1 Знакомство. Почему стоит изучать структуры данных

В этой главе

- ✓ Знакомство с книгой
- ✓ Что такое структуры данных
- ✓ Для чего нужны структуры данных
- ✓ Примеры полезных структур данных
- ✓ Пошаговое руководство по применению в проекте структур данных

Структуры данных правят миром: информация — валюта интернет-эпохи, а структуры данных необходимы для обработки и осмысления информации. Структуры данных позволяют осознанно формировать данные и запрашивать именно то, что нам требуется.

Знакомство с книгой

Добро пожаловать в «Грокаем структуры данных»! Я с огромным удовольствием буду сопровождать вас в путешествии по миру структур данных.

В этой книге я хочу развеять некоторые заблуждения относительно структур данных: на самом деле они в высшей степени полезны в вашей повседневной работе. Даже если вы не ученый, от них многое зависит, а из-

учить их не так уж сложно: чтобы понять структуры данных и использовать их, совершенно необязательно быть знатоком математики!

Во время нашего путешествия я покажу, что структуры данных — вовсе не сухая и скучная теория. Они настолько глубоко вошли в нашу жизнь, что вы наверняка регулярно пользуетесь ими, даже не подозревая об этом. Помимо программирования, вы использовали или видели в действии некоторые из описанных структур, которые можно найти в обычных жизненных ситуациях.

Структуры данных вокруг нас

Не верите? Можно было бы добавить интриги, поспорив на что-нибудь, но это было бы нечестно с моей стороны!

Давайте разберемся. Вы же бывали в продуктовом или универмаге? Вот вы наполняете тележку товарами — пожалуйста, это контейнер. Но какой именно контейнер? Не хочу спойлерить — вы сможете ответить, прочитав книгу.

Итак, набрав все, за чем пришли, вы отправляетесь к кассиру — заплатить. А пока расплачиваются предыдущие покупатели, вы фактически находитесь в другой структуре данных — в очереди!

Ну как, мне удалось вас убедить? Если вы разработчик программного обеспечения, то убедить вас будет еще проще, потому что, если вы пишете код, то как минимум пользовались массивами. Не говоря уже о том, что если вы читаете цифровую версию этой книги, то, конечно, ваш ридер использует множество структур данных для хранения страниц книги, ее слов, добавленных вами закладок и т. д.

Структуры данных для всех

С помощью этой книги структуры данных сможет освоить кто угодно, независимо от подготовки. Для этого не понадобится высшая математика, не понадобится вводный курс программирования да и любой другой учебный курс — необязательно быть ниндзя кодинга, хотя и могут пригодиться неко-

торые знания Python. Эта книга, как и любая другая книга серии «Грокаем», поможет вам понять, что к чему: она объяснит, что такое структура данных, познакомит вас с основными структурами и подскажет, как объективно решить, какая структура данных лучше подойдет для вашей задачи. Эта книга для новичков, то есть не предполагает предварительных знаний и полагается на вашу интуицию и зрительную память. Впрочем, даже если вы уже знакомы с предметом, то, возможно, сочтете полезным для себя отточить свои навыки и еще глубже понять какие-то вопросы.

Что такое структуры данных?

Если вы читаете эту книгу, то, вероятно, уже знаете, что мы живем в так называемую эпоху данных — эру, в которой данные стали неотъемлемой частью нашего существования. Нашу жизнь наводнила информация, производство которой растет с беспрецедентной, экспоненциальной скоростью, подпитываемой технологическими достижениями. Этот поток данных меняет наш образ жизни, работу и взаимоотношения.

Чтобы не утонуть и сориентироваться в этом огромном объеме информации, надо ее как-то упорядочить. Вот тут-то и выручают структуры данных.

Структуры данных — это способ организации и хранения информации на компьютере или в программе. Они помогают эффективно управлять и оперировать данными.

Допустим, вы хотите узнать, есть ли ваш школьный друг на Facebook. Вы сможете это сделать — только потому, что есть структура данных, которая упорядочивает информацию таким образом, что поиск среди миллиарда пользователей становится простым и быстрым.

Алгоритмы и структуры данных

О *структурах данных* часто упоминают в связи с *алгоритмами*. Собственно, настолько часто, что вы можете спросить: а не одно ли это и то же. Нет, алгоритмы и структуры данных — разные вещи, хотя между ними существует тесная связь.

Алгоритм — это набор четко прописанных команд, пошаговая процедура, предназначенная для решения конкретной проблемы или выполнения определенной задачи. В нашем примере с Facebook используется алгоритм, который перебирает имена всех пользователей и предлагает совпадения с запросом или наиболее близкие результаты.

Структура данных — это способ организации и хранения данных на компьютере или в языке программирования. Она определяет, как элементы данных соотносятся друг с другом, какие операции с ними можно выполнять, а также правила и ограничения относительно доступа к данным и возможности их изменять. Данные пользователей Facebook хранятся в базе, организованной так, чтобы обеспечить эффективный поиск по именам.

ПРИМЕЧАНИЕ Понятие алгоритма используется для описания операций, выполняемых со структурами данных. Используя определенную аналогию, структуры данных подобны существительным, тогда как алгоритмы — скорее глаголы.

Структуры данных и алгоритмы взаимосвязаны — так же, как осмысленному предложению в английском языке для описания действия нужны субъект, объект и глагол.

Алгоритмы, преобразующие данные, — как глаголы, оперирующие с существительными[1]

Структуры данных обеспечивают организацию и представление информации (данных), а алгоритмы служат командами для преобразования этих данных. Каждая структура данных неявно определяет алгоритмы для различных операций, включая добавление, выборку и удаление элементов.

Некоторые структуры данных специально разработаны с прицелом на эффективное выполнение определенных алгоритмов — например, хеш-таблицы для поиска по ключу (не беспокойтесь, если вы пока не знаете этих терминов: мы всё это чуть позже разберем).

Таким образом, чтобы описать структуру данных, необходимо точно объяснить алгоритмы, лежащие в основе ее методов. Другими словами, в этой книге вы узнаете о множестве алгоритмов.

Какое мне дело до структур данных?

Структуры данных — это строительные блоки computer science. Они важны, потому что помогают упорядочивать данные, решать сложные задачи, повышать эффективность, оптимизировать использование памяти и избегать

[1] В английском языке названия алгоритмов звучат как глаголы (Quicksort — букв. «быстро сортируй»). — *Примеч. ред.*

рисков с точки зрения безопасности. В общем, это важнейшие инструменты эффективного управления информацией и ее обработки в компьютерных программах.

Последнее время в computer science появились новые тенденции, которые используют преимущества структур данных — например, графовые нейронные сети: еще более мощная версия строительных блоков, предназначенных для машинного обучения, в моделях глубокого обучения.

Сфера баз данных тоже развивается, и недавно возникла концепция *гибкого индексирования* (flexible indexing). Такая модель индексирования основана на структурах данных, которые могут быть вложены (nested) в любом сочетании и на любой глубине. Это чрезвычайно мощный инструмент, и использовать его возможности получится, только если вы хорошо владеете структурами данных.

Впрочем, я приведу еще более убедительный довод: знание структур данных может повысить вашу квалификацию как разработчика программного обеспечения. Искушенность в области структур данных — это как дополнительный инструмент в вашем арсенале.

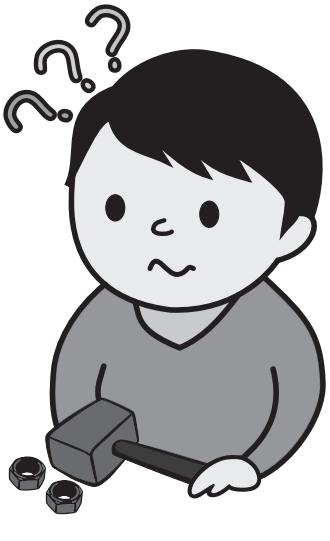
Вы когда-нибудь слышали о «молотке Маслоу», также известном под названиями «золотой молоток» и «закон инструмента»? Этот закон гласит, что если в вашем арсенале только молоток, то у вас будет соблазн рассматривать любую задачу как забивание гвоздей.

По сути, это означает, что люди склонны применять привычные решения в самых разных ситуациях — даже там, где это неуместно.

Как это связано со структурами данных? Если вы умеете пользоваться только одной структурой данных, например хеш-таблицей, у вас появится искушение применять ее повсюду, даже если вам нужно эффективно выполнять такие операции, как переход к *следующему* и *предыдущему* элементу, для чего было бы лучше использовать дерево.

Не огорчайтесь, если этот пример показался туманным и неочевидным или вам незнакомы какие-то термины, — тем больше причин читать дальше, потому что мы эту тему позже разберем.

Книга предоставляет в ваше распоряжение дополнительные инструменты для решения подобных задач и учит распознавать возможности применять эти инструменты для улучшения кода.



Если нужно затянуть болт, от молотка пользы не будет

В каких случаях нужны структуры данных?

В теории структуры данных нужны тогда, когда надо организовать данные так, чтобы их можно было легко и эффективно хранить (store) и извлекать (retrieve) в соответствии с определенными заданными правилами. Это очень формальное определение, и при всей своей формальной корректности оно кажется несколько далеким от наших повседневных дел, от мира, каким мы его знаем.

Рассмотрим несколько примеров структур данных в действии, чтобы лучше понять, о чем идет речь.

Поиск как у профи

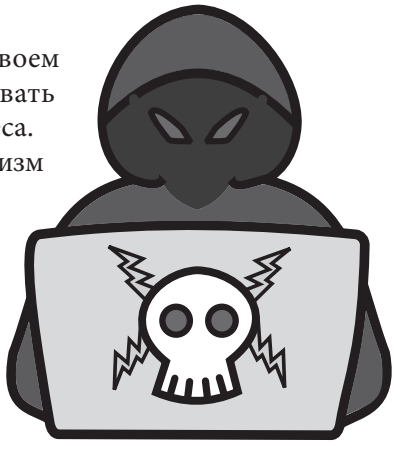
У Тома большая коллекция — представьте себе тысячи бейсбольных карточек или миллионы товаров на сайте его интернет-магазина. У этих позиций есть атрибуты, часть которых (например, имена) уникальны и однозначно их идентифицируют.

Как Тому организовать поиск в своей коллекции? Например, как ему найти карточку Джо Ди Маджио среди всех своих бейсбольных карточек?

Конечно, можно перебирать их одну за другой, пока не найдется нужная. Если вы такой же страстный коллекционер, как и я, то знаете, что поиск в коллекции из тысяч предметов может занять много времени. А представьте, сколько времени уйдет на поиск товара в онлайн-каталоге с миллионами позиций!

Для хранения и поиска Тому нужен способ получше, а заодно неплохо бы ему узнать о компромиссах, необходимых при сбалансированном удовлетворении разноплановых потребностей. Книга предлагает разные варианты структур данных, обеспечивающих эффективный поиск, и может помочь найти ту, что лучше всего подходит для решения конкретной задачи. Как насчет того, чтобы начать с *отсортированных массивов* (sorted arrays) и *двоичного поиска* (binary search)?

Так много пользователей!



Давайте рассмотрим другой сценарий. В своем веб-приложении Кэт необходимо отслеживать залогинившихся пользователей и их IP-адреса.

Поначалу она сама реализовывает механизм отслеживания IP-адресов. Локально все работает прекрасно. Но когда она запускает свои изменения в рабочую версию, структура данных оказывается слишком медленной для реального трафика веб-приложения, что приводит к сбою сервера.

Учитывая срочность вопроса, Кэт передает поиск решения фирме-консультанту в надежде, что там знают лучше. Их решение и правда обеспечивает скорость. Оно даже в рабочей версии хорошо работает… пока не перестает. Оказывается, хакер вычислил, что определенной последовательностью вызовов можно переполнить структуру данных, предложенную внешней компанией, и запросто обрушить приложение Кэт. Что же произошло? В первый раз проблема была в производительности, потому что была выбрана неподходящая структура данных — она оказалась слишком медленной для работы в масштабе.

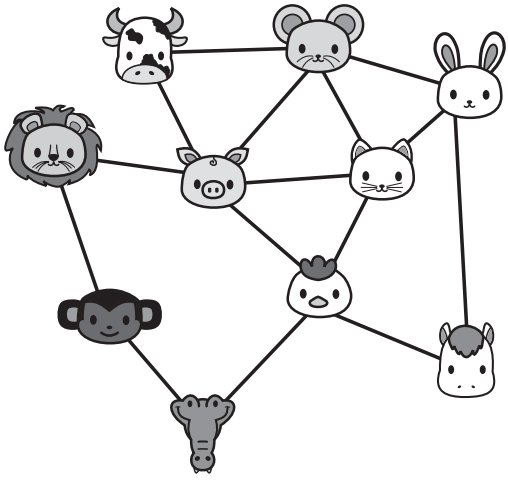
Во второй раз был выбран вариант получше. К сожалению, новое решение использовалось слишком неосторожно и оставило уязвимость для вредоносной последовательности (adversary sequence) (последовательность входов, выбранная для данного конкретного случая с целью создать проблемы со структурой данных). В свою очередь, эта уязвимость открыла возможность атаки *отказа в обслуживании*, или *DoS-атаки* (Denial of Service). В таком сценарии хакер может воспользоваться уязвимостью и замедлить работу приложения настолько, что с ним не смогут взаимодействовать легитимные пользователи. Как же быть? Мы увидим, что *хеш-таблицы*, если их правильно использовать, должны решить большинство проблем Кэт. Когда речь пойдет о хеш-таблицах, мы больше узнаем о проблеме, которая сделала возможной DoS-атаку, о том, как ее устранить, и, что еще важнее, на что следует обратить особое внимание. Даже покупая готовый продукт у сторонней компании, необходимо знать, какие вопросы следует задать, чтобы убедиться, что все сделано правильно.

Моделирование взаимосвязей

Сандра запускает социальную сеть нового поколения, которая навсегда изменит наше общение. По крайней мере, это ее мечта. Она все еще строит минимально жизнеспособный продукт и надеется найти финансирование.

Она добилась значительного прогресса, но слегка споткнулась на отслеживании взаимосвязей между пользователями. Сандра хочет опробовать что-то вроде электронной таблицы (spreadsheet) или табличной структуры (tabular structure), но не уверена в том, как лучше ее хранить и реализовать запросы об отношениях типа «друзья друзей».

Для начала Сандра пытается использовать наивное решение, несколько раз перебрав весь список пользователей, но приложение почему-то перестает реагировать, что ее сильно огорчает.



Сандра пыталась проделать это только в памяти — но что, если понадобится добавить возможность долгосрочного хранения этих данных? А если дальше понадобится находить еще более отдаленные взаимосвязи — типа «друг друга друзей» или «шесть шагов до Кевина Бейкона»[1]? К сожалению, база данных SQL, похоже, не поддерживает всего того, что ей нужно.

Дальше вы узнаете, что справиться с тесно связанными данными Сандре помогут графы, а чтобы обнаруживать опосредованные дружеские связи, можно воспользоваться алгоритмом поиска в ширину (breadth-first search). С графами подтягиваются и графовые базы данных (graph databases) — другой механизм хранения тесно связанных данных, позволяющий быстро выполнять запросы исходя из характера взаимосвязей между различными компонентами данных.

[1] https://ru.wikipedia.org/wiki/Шесть_шагов_до_Кевина_Бейкона.

Мне когда-нибудь придется писать код для всех этих структур данных?

Если не считать должностей, связанных с научными исследованиями, на большинстве позиций инженерам-разработчикам программного обеспечения обычно не приходится писать свои собственные алгоритмы и структуры данных каждый день или неделю. По большей части можно просто брать чужой код. Впрочем, даже в этом случае изучение структур данных поможет делать правильный выбор или дать представление о том, что есть более подходящие варианты.

В некоторых ситуациях придется закатать рукава и написать собственную реализацию — например, если вы используете совершенно новый язык программирования, для которого еще не так много доступных библиотек, или же структуру данных необходимо настроить для решения какой-то специфической задачи.

Но даже если вам никогда не придется писать собственные реализации, все равно, только не понаслышке зная структуры данных, вы сможете лучше понять, на какие компромиссы идете в своем коде и как сделать его более эффективным.

Как выбрать структуру данных?

Из примеров предыдущего раздела очевидно, насколько важно с умом выбирать структуры данных. Но есть и другой, менее очевидный, момент: дело вовсе не в том, чтобы выбрать идеальную структуру. Более того, вам даже не всегда надо брать лучшую из возможных; в большинстве случаев сойдет и просто более-менее близкий к оптимальному вариант. Но что принципиально важно, — так это избежать ошибочного выбора: в пользу структуры данных, которая обрушит ваше приложение или создаст проблемы с точки зрения его безопасности.

Самое важное, что (как я надеюсь) вы вынесете из этой книги, — это метод оценки и выбора структуры данных в той или иной ситуации.

Как это делается? Искусство выбирать правильную структуру данных — это своего рода мускулатура, которую необходимо тренировать. По ходу изложения мы наделим вас знаниями и разовьем вашу интуицию, показывая опасности, с которыми можно столкнуться, как их системно выявлять через оценку сложности алгоритмов, какие аспекты надо сбалансировать и какие компромиссы учесть.