如何学好算法和数据结构

Mark Chen

大纲

- 有哪些需要学习的算法与数据结构
- 如何学好算法与数据结构 - 刷题的重要性
- 算法与数据结构在工作中的应用
- 总结 - 有哪些经典的解题模板

算法为什么离不开数据结构

- 算法是解决问题的一系列操作集合
- 数据结构能使得这些操作更加的高效
- 同样的算法我们可以选择不同的数据结构，会带来不同效率的算法

有哪些需要学习的算法与数据结构

国外主流IT企业面试：算法数据结构 + 系统设计面

国内主流IT企业面试：算法数据结构 + 系统设计面 + 操作系统 + 网络 + 数据库 + ...

面试中的算法和数据结构并不是很多，常见的有：

1. Array（如各种Subarray问题）
2. LinkedList（各种翻转操作链表问题）
3. Queue
4. Stack
5. Binary Tree
6. .....

有哪些需要学习的算法与数据结构

算法与数据结构 - 面试版

VS

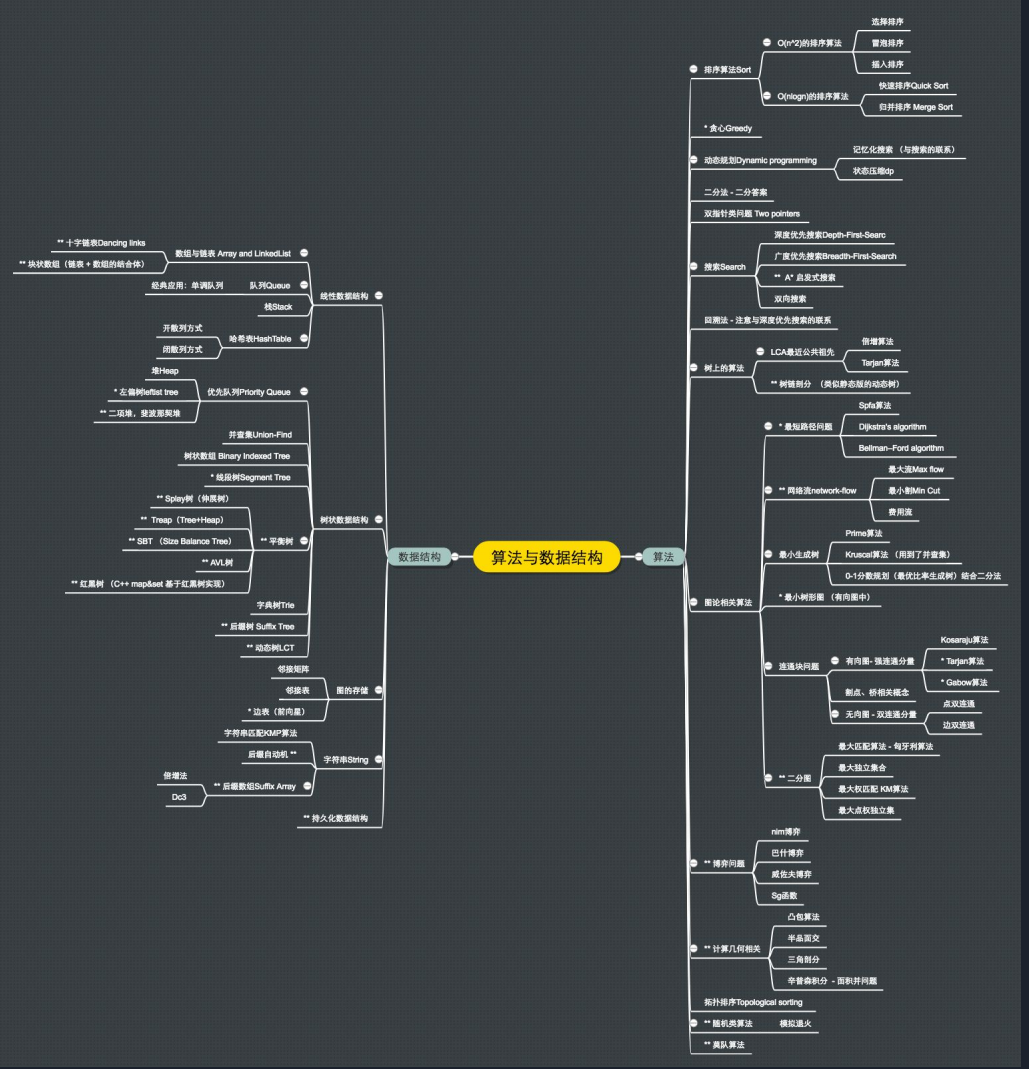
算法与数据结构 - 竞赛版

（如两张图）

算法与数据结构 - 面试版


算法与数据结构
数据结构
Subarray问题
数组Array
前缀和Prefix Sum
链表LinkedList
队列Queue
栈Stack
* 堆Heap
开散列
哈希表HashTable
闭散列
* 并查集Union-find
二叉树Binar Tree
字典树Trie
算法
排序算法
快速排序Quick Sort
归并排序Merge sort
动态规划Dynamic programming
二分法
两根指针Two pointer
搜索Search
深度优先搜索Depth-First-Search
广度优先搜索Breath-First-Search
回溯法

算法与数据结构 - 竞赛版


算法与数据结构
数据结构
线性数据结构
** 十字链表Dancing links
数组与链表 Array and LinkedList
** 块状数组（链表 + 数组的结合体）
经典应用：单调队列
队列Queue
栈Stack
哈希表HashTable
开散列方式
闭散列方式
优先队列Priority Queue
堆Heap
* 左偏树leftist tree
** 二项堆，斐波那契堆
树状数组结构
并查集Union-Find
树状数组 Binary Indexed Tree
* 线段树Segment Tree
** 平衡树
** Splay树（伸展树）
** Treap（Tree+Heap）
** SBT（Size Balance Tree）
** AVL树
** 红黑树 （C++ map&set 基于红黑树实现）
字典树Trie
** 后缀树 Suffix Tree
** 动态树LCT
图的存储
邻接矩阵
邻接表
* 边表（前向星）
字符串String
字符串匹配KMP算法
后缀自动机 **
** 后缀数组Suffix Array
倍增法
Dc3
** 持久化数据结构
算法
排序算法Sort
O(n^2)的排序算法
选择排序
冒泡排序
插入排序
O(nlogn)的排序算法
快速排序Quick Sort
归并排序 Merge Sort
* 贪心Greedy
动态规划Dynamic programming
记忆化搜索 （与搜索的联系）
状态压缩dp
二分法 - 二分答案
双指针类问题 Two pointers
搜索Search
深度优先搜索Depth-First-Searc
广度优先搜索Breadth-First-Search
** A* 启发式搜索
双向搜索
回溯法 - 注意与深度优先搜索的联系
树上的算法
LCA最近公共祖先
倍增算法
Tarjan算法
** 树链剖分 （类似静态版的动态树）
图论相关算法
* 最短路径问题
Spfa算法
Dijkstra's algorithm
Bellman–Ford algorithm
** 网络流network-flow
最大流Max flow
最小割Min Cut
费用流
最小生成树
Prime算法
Kruscal算法 （用到了并查集）
0-1分数规划（最优比率生成树）结合二分法
* 最小树形图（有向图中）
连通块问题
有向图- 强连通分量
Kosaraju算法
* Tarjan算法
* Gabow算法
割点、桥相关概念
无向图 - 双连通分量
点双连通
边双连通
** 二分图
最大匹配算法 - 匈牙利算法
最大独立集合
最大权匹配 KM算法
最大点权独立集
** 博弈问题
nim博弈
巴什博弈
威佐夫博弈
Sg函数
** 计算几何相关
凸包算法
半品面交
三角剖分
辛普森积分 - 面积并问题
拓扑排序Topological sorting
** 随机类算法
模拟退火
** 莫队算法

学习的广度和深度

1. 广度和深度并重
2. 先广度(即系统化学习)，了解数据结构直接的联系
3. 后纵深，深入的挖掘每一种数据结构的应用

数据结构分类

- 线性数据结构 1对1的关系
  - 数组&链
  - 栈和队列
- 树状数据结构 1对多的关系
- 图结构 多对多的数据结构

了解数据结构的特性

1. 数组Array
    a. 可随机访问，可以访问每一个位置，如 A[i]
    b. 前缀和，Prefix Sum - SubArray问题的杀手锏
2. 链表LinkedList
    a. 翻转链表一系列问题，本质是你是否了解 LinkedList
3. 树结构Tree
    a. Tree与LinkedList的关系
        i. LinkedList本质是一叉树
        ii. Tree本质是LinkedList的变种
4. 堆Heap
    a. 优先队列的一种，能够在logn时间复杂度取出一个集合的最小值或者最大值

举例 ListNode 和 TreeNode的定义比较

```python
class ListNode:

    def __init__(self, val):
        self.val = val
        self.next = None


class TreeNode:

    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
        # 本质等于 self.next1, self.next2 = None, None
```

刷题的重要性

1. 检验学习的正确性，实践自己所学

2. 量的积累

   - 没有量的积累，再科学的方法也没有卵用！

刷题的重要性

3. 科学的刷题

a. 给自己20-30分钟的思考时间，why？
b. 学会分类与总结，why？LintCode题目上有tag
    i. http://www.lintcode.com/en/tag/
    ii. 按照专题，类别刷题来学习一个知识点
    iii. **即使自己能够实现该算法或者数据结构，也要看看他们好的代码。同样的算法和数据结构，不同的人实现出来效率上也会有差别**
c. 笔记，而不是做完Accepted后就丢在一边
d. Bug Free的重要性
    i. 需要有时间的限制
    ii. 需要有提交次数的限制

刷题的重要性 - 锻炼程序的正确性

1. 在白纸和白板上写代码
    a. 抄到电脑上看是否可以通 过
2. 直接在LintCode/LeetCode上提交，不做调试

哪里刷题 ？

OI & ACM 竞赛类：

- POJ，ZOJ，HDU etc

面试：

- LintCode/LeetCode
- 面试不推荐去刷上述OJ

算法的设计

1. 了解每一个经典算法的使用场景
2. 在这个特定的场景下，把该算法使用熟练，了解其时空复杂度
    a. 排序算法 - sort integer问题
    b. 回溯法 - subset和排列问题
    c. 动态规划 - 背包问题
3. 算法的设计，突破口来源于问题本身：
    a. 合并两个有序数组 - 有序是一个性质
    b. 在行列都递增的矩阵中找一个数是否出现过 - 行列递增是一个性质
4. 想一个“笨”办法，从这个方法开始不断优化
    a. 找出由原来问题的瓶颈，即他为什么“笨”，能否优化
    b. 不能优化，再找另外“笨”的办法
    c. 举例子：从搜索到动态规划

数据结构 - 算法加速的手段

1. 数据结构的原理和每一种基本操作
    a. 并查集
        i. 判断两个元素是否在同一个集合内
        ii. 合并两个元素所在的集合
    b. Stack
        i. 满足元素先进后出的性质
2. 算法中的某些操作能否转变成特定的数据结构的基本操作
    a. http://www.lintcode.com/en/problem/connecting-graph-iii/
        i. 题目大意：
            1. 给出一个点的集合，可以合并两个点所在的集合
            2. 查询当前有多少个集合
    b. 查询当前有多少个集合，这个操作并没有出现在并查集的基本操作里面

数据结构 - 算法的辅助工具

问题：http://www.lintcode.com/en/problem/number-of-islands-ii/

经典面试题岛屿问题：

1. 创建一个小岛，相邻的岛屿连接起来，形成一个块
2. 每次会询问，当前有多少块
3. Follow up: 如果创建岛屿的操作可以撤销？
    a. 跟面试官交流之后发现 - 撤销是按照创建的顺序撤销的
    b. 先创建后撤销，我们想到了“先进先出”，Stack，用栈去维护当前的信息即可

算法与数据结构在工作中的应用

Queue 作为消息队列的应用

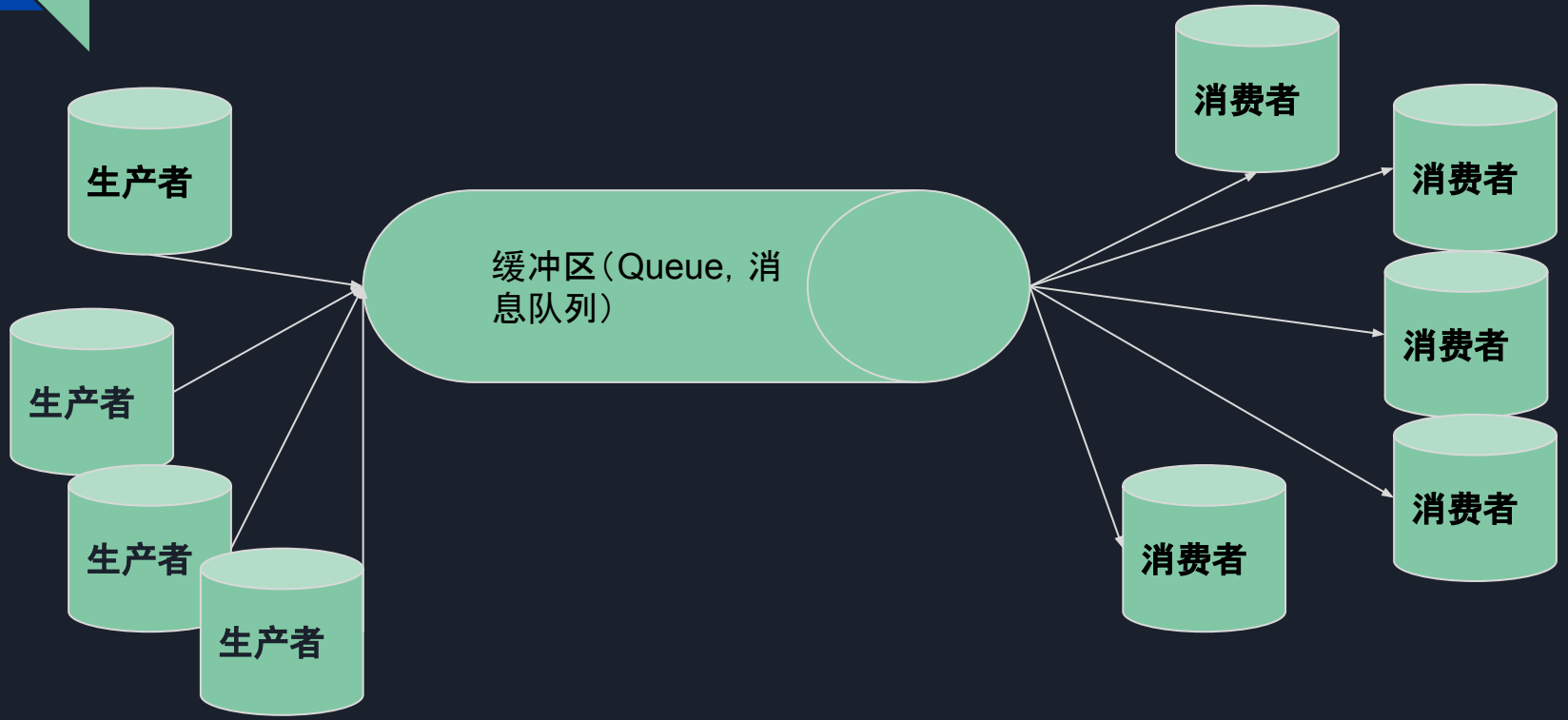
Google的protocol buffer实现序列化与反序列

- https://developers.google.com/protocol-buffers/
- 序列化与反序列化数据结构：
  - http://www.lintcode.com/en/problem/binary-tree-serialization/
  -

LRU算法

- 数组与链表的应用

生产者消费者模式(Producer-Consumer)



总结 - 有哪些经典的解题模板

二分法模板

广度优先搜索模板：
http://www.jiuzhang.com/solution/bfs-template

形成代码模板的好处：
1. 每次写保持统一的方式和风格，减少出错（每写一次code不同，出错的概率太高了，面试要求bug free！）

2. 在该算法模板上形成流式思维

书籍推荐

算法与数据结构相关书籍：

- 没有推荐的书籍

非算法与数据结构相关书籍：

- 《深入理解计算机系统》（如果你愿意坚持看下来的话）