高级数据结构 /Advanced Data Structure

- 优先队列 / Priority Queue
- 图 / Graph
- 前缀树 / Trie
- 线段树 / Segment Tree
- 树状数组 / Fenwick Tree / Binary Indexed Tree

优先队列 / Priority Queue

与普通队列的区别

保证每次取出的元素是队列中优先级最高的

优先级别可自定义

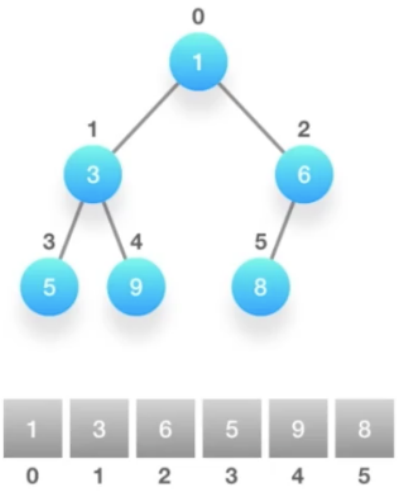
最常用的场景

从杂乱无章的数据中按照一定的顺序（或者优先级）筛选数据

本质

二叉堆的结构，堆在英文里叫 Binary Heap

利用一个数组结构来实现完全二叉树



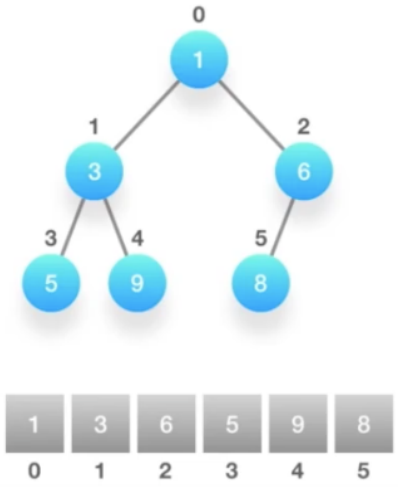
特性

数组里的第一个元素 `array[0]` 拥有最高的优先级

给定一个下标 `i`，那么对于元素 `array[i]` 而言

- **父节点** 对应的元素下标是 `(i-1)/2`
- **左侧子节点** 对应的元素下标是 `2*i + 1`
- **右侧子节点** 对应的元素下标是 `2*i + 2`

数组中每个元素的优先级都必须要高于它两侧子节点



其基本操作为以下两个

向上筛选（sift up / bubble up）

向下筛选（sift down / bubble down）

另一个最重要的时间复杂度：优先队列的初始化

$$T(n) = \sum_{h=0}^{log(n)} [\frac{n}{2^{h+1}}] * O(h)$$

$$= O(n * \sum_{h=0}^{log(n)} \frac{h}{2^h})$$

$$= O(n * \sum_{h=0}^{\infty} \frac{h}{2^h})$$

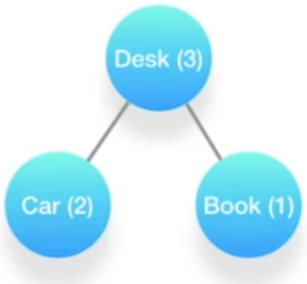
347. 前 K 个高频元素

给定一个非空的整数数组，返回其中出现频率前 ***k*** 高的元素。

说明：

你可以假设给定的 *k* 总是合理的，且 1 ≤ k ≤ 数组中不相同的元素的个数。

你的算法的时间复杂度**必须**优于 *O(nlogn)* ，*n* 是数组的大小。



示例：car, car, book, desk, desk, desk

图 / Graph

围绕图的算法也是各式各样

图的遍历：深度优先、广度优先

环的检测：有向图、无向图

拓扑排序

最短路径算法：Dijkstra、Bellman-Ford、Floyd Warshall

连通性相关算法：Kosaraju、Tarjan、求解孤岛的数量、判断是否为树

图的着色、旅行商问题等

必需熟练掌握的知识点

图的存储和表达方式：邻接矩阵、邻接链表

图的遍历：深度优先、广度优先

二部图的检测（Bipartite）、树的检测、环的检测：有向图、无向图

拓扑排序

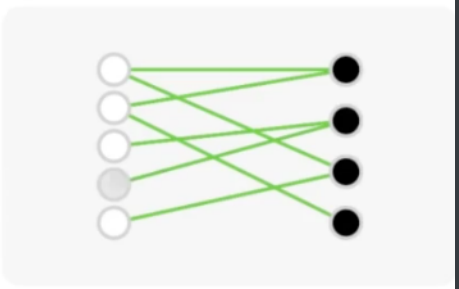
联合-查找算法（Union-Find）

最短路径：Dijkstra、Bellman-Ford

785. 判断二分图

给定一个无向图 `graph`，当这个图为二分图时返回 `true`。

如果我们能将一个图的节点集合分割成两个独立的子集 A 和 B，并使图中的每一条边的两个节点一个来自 A 集合，一个来自 B 集合，我们就将这个图称为二分图。



前缀树 / Trie

也称字典树

这种数据结构被广泛地运用在字典查找当中

什么是字典查找?

例如：给定一系列构成字典的字符串，要求在字典当中找出所有以“ABC” 开头的字符串

- **方法一**：暴力搜索法

时间复杂度：O(M*N)

- **方法二**：前缀树

时间复杂度：O(M)

经典应用

搜索框输入搜索文字，会罗列以搜索词开头的相关搜索

重要性质

每个节点至少包含两个基本属性

- children：数组或者集合，罗列出每个分支当中包含的所有字符
- isEnd: 布尔值，表示该节点是否为某字符串的结尾

根节点是空的

除了根节点，其他所有节点都可能是单词的结尾，叶子节点一定都是单词的结尾

最基本的操作

创建

方法

遍历一遍输入的字符串，对每个字符串的字符进行遍历
从前缀树的根节点开始，将每个字符加入到节点的 children 字符集当中
如果字符集已经包含了这个字符，跳过
如果当前字符是字符串的最后一个，把当前节点的 isEnd 标记为真

最基本的操作

搜索

方法

从前缀树的根节点出发，逐个匹配输入的前缀字符
如果遇到了，继续往下一层搜索
如果没遇到，立即返回

212. 单词搜索 II

给定一个二维网格 board 和一个字典中的单词列表 words，找出所有同时在二维网格和字典中出现的单词。

单词必须按照字母顺序，通过相邻的单元格内的字母构成，其中“相邻”单元格是那些水平相邻或垂直相邻的单元格。同一个单元格内的字母在一个单词中不允许被重复使用。

说明:
你可以假设所有输入都由小写字母 a-z 组成。

线段树 / Segment Tree

力扣 LeetCode

先从一个例题出发

假设我们有一个数组 array[0 … n-1]， 里面有 n 个元素，现在我们要经常对这个数组做两件事：

1. 更新数组元素的数值
2. 求数组任意一段区间里元素的总和（或者平均值）

- **方法一**：遍历一遍数组

时间复杂度：O(n)

- **方法二**：线段树

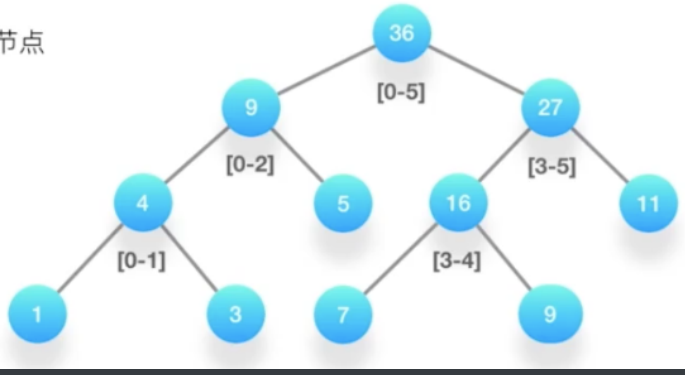
时间复杂度：O(logn)

什么是线段树

一种按照二叉树的形式存储数据的结构，每个节点保存的都是数组里某一段的总和

例如

数组是 [1, 3, 5, 7, 9, 11]



315. 计算右侧小于当前元素的个数

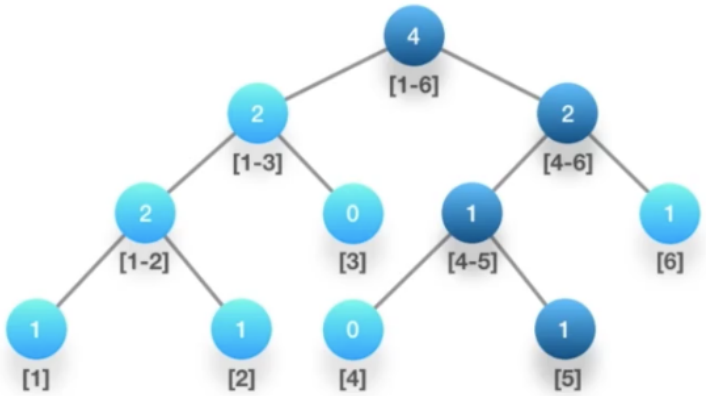
给定一个整数数组 nums，按要求返回一个新数组 counts，使得数组 counts 有该性质：counts[i] 的值是 nums[i] 右侧小于 nums[i] 的元素的数量。

示例：
输入：[5, 2, 6, 1]
输出：[2, 1, 1, 0]

解释：
5 的右侧有 2 个更小的元素（2 和 1）
2 的右侧仅有 1 个更小的元素（1）
6 的右侧有 1 个更小的元素（1）
1 的右侧有 0 个更小的元素

315. 计算右侧小于当前元素的个数

[5 2, 6, 1]



树状数组 / Fenwick Tree / Binary Indexed Tree

也被称为 Binary Indexed Tree

先从一个例题出发

假设我们有一个数组 array[0 … n-1]， 里面有 n 个元素，现在我们要经常对这个数组做两件事：

1. 更新数组元素的数值
2. 求数组前 k 个元素的总和（或者平均值）

- **方法一**：线段树
  时间复杂度： O(logn)
- **方法二**：树状数组
  时间复杂度：O(logn)

树状数组 / Fenwick Tree / Binary Indexed Tree

重要的基本特征

利用数组来表示多叉树的结构，和优先队列有些类似

优先队列是用数组来表示完全二叉树，而树状数组是多叉树

树状数组的第一个元素是空节点

如果节点 tree[y] 是 tree[x] 的父节点，那么需要满足 y = x - (x & (-x))

拓展练习

力扣 308. 二维区域和检索 - 可变

求一个动态变化的二维矩阵里，任意子矩阵里的数的总和

本节课总结

优先队列：常见面试考点，实现过程比较繁琐。在解决面试中的问题时，实行“拿来主义”即可

图：被广泛运用的数据结构，如大数据问题都得运用图论

- 在社交网络里，每个人可以用图的顶点表示，人与人直接的关系可以用图的边表示
- 在地图上，要求解从起始点到目的地，如何行驶会更快捷，需要运用图论里的最短路径算法

前缀树：出现在面试的难题中，要求能熟练地书写它的实现以及运用

线段树和树状数组：应用场合比较明确
如果要求在一幅图片当中修改像素的颜色，求解任意矩形区间的灰度平均值，则需要采用二维的线段树