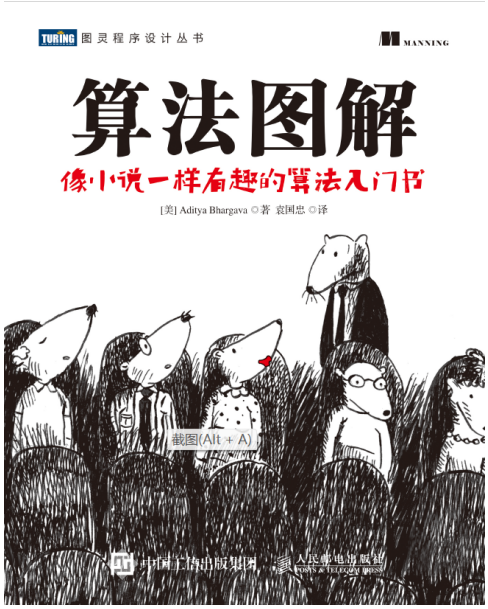
Grokking Algorithms算法图解

这是我（Samwee黄国洪）在阅读《算法图解》这本书时顺手写的读书笔记，写得比较乱，估计也只有我自己回头复习的时候看得懂。但这本书确实简单有趣，而且篇幅很短，想看的不妨去翻翻原书，很推荐。

0 封面



1 算法简介

1.2 二分查找最多只需要log2 n步，好理解吧

1.3 大O表示法

也很好理解

1.3.4 一些常见的大 O 运行时间

下面按从快到慢的顺序列出了你经常会遇到的5种大O运行时间。

- $O(\log n)$，也叫对数时间，这样的算法包括二分查找。
- $O(n)$，也叫线性时间，这样的算法包括简单查找。
- $O(n * \log n)$，这样的算法包括第4章将介绍的快速排序——一种速度较快的排序算法。
- $O(n^2)$，这样的算法包括第2章将介绍的选择排序——一种速度较慢的排序算法。
- $O(n!)$，这样的算法包括接下来将介绍的旅行商问题的解决方案——一种非常慢的算法。

2 选择排序

2.1 内存的工作原理

2.2 数组和链表

数组的话，就像你与朋友去看电影，找到地方就坐后又来了一位朋友，但原来坐的地方没有空位置，只得再找一个可坐下所有人的地方。

但链表，链表中的元素可存储在内存的任何地方。像寻宝游戏一样

也就是说链表可以分开坐，好理解吧

那数组的优势又是什么呢？

2.2.2 数组

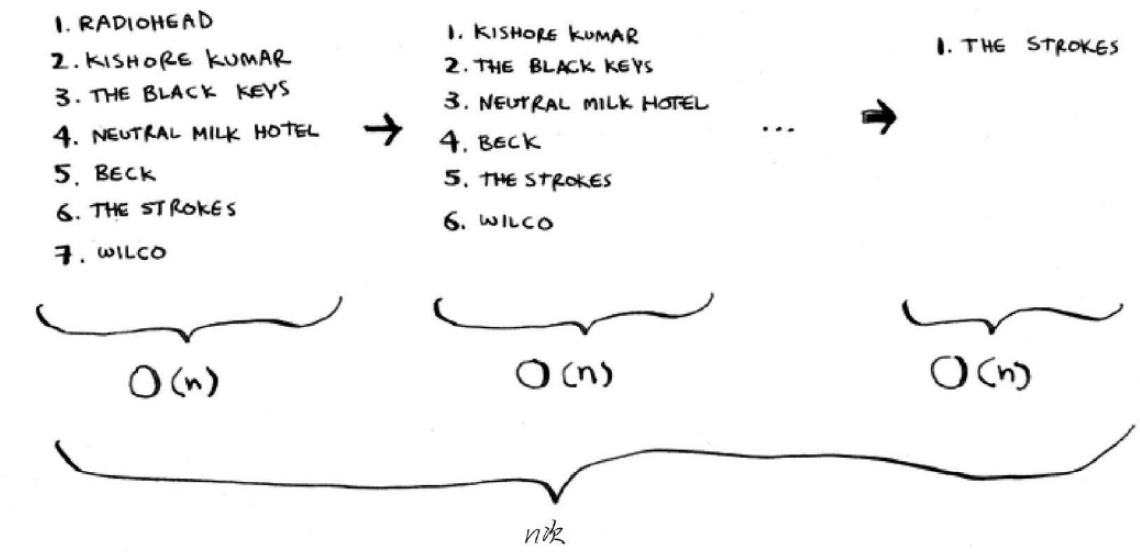
排行榜网站使用卑鄙的手段来增加页面浏览量。它们不在一个页面中显示整个排行榜，而将排行榜的每项内容都放在一个页面中，并让你单击Next来查看下一项内容。例如，显示十大电视反派时，不在一个页面中显示整个排行榜，而是先显示第十大反派（Newman）。你必须在每个页面中单击Next，才能看到第一大反派（Gustavo Fring）。这让网站能够在10个页面中显示广告，但用户需要单击Next九次才能看到第一个，真的是很烦。

如果整个排行榜都显示在一个页面中，将方便得多。这样，用户可单击排行榜中的人名来获得更详细的信息。

就是需要随机地读取元素时，数组的效率很高，因为可迅速找到数组的任何元素。

数组和链表哪个用得更多呢？显然要看情况。但数组用得很多，因为它支持随机访问。有两种访问方式：随机访问和顺序访问。顺序访问意味着从第一个元素开始逐个地读取元素。链表只能顺序访问：要读取链表的第十个元素，得先读取前九个元素，并沿链接找到第十个元素。随机访问意味着可直接跳到第十个元素。本书经常说数组的读取速度更快，这是因为它们支持随机访问。很多情况都要求能够随机访问，因此数组用得很多。数组和链表还被用来实现其他数据结构，这将在本书后面介绍。

2.3 选择排序



需要的总时间为 $O(n \times n)$，即$O(n^2)$。

我觉得选择排序就是傻傻慢慢的排序方法，但也很符合人的直觉。

3 递归

3.1 递归

用人话说就是，在需要实现循环的时候，让函数调用它自己。那这样有什么好处呢？

书中原话：递归只是让解决方案更清晰，并没有性能上的优势。

也就是说，更容易理解，这样子而已。

3.3 栈

只有一个口，上入上出

4 快速排序

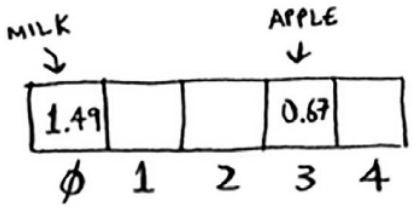
4.1 divide and conquer,D&C

需要理解基线条件

4.2 快速排序

最佳情况也是平均情况。只要你每次都随机地选择一个数组元素作为基准值，快速排序的平均运行时间就将为O(n log n)。快速排序是最快的排序算法之一，也是D&C典范。

5 散列表 hash table



5.2.4 小结

这里总结一下，散列表适合用于：

- 模拟映射关系；
- 防止重复；
- 缓存/记住数据，以免服务器再通过处理来生成它们。

5.4 性能

较小的装填因子

一旦填装因子超过0.7，就该调整散列表的长度。

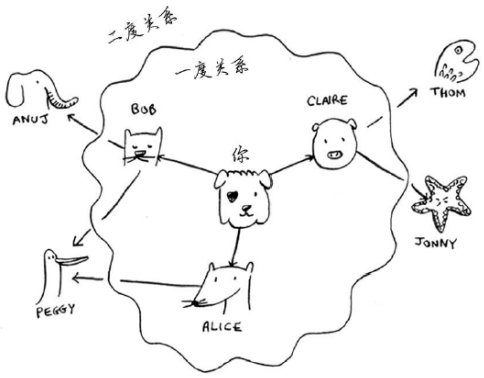
良好的散列函数

6 广度优先搜索 breadth-first search，BFS

6.1 6.2 图

6.3 广度优先搜索

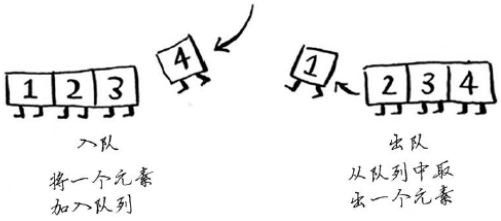
1 查找最短路径



在你看来，一度关系胜过二度关系，二度关系胜过三度关系，以此类推。因此，你应先在一度关系中搜索，确定其中没有芒果销售商后，才在二度关系中搜索。广度优先搜索就是这样做的！

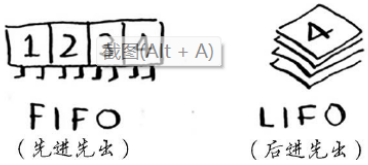
广度优先搜索用来找最短路径

2 队列 queue



如果你将两个元素加入队列，先加入的元素将在后加入的元素之前出队。因此，你可使用队列来表示查找名单！这样，先加入的人将先出队并先被检查。

队列是一种先进先出（First In First Out，FIFO）的数据结构，而栈是一种后进先出（Last In First Out，LIFO）的数据结构。

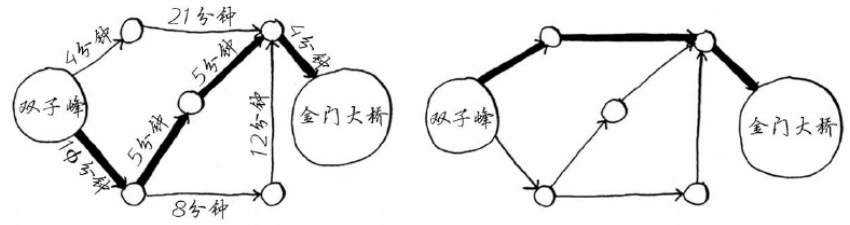


6.4 实现图

6.5 实现算法

7 狄克斯特拉算法

在前一章，你使用了广度优先搜索来查找两点之间的最短路径，那时“最短路径”的意思是段数最少。在狄克斯特拉算法中，你给每段都分配了一个数字或权重，因此狄克斯特拉算法找出的是总权重最小的路径。



加权图
（使用狄克斯特拉算法）

非加权图
（使用广度优先搜索）

这里重述一下，狄克斯特拉算法包含4个步骤。

(1) 找出最便宜的节点，即可在最短时间内前往的节点。

(2) 对于该节点的邻居，检查是否有前往它们的更短路径，如果有，就更新其开销。

(3) 重复这个过程，直到对图中的每个节点都这样做了。

(4) 计算最终路径。（下一节再介绍！）

因此，不能将狄克斯特拉算法用于包含负权边的图。在包含负权边的图中，要找出最短路径，可使用另一种算法——贝尔曼-福德算法（Bellman-Fordalgorithm）。本书不介绍这种算法

8 贪婪算法

8.1 教室调度问题

8.2 背包问题

讲小偷怎么才能往背包装最多东西的

8.3 集合覆盖问题

讲广播电台如何覆盖50个州的

8.4 NP完全问题

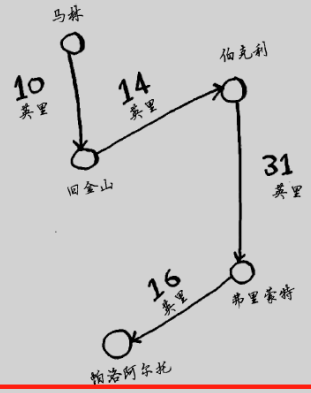
8.4.1 旅行商问题详解

旅行商问题和集合覆盖问题有一些共同之处：你需要计算所有的解，并从中选出最小/最短的那个。这两个问题都属于NP完全问题。

近似求解

对旅行商问题来说，什么样的近似算法不错呢？能找到较短路径的算法就算不错。在继续往下阅读前，看看你能设计出这样的算法吗？

我会采取这样的做法：随便选择出发城市，然后每次选择要去的下一个城市时，都选择还没去的最近的城市。假设旅行商从马林出发。



总旅程为71英里。这条路径可能不是最短的，但也相当短了。

8.4.2 如何识别NP 完全问题

- 元素较少时算法的运行速度非常快，但随着元素数量的增加，速度会变得非常慢。
- 涉及“所有组合”的问题通常是NP完全问题。
- 不能将问题分成小问题，必须考虑各种可能的情况。这可能是NP完全问题。
- 如果问题涉及序列（如旅行商问题中的城市序列）且难以解决，它可能就是NP完全问题。
- 如果问题涉及集合（如广播台集合）且难以解决，它可能就是NP完全问题。
- 如果问题可转换为集合覆盖问题或旅行商问题，那它肯定是NP完全问题。

8.5 小结

- 贪婪算法寻找局部最优解，企图以这种方式获得全局最优解。
- 对于NP完全问题，还没有找到快速解决方案。
- 面临NP完全问题时，最佳的做法是使用近似算法。
- 贪婪算法易于实现、运行速度快，是不错的近似算法。

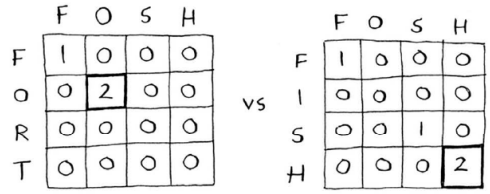
9 动态规划

9.1 - 9.2 背包问题

动态规划功能强大，它能够解决子问题并使用这些答案来解决大问题。但仅当每个子问题都是离散的，即不依赖于其他子问题时，动态规划才管用。

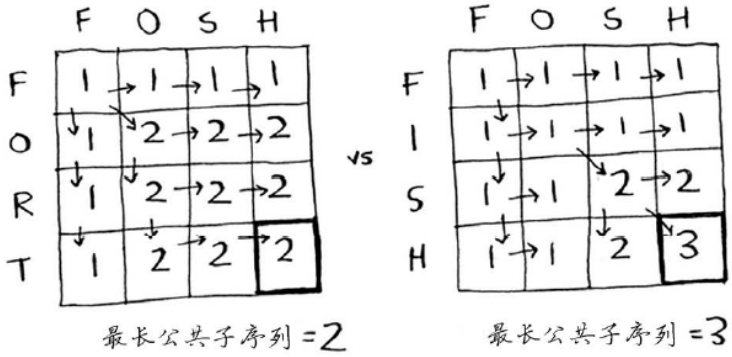
9.3 最长公共子串

我们使用最长公共子串公式来比较它们。



最长公共子串的长度相同，都包含两个字母！但fosh与fish更像。

区别于 最长公共子序列



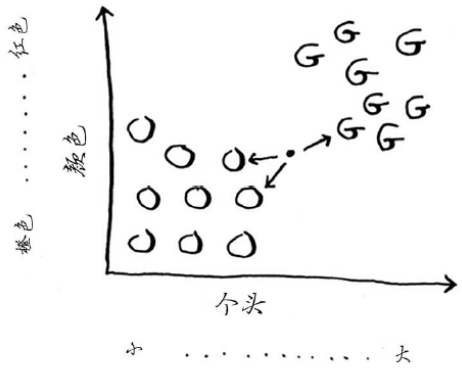
应用场景

- 生物学家根据最长公共序列来确定DNA链的相似性，进而判断度两种动物或疾病有多相似。最长公共序列还被用来寻找多发性硬化症治疗方案。
- 你使用过诸如`git diff`等命令吗？它们指出两个文件的差异，也是使用动态规划实现的。
- 前面讨论了字符串的相似程度。编辑距离（levenshtein distance）指出了两个字符串的相似程度，也是使用动态规划计算得到的。编辑距离算法的用途很多，从拼写检查到判断用户上传的资料是否是盗版，都在其中。
- 你使用过诸如Microsoft Word等具有断字功能的应用程序吗？它们如何确定在什么地方断字以确保行长一致呢？使用动态规划！

10 K最近邻算法（k-nearest neighbours，KNN）

10.1 橙子还是柚子？

如果判断这个水果是橙子还是柚子呢？一种办法是看它的邻居。来看看离它最近的三个邻居。



在这三个邻居中，橙子比柚子多，因此这个水果很可能是橙子。祝贺你，你刚才就是使用K最近邻（k-nearest neighbours，KNN）算法进行了分类！这个算法非常简单。

10.2 创建推荐系统

1 特征抽取

2 回归 regression

假设你要预测Priyanka会给电影*Pitch Perfect*打多少分。Justin、JC、Joey、Lance和Chris都给它打了多少分呢？

JUSTIN：5
JC：4
JOEY：4
LANCE：5
CHRIS：3

你求这些人打的分的平均值，结果为4.2。这就是回归（regression）。你将使用KNN来做两项基本工作——分类和回归：

- 分类就是编组；
- 回归就是预测结果（如一个数字）。

3 挑选合适的特征

10.3 机器学习简介

10.3.1 OCR

OCR指的是光学字符识别（optical character recognition），这意味着你可拍摄印刷页面的照片，计算机将自动识别出其中的文字。Google使用OCR来实现图书数字化。OCR是如何工作的呢？我们来看一个例子。请看下面的数字。

如何自动识别出这个数字是什么呢？可使用KNN。

(1) 浏览大量的数字图像，将这些数字的特征提取出来。

(2) 遇到新图像时，你提取该图像的特征，再找出它最近的邻居都是谁！

11 接下来如何做

11.1 树

二叉查找树 binary search tree

11.2 反向索引

反向索引（inverted index），常用于创建搜索引擎。

11.3 傅里叶变换

11.4 并行算法

11.5 MapReduce

MapReduce是一种流行的分布式算法

11.6 布隆过滤器和HyperLogLog

11.7 SHA 算法

安全散列算法（secure hash algorithm，SHA）函数。给定一个字符串，SHA返回其散列值。

11.8 局部敏感的散列算法

11.9 Diffie-Hellman 密钥交换

11.10 线性规划