

Оглавление

Вступительное слово Алексея Малеева, основателя Moscow Workshops ICPC.....	11
Отзыв Нияза Нигматуллина, двукратного чемпиона мира АСМ ICPC 2012 и 2013 годов.....	13
От автора	14
Благодарность от редакции	14
Предисловие.....	15
Глава 1. Введение.....	17
1.1. Что такое олимпиадное программирование?	17
1.1.1. Соревнования по программированию	18
1.1.2. Рекомендации желающим поучаствовать.....	19
1.2. Об этой книге	19
1.3. Сборник задач CSES	21
1.4. Другие ресурсы	23
Глава 2. Техника программирования.....	25
2.1. Языковые средства	25
2.1.1. Ввод и вывод.....	26
2.1.2. Работа с числами.....	27
2.1.3. Сокращение кода	30
2.2. Рекурсивные алгоритмы.....	31
2.2.1. Порождение подмножеств.....	31
2.2.2. Порождение перестановок.....	32
2.2.3. Перебор с возвратом.....	33
2.3. Поразрядные операции.....	35
2.3.1. Поразрядные операции.....	37
2.3.2. Представление множеств	39
Глава 3. Эффективность	42
3.1. Временная сложность.....	42
3.1.1. Правила вычисления	42
3.1.2. Часто встречающиеся оценки временной сложности	45
3.1.3. Оценка эффективности.....	46
3.1.4. Формальные определения	47
3.2. Примеры.....	48
3.2.1. Максимальная сумма подмассивов.....	48
3.2.2. Задача о двух ферзях.....	50
Глава 4. Сортировка и поиск.....	53
4.1. Алгоритмы сортировки.....	53
4.1.1. Пузырьковая сортировка	54

4.1.2. Сортировка слиянием.....	55
4.1.3. Нижняя граница временной сложности сортировки	56
4.1.4. Сортировка подсчетом.....	57
4.1.5. Сортировка на практике.....	57
4.2. Решение задач с применением сортировки	60
4.2.1. Алгоритмы заметающей прямой.....	60
4.2.2. Составление расписания.....	61
4.2.3. Работы и сроки исполнения	62
4.3. Двоичный поиск.....	63
4.3.1. Реализация поиска	63
4.3.2. Нахождение оптимальных решений.....	65
Глава 5. Структуры данных.....	68
5.1. Динамические массивы	68
5.1.1. Векторы	68
5.1.2. Итераторы и диапазоны	69
5.1.3. Другие структуры данных.....	71
5.2. Множества	72
5.2.1. Множества и мультимножества	72
5.2.2. Отображения.....	74
5.2.3. Очереди с приоритетом	75
5.2.4. Множества, основанные на политиках.....	76
5.3. Эксперименты	77
5.3.1. Сравнение множества и сортировки.....	77
5.3.2. Сравнение отображения и массива.....	78
5.3.3. Сравнение очереди с приоритетом и мультимножества.....	78
Глава 6. Динамическое программирование	80
6.1. Основные понятия	80
6.1.1. Когда жадный алгоритм не работает	80
6.1.2. Нахождение оптимального решения.....	81
6.1.3. Подсчет решений	85
6.2. Другие примеры.....	86
6.2.1. Наибольшая возрастающая подпоследовательность.....	86
6.2.2. Пути на сетке.....	87
6.2.3. Задачи о рюкзаке.....	89
6.2.4. От перестановок к подмножествам	91
6.2.5. Подсчет количества замощений	92
Глава 7. Алгоритмы на графах.....	95
7.1. Основы теории графов	95
7.1.1. Терминология	96
7.1.2. Представление графа.....	98
7.2. Обход графа	101
7.2.1. Поиск в глубину.....	101

7.2.2. Поиск в ширину	103
7.2.3. Применения	104
7.3. Кратчайшие пути	105
7.3.1. Алгоритм Беллмана–Форда	106
7.3.2. Алгоритм Дейкстры	108
7.3.3. Алгоритм Флойда–Уоршелла	110
7.4. Ориентированные ациклические графы	112
7.4.1. Топологическая сортировка	113
7.4.2. Динамическое программирование	114
7.5. Графы преемников	116
7.5.1. Нахождение преемников	117
7.5.2. Обнаружение циклов	118
7.6. Минимальные остовные деревья	119
7.6.1. Алгоритм Краскала	120
7.6.2. Система непересекающихся множеств	122
7.6.3. Алгоритм Прима	124
Глава 8. Избранные вопросы проектирования алгоритмов	126
8.1. Алгоритмы с параллельным просмотром разрядов	126
8.1.1. Расстояние Хэмминга	126
8.1.2. Подсчет подсеток	128
8.1.3. Достижимость в графах	129
8.2. Амортизационный анализ	130
8.2.1. Метод двух указателей	130
8.2.2. Ближайшие меньшие элементы	132
8.2.3. Минимум в скользящем окне	133
8.3. Нахождение минимальных значений	134
8.3.1. Троичный поиск	135
8.3.2. Выпуклые функции	136
8.3.3. Минимизация сумм	136
Глава 9. Запросы по диапазону	138
9.1. Запросы к статическим массивам	138
9.1.1. Запросы о сумме	138
9.1.2. Запросы о минимуме	140
9.2. Древовидные структуры	141
9.2.1. Двоичные индексные деревья	141
9.2.2. Деревья отрезков	144
9.2.3. Дополнительные приемы	148
Глава 10. Алгоритмы на деревьях	151
10.1. Базовые понятия	151
10.1.1. Обход дерева	152
10.1.2. Вычисление диаметра	153
10.1.3. Все максимальные пути	155

10.2. Запросы к деревьям	156
10.2.1. Нахождение предков	156
10.2.2. Поддеревья и пути	157
10.2.3. Наименьшие общие предки	159
10.2.4. Объединение структур данных	162
10.3. Более сложные приемы	163
10.3.1. Центроидная декомпозиция	164
10.3.2. Разновесная декомпозиция	164
Глава 11. Математика	166
11.1. Теория чисел	166
11.1.1. Простые числа и разложение на простые множители	166
11.1.2. Решето Эратосфена	169
11.1.3. Алгоритм Евклида	170
11.1.4. Возведение в степень по модулю	172
11.1.5. Теорема Эйлера	172
11.1.6. Решение уравнений в целых числах	174
11.2. Комбинаторика	175
11.2.1. Биномиальные коэффициенты	175
11.2.2. Числа Каталана	178
11.2.3. Включение-исключение	180
11.2.4. Лемма Бёрнсайда	182
11.2.5. Теорема Кэли	183
11.3. Матрицы	184
11.3.1. Операции над матрицами	184
11.3.2. Линейные рекуррентные соотношения	186
11.3.3. Графы и матрицы	188
11.3.4. Метод исключения Гаусса	190
11.4. Вероятность	193
11.4.1. Операции с событиями	194
11.4.2. Случайные величины	195
11.4.3. Марковские цепи	197
11.4.4. Рандомизированные алгоритмы	199
11.5. Теория игр	201
11.5.1. Состояния игры	201
11.5.2. Игра ним	203
11.5.3. Теорема Шпрага–Гранди	204
Глава 12. Дополнительные алгоритмы на графах	208
12.1. Сильная связность	208
12.1.1. Алгоритм Косарайо	209
12.1.2. Задача 2-выполнимости	210
12.2. Полные пути	212
12.2.1. Эйлеровы пути	212
12.2.2. Гамильтоновы пути	215

12.2.3. Применения	215
12.3. Максимальные потоки	217
12.3.1. Алгоритм Форда–Фалкерсона	218
12.3.2. Непересекающиеся пути	221
12.3.3. Максимальные паросочетания	222
12.3.4. Покрытие путями	224
12.4. Деревья поиска в глубину	226
12.4.1. Двусвязность	227
12.4.2. Эйлеровы подграфы	228
Глава 13. Геометрия	230
13.1. Технические средства в геометрии	230
13.1.1. Комплексные числа	230
13.1.2. Точки и прямые	232
13.1.3. Площадь многоугольника	235
13.1.4. Метрики	237
13.2. Алгоритмы на основе заметающей прямой	239
13.2.1. Точки пересечения	239
13.2.2. Задача о ближайшей паре точек	240
13.2.3. Задача о выпуклой оболочке	241
Глава 14. Алгоритмы работы со строками	243
14.1. Базовые методы	243
14.1.1. Префиксное дерево	244
14.1.2. Динамическое программирование	244
14.2. Хэширование строк	246
14.2.1. Полиномиальное хэширование	246
14.2.2. Применения	246
14.2.3. Коллизии и параметры	247
14.3. Z-алгоритм	249
14.3.1. Построение Z-массива	249
14.3.2. Применения	251
14.4. Суффиксные массивы	252
14.4.1. Метод удвоения префикса	252
14.4.2. Поиск образцов	253
14.4.3. LCP-массивы	254
Глава 15. Дополнительные темы	256
15.1. Квадратный корень в алгоритмах	256
15.1.1. Структуры данных	256
15.1.2. Подалгоритмы	258
15.1.3. Целые разбиения	260
15.1.4. Алгоритм Мо	262
15.2. И снова о деревьях отрезков	263
15.2.1. Ленивое распространение	264

15.2.2. Динамические деревья	267
15.2.3. Структуры данных в вершинах	269
15.2.4. Двумерные деревья.....	270
15.3. Дучи	271
15.3.1. Разбиение и слияние	272
15.3.2. Реализация	273
15.3.3. Дополнительные методы.....	275
15.4. Оптимизация динамического программирования.....	275
15.4.1. Трюк с выпуклой оболочкой	276
15.4.2. Оптимизация методом «разделяй и властвуй»	278
15.4.3. Оптимизация Кнута.....	279
15.5. Разное	280
15.5.1. Встреча в середине.....	281
15.5.2. Подсчет подмножеств	281
15.5.3. Параллельный двоичный поиск	283
15.5.4. Динамическая связность.....	284
Приложение. Сведения из математики.....	287
Формулы сумм	287
Множества	289
Математическая логика	290
Функции.....	291
Логарифмы.....	291
Системы счисления.....	292
Библиография	293
Предметный указатель	295

Вступительное слово

Алексея Малеева, основателя

Moscow Workshops ICPC

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 32 золотые медали. Для сравнения: студенты из Китая всего 13 раз удостоивались золота за этот период, европейские участники – 11, США – всего 6. 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 структуру 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. 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 ICPC**. В формате учебно-тренировочных сборов студенты решают контексты и разбирают их с лучшими тренерами. Важная компонента **Moscow Workshops ICPC** – это интернациональность. Кэмп-пы 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 ICPC** 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 167 университетов из 50 стран Европы, Азии, Южной и Северной Америки, Африки и Австралии. В 2016 и 2017 годах восемь из двенадцати команд, завоевавших медали в финале чемпионата ICPC, принимали участие в подготовительных сборах **Moscow Workshops ICPC**. В 2018 году их число выросло – 10 из 13 медалистов готовились на воркшопах.

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 ICPC** также включает онлайн-курсы на платформе Coursera и онлайн-чемпионаты Opencup.ru, что в совокупности дает любому студенту из любого уголка мира, вне зависимости от принадлежности к ведущим мировым университетам, реализовать в области алгоритмов и программирования.

Уверены, что последователей ICPC будет все больше. Всех студентов, которые горят желанием развиваться в программировании и «кодят», не замечая хода времени, мы ждем на кэмпях **Moscow Workshops ICPC** и желаем всегда стремиться к результатам, которые кажутся невозможными.

С уважением,

Алексей Малеев,

*директор по технологическому предпринимательству МФТИ,
основатель Moscow Workshops ICPC*

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.

От автора

Я надеюсь, что русское издание моей книги будет полезно будущим спортивным программистам России. Я ценю российскую культуру олимпиадного программирования, а в особенности международные учебные кэмп-пы, 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

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 за помощь в подготовке выпуска этой книги и, конечно, за отличную подготовку наших чемпионов по программированию.

Предисловие

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 подход 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 (Roope 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 (APOI).

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 году 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 [15] 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. хэширование, 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 эта 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

Это означает, что некоторые тесты наш код прошел (ACCEPTED), на некоторых оказался слишком медленным (TIME LIMIT EXCEEDED), а в одном случае дал неверный результат (WRONG ANSWER). Вот уж действительно неожиданно!

Первым не прошел тест для $n = 138\,367$. Локально прогнав программу в этом случае, мы убедимся, что она действительно работает долго. На самом деле она вообще не завершается.

Причина ошибки в том, что в процессе моделирования n может оказаться слишком большим, в том числе больше максимального значения, представимого типом `int`. Чтобы решить проблему, достаточно заменить тип `int` на `long long`. Тогда получится то, что нужно:

test	verdict	time (s)
#1	ACCEPTED	0.05 / 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	ACCEPTED	0.05 / 1.00
#7	ACCEPTED	0.06 / 1.00
#8	ACCEPTED	0.05 / 1.00
#9	ACCEPTED	0.07 / 1.00
#10	ACCEPTED	0.06 / 1.00

Как видно из этого примера, даже в очень простые алгоритмы могут вкрасться тонкие ошибки. Олимпиадное программирование учит, как писать алгоритмы, которые действительно работают.

1.4. Другие ресурсы

Помимо этой, уже есть и другие книги по олимпиадному программированию. Первой была книга Skiena, Revilla «Programming Challenges» [28], вышедшая в 2003 году. Позже вышла книга Halim, Halim «Competitive Programming 3» [14]. Обе ориентированы на читателя, не имеющего опыта олимпиадного программирования.

Книга «Looking for a Challenge?» [7] – сборник трудных задач, предлагавшихся на польских олимпиадах, – рассчитана на более подготовленного читателя. Самое интересное в ней – подробный анализ решений.

Разумеется, для подготовки к олимпиадам подойдут и общие книги по алгоритмам. Самая всеобъемлющая из них – книга Cormen, Leiserson,

Rivest, Stein «Introduction to Algorithms»² [6], которую также называют просто *CLRS*. Это прекрасный источник для тех, кто хочет узнать обо всех деталях алгоритма и познакомиться со строгим доказательством.

В книге Kleinberg, Tardos «Algorithm Design» [19] основное внимание уделяется технике проектирования алгоритмов и во всех деталях обсуждаются такие темы, как метод «разделяй и властвуй», жадные алгоритмы, динамическое программирование и вычисление максимального потока. Книга Skiena «The Algorithm Design Manual» [27] – практическое руководство, содержащее обширный каталог вычислительных задач и способов их решения.

² Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы. Построение и анализ. – М.: Вильямс, 2016.