Оглавление

Предисловие 22
Вступление 23
Благодарности 25
О книге 27
Кому адресована эта книга 28
Структура книги 29
О примерах программного кода 31
От издательства 32
Об авторе 33
Иллюстрация на обложке 34
Глава 1. Введение в структуры данных 35
1.1. Структуры данных 36
1.1.1. Определение структуры данных 37
1.1.2. Описание структуры данных 39
1.1.3. Алгоритмы и структуры данных: так есть ли разница? 40
1.2. Целеполагание: ваши ожидания от прочтения этой книги 41
1.3. Собираем рюкзак: структуры данных в реальном мире 42
1.3.1. Абстрагирование задачи 42
1.3.2. Поиск решений 43
1.3.3. Спасительные алгоритмы 45
1.3.4. Выходим за рамки (в буквальном смысле) 46
1.3.5. Счастливый конец 47
Резюме 48

Часть I
Улучшаем базовые структуры данных

Глава 2. Улучшаем очереди с приоритетом: d-кучи 51
2.1. Структура этой главы 52
2.2. Задача: как работать с приоритетами?. 53
2.2.1. Приоритеты на практике: отслеживание ошибок. 53
2.3. Простое решение: храним отсортированный список. 55
2.3.1. От отсортированных списков к очередям с приоритетом 55
2.4. Описание API структуры данных: очереди с приоритетом. 56
2.4.1. Примеры работы очереди с приоритетом. 57
2.4.2. Приоритет имеет значение: обобщаем FIFO. 59
2.5. Конкретные структуры данных 59
2.5.1. Сравнение производительности 60
2.5.2. Какая конкретная структура данных наиболее верна? 60
2.5.3. Куча 61
2.5.4. Приоритет, неубывающая и невозрастающая куча. 63
2.5.5. Продвинутый вариант: d-куча 65
2.6. Как реализовать кучу 66
2.6.1. BubbleUp. 67
2.6.2. PushDown 72
2.6.3. Вставка 75
2.6.4. Метод top 78
2.6.5. Метод update 81
2.6.6. Обработка дубликатов 82
2.6.7. Преобразование в кучу 83
2.6.8. Методы вне API: contains 86
2.6.9. Еще раз о производительности 86
2.6.10. От псевдокода к реализации 88
2.7. Пример: поиск k наибольших элементов 88
2.7.1. Правильная структура данных... 89
2.7.2. ...И правильное ее использование 89
2.7.3. Реализация 90
2.8. Другие случаи использования. 90
2.8.1. Поиск кратчайшего пути в графе: алгоритм Дейкстры 91
2.8.2. Еще о графах: алгоритм Прима 91
2.8.3. Сжатие данных: коды Хаффмана. 92

2.9. Анализ коэффициента ветвления 97
2.9.1. Нужны ли d-ичные кучи? 97
2.9.2. Время выполнения. 99
2.9.3. Поиск оптимального коэффициента ветвления 99
2.9.4. Коэффициент ветвления и память 100
2.10. Анализ производительности: поиск наилучшего коэффициента ветвления 101
2.10.1. Добро пожаловать в профилирование 102
2.10.2. Интерпретация результатов 105
2.10.3. Загадка _heapify. 109
2.10.4. Выбор лучшего коэффициента ветвления 111
Резюме 112

Глава 3. Декартовы деревья: применение рандомизации для получения сбалансированных двоичных деревьев поиска 113
3.1. Задача: множественная индексация 114
3.1.1. Суть решения 115
3.2. Решение: описание и API. 116
3.3. Декартово дерево 117
3.3.1. Поворот. 120
3.3.2. Несколько вопросов по организации 125
3.3.3. Реализация поиска. 127
3.3.4. Вставка 128
3.3.5. Удаление 132
3.3.6. Методы top, peek и update 135
3.3.7. Методы min и max 135
3.3.8. Еще раз о производительности 136
3.4. Применение: рандомизированные декартовы деревья 137
3.4.1. Сбалансированные деревья 138
3.4.2. Введение в рандомизацию 141
3.4.3. Применение рандомизированных декартовых деревьев 143
3.5. Анализ производительности и профилирование 143
3.5.1. Теория: ожидаемая высота. 144
3.5.2. Экспериментальное определение высоты 147
3.5.3. Профилирование времени выполнения 151
3.5.4. Профилирование потребления памяти 153
3.5.5. Выводы 155
Резюме 157

Глава 4. Фильтры Блума: отслеживание содержимого с меньшими затратами памяти 158
4.1. Словари: отслеживание содержимого 159
4.2. Альтернативы реализации словаря 161
4.3. Описание API структуры данных: ассоциативный массив 162
4.4. Конкретные структуры данных 163
4.4.1. Несортированный массив: быстрая вставка, медленный поиск 164
4.4.2. Отсортированные массивы и поиск методом дихотомии: медленная вставка, быстрый поиск 164
4.4.3. Хеш-таблица: в среднем время постоянно, если порядок неважен . . . 166
4.4.4. Двоичное дерево поиска: логарифмическое время выполнения всех операций 167
4.4.5. Фильтр Блума: так же быстр, как хеш-таблицы, при этом экономит память (но есть подвох) 168
4.5. За кулисами: как работают фильтры Блума 169
4.6. Реализация. 171
4.6.1. Использование фильтра Блума. 172
4.6.2. Чтение и запись битов 173
4.6.3. Поиск места, где хранится ключ 175
4.6.4. Генерирование хеш-функций 176
4.6.5. Конструктор. 176
4.6.6. Проверка ключа 178
4.6.7. Сохранение ключа 180
4.6.8. Оценка точности 182
4.7. Применение 182
4.7.1. Кэш 182
4.7.2. Маршрутизаторы 184
4.7.3. Интернет-роботы. 184
4.7.4. Сборщик операций ввода/вывода 185
4.7.5. Проверка орфографии 185
4.7.6. Распределенные базы данных и файловые системы 185
4.8. Как работают фильтры Блума. 185
4.8.1. Почему невозможны ложноотрицательные результаты проверок... . . . 188
4.8.2. ...Но ложноположительные возможны 188
4.8.3. Фильтры Блума как рандомизированные алгоритмы 189
4.9. Анализ производительности. 189
4.9.1. Время выполнения. 189
4.9.2. Конструктор. 190

4.9.3. Сохранение элемента 190
4.9.4. Поиск элемента 191
4.10. Оценка точности фильтра Блума 191
4.10.1. Объяснение формулы отношения ложноположительных результатов 193
4.11. Улучшенные варианты 196
4.11.1. Фильтр Блумьера 196
4.11.2. Комбинирование фильтров Блума 196
4.11.3. Многослойный фильтр Блума 197
4.11.4. Сжатый фильтр Блума 197
4.11.5. Масштабируемый фильтр Блума 198
Резюме 198

Глава 5. Непересекающиеся множества: обработка за сублинейное время 200
5.1. Задача разделения на подмножества 201
5.2. Размышления о возможных решениях 204
5.3. Описание API структуры данных: непересекающееся множество 206
5.4. Простейшее решение 208
5.4.1. Реализация простейшего решения 209
5.5. Использование древовидной структуры 213
5.5.1. От списка к деревьям 213
5.5.2. Реализация версии с деревьями 215
5.6. Эвристика для улучшения времени выполнения 217
5.6.1. Сжатие пути 219
5.6.2. Реализация балансировки и сжатия пути 221
5.7. Применение 224
5.7.1. Графы: связанные компоненты 224
5.7.2. Графы: алгоритм Краскала для минимального остовного дерева 224
5.7.3. Кластеризация 225
5.7.4. Унификация 227
Резюме 227

Глава 6. Префиксные деревья, компактные префиксные деревья: эффективный поиск строк 228
6.1. Проверка орфографии 229
6.1.1. Прнцесса, Деймон и эльф звходят в бар 230
6.1.2. Сжатие — ключ к успеху 232
6.1.3. Описание и API 232

6.2. Префиксное дерево 233
6.2.1. Почему снова лучше? 237
6.2.2. Поиск 239
6.2.3. Вставка 244
6.2.4. Удаление 247
6.2.5. Самый длинный префикс 250
6.2.6. Ключи, соответствующие префиксу 251
6.2.7. Когда следует использовать префиксные деревья 254
6.3. Компактные префиксные деревья 256
6.3.1. Узлы и ребра 258
6.3.2. Поиск 262
6.3.3. Вставка 264
6.3.4. Удаление 266
6.3.5. Самый длинный общий префикс 268
6.3.6. Поиск ключей, начинающихся с префикса 269
6.4. Варианты применения 270
6.4.1. Проверка орфографии 271
6.4.2. Сходство строк 273
6.4.3. Сортировка строк 274
6.4.4. T9 275
6.4.5. Автодополнение 276
Резюме 277

Глава 7. Примеры использования: кэш LRU 279
7.1. Не вычисляйте одно и то же дважды 280
7.2. Первая попытка: запоминание значений 284
7.2.1. Описание и API 286
7.2.2. Освежите данные, пожалуйста 286
7.2.3. Обработка асинхронных вызовов 288
7.2.4. Маркировка значений в кэше признаком «загружается» 289
7.3. Нехватка памяти (буквально) 290
7.4. Удаление устаревших данных: кэш LRU 292
7.4.1. Иногда важно удвоить усилия для решения проблем 294
7.4.2. Упорядочение по времени 295
7.4.3. Производительность 302
7.5. Когда более свежие данные ценнее: LFU 302
7.5.1. Как сделать правильный выбор 304
7.5.2. Отличительные особенности LFU 304

7.5.3. Производительность . . . 307
7.5.4. Недостатки политики LFU . . . 308
7.6. Порядок использования кэша не менее важен. . . . 309
7.7. Введение в синхронизацию . . . 310
7.7.1. Решение проблемы параллельного выполнения (на Java) . . . 313
7.7.2. Блокировки . . . 314
7.7.3. Получение блокировки . . . 316
7.7.4. Реентерабельные блокировки . . . 317
7.7.5. Блокировки для чтения. . . . 318
7.7.6. Другие подходы к решению проблемы параллелизма . . . 319
7.8. Примеры применения кэша . . . 320
Резюме . . . 322

Часть II
Многомерные запросы

Глава 8. Поиск ближайших соседей . . . 326
8.1. Задача поиска ближайшего соседа . . . 327
8.2. Решения . . . 328
8.2.1. Первые попытки . . . 329
8.2.2. Иногда кэширование не является решением . . . 329
8.2.3. Упрощение, дающее подсказку . . . 330
8.2.4. Тщательно выбирайте структуру данных . . . 332
8.3. Описание и API. . . . 334
8.4. Переход к k-мерным пространствам . . . 335
8.4.1. Одномерный двоичный поиск . . . 336
8.4.2. Переход к более высоким измерениям . . . 336
8.4.3. Моделирование двумерных разделов с помощью структуры данных . 338
Резюме . . . 340

Глава 9. k-мерные деревья: индексирование многомерных данных . . . 341
9.1. Продолжаем с того места, на котором остановились. . . . 341
9.2. Переход к k-мерным пространствам: циклический перебор измерений. . . . 343
9.2.1. Конструирование двоичного дерева поиска . . . 344
9.2.2. Инварианты . . . 350
9.2.3. Важность сбалансированности . . . 351
9.3. Методы . . . 351
9.3.1. Метод search. . . . 353
9.3.2. Метод insert . . . 356

9.3.3. Сбалансированное дерево 359
9.3.4. Метод remove 362
9.3.5. Ближайший сосед 371
9.3.6. Поиск области 381
9.3.7. Обзор всех методов 386
9.4. Ограничения и возможные улучшения 387
Резюме 389

Глава 10. Деревья поиска по сходству: приближенный поиск ближайших соседей для выбора похожих изображений 390
10.1. Продолжаем с того места, на котором остановились 391
10.1.1. Новый более сложный пример 392
10.1.2. Преодоление недостатков k-мерных деревьев 393
10.2. R-дерево. 394
10.2.1. Шаг назад: знакомство с B-деревьями 394
10.2.2. От B-дерева к R-дереву 395
10.2.3. Вставка точек в R-дерево 398
10.2.4. Поиск 400
10.3. Деревья поиска по сходству 402
10.3.1. Поиск в SS-дереве 406
10.3.2. Вставка 411
10.3.3. Вставка: дисперсия, средние значения и проекции 418
10.3.4. Вставка: разделение узлов 422
10.3.5. Удаление 426
10.4. Поиск по сходству 433
10.4.1. Поиск ближайшего соседа 434
10.4.2. Поиск области 438
10.4.3. Приближенный поиск по сходству 439
10.5. Деревья SS$^+$. 443
10.5.1. SS-деревья лучше? 443
10.5.2. Смягчение недостатков гиперсфер 445
10.5.3. Улучшение эвристики разделения 446
10.5.4. Уменьшение площади перекрытия 447
Резюме 450

Глава 11. Применение поиска ближайшего соседа на практике 451
11.1. Приложение: поиск ближайшего склада 452
11.1.1. Набросок решения 453
11.1.2. Неожиданные проблемы 455

11.2. Централизованное приложение 457
11.2.1. Фильтрация точек 457
11.2.2. Сложные решения 459
11.3. Переход к распределенному приложению 462
11.3.1. Проблемы, связанные со взаимодействиями через HTTP 464
11.3.2. Синхронизация информации о товарах на складах. 466
11.3.3. Извлеченные уроки 468
11.4. Другие приложения 468
11.4.1. Уменьшение количества цветов. 468
11.4.2. Взаимодействие частиц 471
11.4.3. Оптимизация многомерных запросов к БД 473
11.4.4. Кластеризация 476
Резюме 477

Глава 12. Кластеризация. 478
12.1. Введение в кластеризацию 480
12.1.1. Типы обучения 480
12.1.2. Типы кластеризации. 482
12.2. k-средних 484
12.2.1. Недостатки алгоритма k-средних. 489
12.2.2. Проклятие размерности снова в действии 492
12.2.3. Анализ производительности метода k-средних 493
12.2.4. Ускорение метода k-средних с помощью k-мерных деревьев 494
12.2.5. Заключительные замечания об алгоритме кластеризации методом k-средних 498
12.3. DBSCAN 499
12.3.1. Прямая достижимость и достижимость по плотности. 500
12.3.2. От определений к алгоритму 501
12.3.3. И наконец, реализация 504
12.3.4. Достоинства и недостатки DBSCAN. 505
12.4. OPTICS 508
12.4.1. Определения. 510
12.4.2. Алгоритм OPTICS 510
12.4.3. От расстояния достижимости к кластеризации 516
12.4.4. Иерархическая кластеризация 519
12.4.5. Анализ производительности и заключительные замечания 525
12.5. Оценка результатов кластеризации: метрики оценки 526
12.5.1. Интерпретация результатов 530
Резюме 531

Глава 13. Параллельная кластеризация: MapReduce и кластеризация методом купола 532
13.1. Параллельные вычисления 533
13.1.1. Параллельные и распределенные вычисления 534
13.1.2. Распараллеливание метода k-средних 535
13.1.3. Кластеризация методом купола 536
13.1.4. Применение кластеризации методом купола 539
13.2. MapReduce 540
13.2.1. Представьте, что вы Дональд Дак.... 541
13.2.2. Сначала отображение, потом свертка 545
13.2.3. Еще кое-что 549
13.3. Реализация метода k-средних с помощью модели MapReduce 550
13.3.1. Распараллеливание кластеризации методом купола 554
13.3.2. Инициализация центроидов с помощью кластеризации методом купола 556
13.3.3. Кластеризация методом купола с использованием модели MapReduce 559
13.4. Реализация DBSCAN с использованием MapReduce 563
Резюме 572

Часть III
Планарные графы и минимальное число пересечений

Глава 14. Введение в графы: поиск кратчайшего пути 575
14.1. Определения 576
14.1.1. Реализация графов 577
14.1.2. Графы как алгебраические типы 579
14.1.3. Псевдокод 580
14.2. Свойства графа 582
14.2.1. Неориентированный 582
14.2.2. Связный 583
14.2.3. Ациклический 584
14.3. Обход графа: алгоритмы BFS и DFS 586
14.3.1. Оптимизация маршрутов доставки 586
14.3.2. Поиск в ширину 588
14.3.3. Реконструкция пути к цели 592
14.3.4. Поиск в глубину 593
14.3.5. И снова о выборе между очередью и стеком 597
14.3.6. Лучший маршрут доставки посылки 598

14.4. Кратчайший путь во взвешенных графах: алгоритм Дейкстры 599
14.4.1. Отличия от BFS. 600
14.4.2. Реализация. 601
14.4.3. Анализ. 602
14.4.4. Кратчайший путь доставки. 604
14.5. За пределами алгоритма Дейкстры: A* 606
14.5.1. Насколько хорош алгоритм A*?. 610
14.5.2. Эвристика как способ балансировки в реальном времени 614
Резюме. 616

Глава 15. Представление графов и планарность: рисование графа с минимальным числом пересечений ребер. 617
15.1. Представление графов. 618
15.1.1. Некоторые основные определения 620
15.1.2. Полные и двудольные графы 622
15.2. Планарные графы. 623
15.2.1. Практическое применение теоремы Куратовского 624
15.2.2. Проверка планарности 626
15.2.3. Наивный алгоритм проверки планарности. 628
15.2.4. Увеличение производительности. 632
15.2.5. Эффективные алгоритмы. 635
15.3. Непланарные графы 637
15.3.1. Поиск числа пересечений. 639
15.3.2. Количество прямолинейных пересечений 641
15.4. Пересечения ребер 643
15.4.1. Прямолинейные отрезки 643
15.4.2. Ломаные линии 648
15.4.3. Кривые Безье 648
15.4.4. Пересечение квадратичных кривых Безье 651
15.4.5. Пересечения «вершина — вершина» и «ребро — вершина». 654
Резюме. 656

Глава 16. Градиентный спуск: оптимизация задач (не только) на графах 657
16.1. Эвристика для определения числа пересечений. 659
16.1.1. Мне послышалось, вы только что сказали «эвристики»? 659
16.1.2. Расширение до поддержки криволинейных ребер 666
16.2. Как работает оптимизация 668
16.2.1. Функции стоимости 669
16.2.2. Ступенчатые функции и локальные минимумы 671
16.2.3. Оптимизация случайной выборки 672

16.3. Градиентный спуск 675
16.3.1. Математика градиентного спуска 677
16.3.2. Геометрическая интерпретация 678
16.3.3. Когда применяется градиентный спуск 680
16.3.4. Задачи с градиентным спуском 681
16.4. Применение градиентного спуска 683
16.4.1. Пример: линейная регрессия 685
16.5. Градиентный спуск для поиска представления графа 688
16.5.1. Другие критерии 689
16.5.2. Реализация 691
Резюме 693

Глава 17. Имитация отжига: оптимизация за пределами локальных минимумов . . 694
17.1. Имитация отжига 696
17.1.1. Иногда нужно подняться, чтобы спуститься 698
17.1.2. Реализация 700
17.1.3. Почему имитация отжига работает 701
17.1.4. Переходы на короткие и большие расстояния 705
17.1.5. Варианты 706
17.1.6. Имитация отжига или градиентный спуск: что использовать? 708
17.2. Имитация отжига + коммивояжер 709
17.2.1. Точное и приближенное решения 711
17.2.2. Визуализация стоимости 712
17.2.3. Сужение пространства задачи 714
17.2.4. Переходы между состояниями 715
17.2.5. Перестановка смежных и случайно выбранных вершин 719
17.2.6. Применение решения задачи о коммивояжере 721
17.3. Имитация отжига и представление графа 721
17.3.1. Минимальное число пересечений ребер 721
17.3.2. Силовые алгоритмы визуализации 724
Резюме 730

Глава 18. Генетические алгоритмы: заимствованная из биологии быстросходящаяся оптимизация 732
18.1. Генетические алгоритмы 733
18.1.1. Заимствование у природы 735
18.1.2. Хромосомы 739
18.1.3. Популяция 742
18.1.4. Приспособленность 743
18.1.5. Естественный отбор 744

18.1.6. Отбор индивидуумов для спаривания . . . 748
18.1.7. Скрещивание . . . 755
18.1.8. Мутации . . . 758
18.1.9. Шаблон генетического алгоритма . . . 761
18.1.10. Когда генетический алгоритм работает лучше всего . . . 761
18.2. Задача о коммивояжере . . . 763
18.2.1. Приспособленность, хромосомы и инициализация . . . 764
18.2.2. Мутации . . . 764
18.2.3. Скрещивание . . . 765
18.2.4. Настройка результатов и параметров . . . 767
18.2.5. За пределами задачи о коммивояжере: оптимизация маршрутов для всего парка грузовиков . . . 771
18.3. Минимальное вершинное покрытие . . . 772
18.3.1. Практическое применение вершинного покрытия . . . 774
18.3.2. Реализация генетического алгоритма . . . 775
18.4. Другие варианты применения генетического алгоритма . . . 777
18.4.1. Максимальный поток . . . 777
18.4.2. Свертывание белков . . . 780
18.4.3. За пределами генетических алгоритмов. . . . 781
18.4.4. За пределами круга алгоритмов, рассмотренных в этой книге. . . . 782
Резюме . . . 783

Приложения

Приложение А. Краткое руководство по псевдокоду . . . 786
A.1. Переменные и основы синтаксиса . . . 787
A.2. Массивы . . . 788
A.3. Условные инструкции . . . 789
A.3.1. else-if . . . 790
A.3.2. switch . . . 791
A.4. Циклы . . . 792
A.4.1. Цикл for . . . 792
A.4.2. Цикл while. . . . 793
A.4.3. break и continue . . . 794
A.5. Блоки и отступы . . . 794
A.6. Функции . . . 795
A.6.1. Перегрузка и аргументы по умолчанию . . . 796
A.6.2. Кортежи . . . 796
A.6.3. Кортежи и деструктурирование объектов . . . 797

Приложение Б. Нотация «О большое» 799
Б.1. Алгоритмы и производительность 799
Б.2. Модель памяти 800
Б.3. Порядок величины 801
Б.4. Нотация 802
Б.5. Примеры 805

Приложение В. Основные структуры данных 808
В.1. Основные структуры данных 808
В.2. Массив 809
В.3. Связанный список 811
В.4. Дерево 814
В.4.1. Двоичные деревья поиска 816
В.5. Хеш-таблица 818
В.5.1. Хранение пар «ключ — значение» 818
В.5.2. Хеширование 819
В.5.3. Разрешение коллизий в хешировании 820
В.5.4. Производительность 822
В.6. Сравнительный анализ основных структур данных 822

Приложение Г. Контейнеры в роли очередей с приоритетами 825
Г.1. Мультимножество 826
Г.2. Стек 827
Г.3. Очередь 828
Г.4. Сравнительный анализ контейнеров 828

Приложение Д. Рекурсия 830
Д.1. Простая рекурсия 831
Д.1.1. Ловушки 831
Д.1.2. Оправданная рекурсия 833
Д.2. Хвостовая рекурсия 834
Д.3. Взаимная рекурсия 837

Приложение Е. Задачи классификации и рандомизированные алгоритмы 839
Е.1. Задачи принятия решений 840
Е.2. Алгоритмы Лас-Вегаса 840
Е.3. Алгоритмы Монте-Карло 841
Е.4. Метрики классификации 842
Е.4.1. Достоверность 842
Е.4.2. Точность и полнота 843
Е.4.3. Другие метрики и подведение итогов 845