数据结构 - 栈，队列，哈希表与堆
Stack, Queue, Hash & Heap

课程版本 v5.0　　主讲 令狐冲

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
知乎: http://zhuanlan.zhihu.com/jiuzhang
官网: http://www.jiuzhang.com

先修知识 & 大纲 Outline

本章节的先修知识有：

什么是数据结构

什么结构类问题的三种考法

如何衡量数据结构类问题的时间复杂度

队列及相关面试问题

栈及相关面试问题

数据结构类面试问题的三种考法

数据结构可以认为是一个数据存储集合以及定义在这个集合上的若干操作(功能)

他有如下的三种考法:

考法1:问某种数据结构的基本原理，并要求实现

例题:说一下 Hash 的原理并实现一个 Hash 表

考法2:使用某种数据结构完成事情

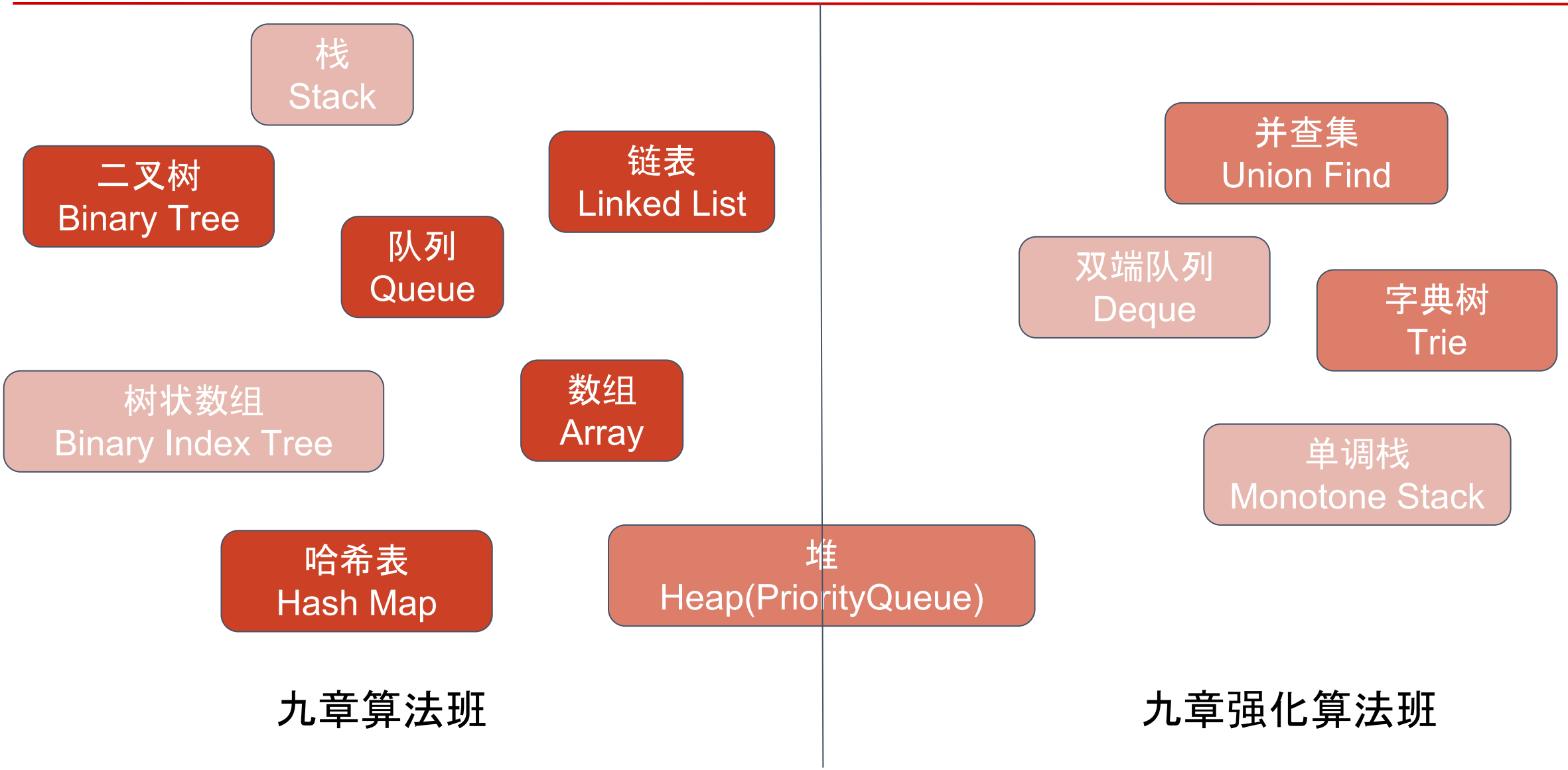
例题:归并 K 个有序数组

考法3:实现一种数据结构，提供一些特定的功能

例题:最高频 K 项问题

通常需要一个或者多个数据结构配合在一起使用

面试中可能会考察的数据结构



队列 Queue

支持操作：O(1) Push / O(1) Pop / O(1) Top

队列的基本作用就是用来做 BFS

Sliding Window Average from Data Stream

http://www.lintcode.com/problem/sliding-window-average-from-data-stream/

http://www.jiuzhang.com/solutions/sliding-window-average-from-data-stream/

又名：Moving Average from Data Stream

更多 Sliding Window 相关的面试难题

将在《九章算法强化班》中讲解

Sliding Window Median

Sliding Window Maximum

Sliding Window Matrix Maximum

Sliding Window Unique Element Sum

栈 Stack

支持操作：O(1) Push / O(1) Pop / O(1) Top

非递归实现DFS的主要数据结构

单调栈 Monotone Stack

将在《九章算法强化班》中讲解

数据结构时间复杂度的衡量方法

数据结构通常会提供“**多个**”对外接口

只用一个时间复杂度是很难对其进行正确评价的

所以通常要对每个接口的时间复杂度进行描述

比如你需要设计一个 Set 的数据结构，提供 lowerBound 和 add 两个方法。lowerBound 的意思是，找到比某个数小的最大值

算法1：O(n) lowerBound O(1) add

使用数组存储，每次打擂台进行比较，插入就直接插入到数组最后面

算法2：O(logn) lowerBound O(logn) add

使用红黑树（Red-black Tree）存储，Java 里的 TreeSet，C++ 里的 map

上面两个算法谁好谁坏呢？

不一定谁好谁坏！要看这两个方法被调用的频率如何。

如果 lowerBound 很少调用, add 非常频繁, 则算法1好。

如果 lowerBound 和 add 调用的频率都差不多，或者 lowerBound 被调用得更多，则算法2好

不过通常来说，在面试中的问题，我们会很容易找到一个像算法1这样的实现方法，其中一个操作时间复杂度很大，另外一个操作时间复杂度很低。

通常的解决办法都是想办法增大较快操作的时间复杂度来加速较慢操作的时间复杂度。

哈希表 Hash

支持操作：O(1) Insert / O(1) Find / O(1) Delete

问：这些操作都是 O(1) 的前提条件是什么？

独孤九剑 —— 破箭式

哈希表（HashMap / unordered_map / dict）

任何操作的时间复杂度从严格意义上来说

都是 O(keySize) 而不是 O(1)

请在随课教程中学习如下先修知识

Hash Table, Hash Map 和 Hash Set 的区别是啥

什么是 Hash Function (产生HashCode的函数), 作用以及实现原理

什么是 Open Hashing 什么是 Closed Hashing

什么是 Rehashing（重哈希）

LRU Cache

http://www.lintcode.com/problem/lru-cache/

http://www.jiuzhang.com/solutions/lru-cache/

Example: [2 1 3 2 5 3 6 7]

LRU Cache

- LinkedHashMap = DoublyLinkedList + HashMap

- HashMap<key, DoublyListNode> DoublyListNode {
- prev, next, key, value;
- }

- Newest node append to tail.
- Eldest node remove from head.

问：Singly List 是否可行？

Singly List 是否可行？

可以，在 Hash 中存储 Singly List 中的 prev node 即可

如 linked list = dummy->1->2->3->null 时

hash[1] = dummy, hash[2] = node1 ...

Insert Delete GetRandom O(1)

http://www.lintcode.com/problem/insert-delete-getrandom-o1/

http://www.jiuzhang.com/solutions/insert-delete-getrandom-o1/

一样的题：http://www.lintcode.com/problem/load-balancer/

Follow up: 允许重复的数

http://www.lintcode.com/problem/insert-delete-getrandom-o1-duplicates-allowed/

http://www.jiuzhang.com/solutions/insert-delete-getrandom-o1-duplicates-allowed/

实现较为困难，看懂参考代码即可，不用太纠结

99%的人面试的时候都做不出来，面试时通常给个思路就可以了

Insert Delete GetRandom O(1) 面试评分标准

Strong Hire:

Bug Free 的实现无重复版本的代码，并给出有重复版的基本思路即可

Hire / Weak Hire:

实现无重复版本的代码，无需太多提示或者较少提示，代码 Bug 不多

No Hire / Strong No:

无法无重复版本的给出正确算法，或者无法正确实现，代码 Bug 过多

First Unique Character in a String

http://www.lintcode.com/problem/first-unique-character-in-a-string/

http://www.jiuzhang.com/solutions/first-unique-character-in-a-string/

Follow up: 如果是 Data Stream 怎么办？

什么是 Data Stream 类问题?

只允许遍历一次！

http://www.lintcode.com/problem/middle-of-linked-list/

http://www.lintcode.com/problem/first-unique-number-in-stream/

Related Questions

- http://www.lintcode.com/problem/subarray-sum/
- http://www.lintcode.com/problem/copy-list-with-random-pointer/
- http://www.lintcode.com/problem/anagrams/
- http://www.lintcode.com/problem/longest-consecutive-sequence/

休息 5 分钟

总结一道题的经验，胜过刷十道题
把你的代码和总结发到九章面试题交流社区
www.jiuzhang.com/solutions

Heap

支持操作：O(log N) Add / O(log N) Remove / O(1) Min or Max

Max Heap vs Min Heap

PriorityQueue vs Heap

Heap 的基本原理和具体实现

请见随课教程

Ugly Number II

http://www.lintcode.com/problem/ugly-number-ii/

http://www.jiuzhang.com/solutions/ugly-number-ii/

Merge K Sorted Lists

http://www.lintcode.com/problem/merge-k-sorted-lists/

http://www.jiuzhang.com/solution/merge-k-sorted-lists/

方法一：使用 PriorityQueue

方法二：类似归并排序的分治算法

方法三：自底向上的两两归并算法

时间复杂度均为 O(NlogK)

Strong Hire:

能够用至少2种方法进行实现，代码无大 BUG

Hire / Weak Hire:

能够用一种方法进行实现，对其他的方法提出正确想法，代码无 Bug 或者少 Bug

No Hire / Strong No:

无法用正确的算法实现，或者代码 BUG 很多

K Closest Points

http://www.lintcode.com/problem/k-closest-points/

http://www.jiuzhang.com/solutions/k-closest-points/

Top K Largest Number II

http://www.lintcode.com/problem/top-k-largest-numbers-ii/

http://www.jiuzhang.com/solutions/top-k-largest-number-ii/

Follow up: Top K Frequent Elements

Top K Largest Number II 面试评分标准

Strong Hire:

能完整实现代码，时间复杂度最优，代码没有大 Bug，无需提示

能够对 Follow up 问题提出解决方案（不一定要实现）

Hire / Weak Hire:

能够完整实现代码，代码 Bug 不多

无需提示或者很少需要提示做出最优复杂度的版本

No Hire / Strong No:

无法用最优的算法实现出来

怎么解决 Top K Frequent Elements 问题？

这是海量数据类面试题最经典的一类问题

包含了非常多的算法和解决办法

详情请看九章微课堂——《海量数据处理算法与面试题全集》

http://www.jiuzhang.com/tutorial/big-data-interview-questions

Related Questions

- http://www.lintcode.com/en/problem/high-five/ (A)
- http://www.lintcode.com/problem/merge-k-sorted-arrays/
- http://www.lintcode.com/problem/data-stream-median/
- http://www.lintcode.com/problem/top-k-largest-numbers/
- http://www.lintcode.com/problem/kth-smallest-number-in-sorted-matrix/