решить этим методом.
- В главе 7 рассматриваются элементарные алгоритмы на графах, в т. ч. поиск кратчайших путей и минимальные остовные деревья.
- В главе 8 речь пойдет о более сложных вопросах проектирования алгоритмов, в частности об алгоритмах с параллельным просмотром разрядов и амортизационном анализе.
- Тема главы 9 – эффективная обработка запросов по диапазонам массива, таких как вычисление сумм элементов и нахождение минимальных элементов.
- В главе 10 представлены специальные алгоритмы для деревьев, в т. ч. методы обработки запросов к дереву.
- В главе 11 обсуждаются разделы математики, часто встречающиеся в олимпиадном программировании.

- В главе 12 описываются дополнительные алгоритмы на графах, например поиск компонент сильной связности и вычисление максимального потока.
- Глава 13 посвящена геометрическим алгоритмам, в ней описаны методы, позволяющие удобно решать геометрические задачи.
- В главе 14 рассматриваются методы работы со строками, в т. ч. хеширование, Z-алгоритм и суффиксные массивы.
- В главе 15 обсуждаются избранные дополнительные темы, например алгоритмы, основанные на идее квадратного корня, и оптимизация динамического программирования.

1.3. Сборник задач CSES

В сборнике задач CSES представлены задачи для практики в олимпиадном программировании. Расположены они в порядке возрастания трудности, а в этой книге обсуждаются все методы, необходимые для их решения. Сборник задач доступен по адресу:

<https://cses.fi/problemset/>.

Посмотрим, как решить первую задачу из него. Она называется «Странный алгоритм» и формулируется следующим образом.

Рассмотрим алгоритм, принимающий на входе целое положительное число n . Если n чётно, то алгоритм делит его на два, а если нечётно, то умножает на три и прибавляет 1. Например, для $n = 3$ получается следующая последовательность:

$$3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1.$$

Ваша задача заключается в том, чтобы промоделировать выполнение этого алгоритма для любого заданного n .

Вход

Единственная строка, содержащая целое число n .

Выход

Печатает строку, содержащую все значения, вычисляемые алгоритмом.

Ограничения

- $1 \leq n \leq 10^6$

Пример

Вход:

3

Выход:

3 10 5 16 8 4 2 1

Эта проблема имеет отношение к знаменитой *гипотезе Коллатца*, которая утверждает, что описанный выше алгоритм завершается для любого n . До сих пор она остается недоказанной. Но в этой задаче мы знаем, что начальное значение n не превышает миллиона, что существенно упрощает проблему.

Это простая задача моделирования, не требующая глубоких размышлений. Вот как ее можно было бы решить на C++:

```
#include <iostream>

using namespace std;

int main() {
    int n;
    cin >> n;
    while (true) {
        cout << n << " ";
        if (n == 1) break;
        if (n%2 == 0) n /= 2;
        else n = n*3+1;
    }
    cout << "\n";
}
```

Сначала программа читает входное число n , затем моделирует алгоритм и печатает значение n после каждого шага. Легко проверить, что этот код правильно обрабатывает случай $n = 3$, приведенный в формулировке задачи.

Теперь время *отправить* этот код в CSES. Для каждого теста CSES сообщает, пройден он или нет, и показывает вход, ожидаемый и реальный выход.

По результатам тестирования CSES формирует следующий отчет:

test	verdict	time (s)
#1	ACCEPTED	0.06 / 1.00
#2	ACCEPTED	0.06 / 1.00
#3	ACCEPTED	0.07 / 1.00
#4	ACCEPTED	0.06 / 1.00
#5	ACCEPTED	0.06 / 1.00
#6	TIME LIMIT EXCEEDED	_ / 1.00
#7	TIME LIMIT EXCEEDED	_ / 1.00
#8	WRONG ANSWER	0.07 / 1.00
#9	TIME LIMIT EXCEEDED	_ / 1.00
#10	ACCEPTED	0.06 / 1.00

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru