Leetcode 总结 （基于Python3）

面试题目主要考察 `数据结构` (Data Structure)， `算法` (Algorithm)，以及 `套路` (Pattern)

资源

- William Fiset 讲解数据结构
- Algorithms with Attitude 讲数据结构
- 花花酱 讲 leetcode
- Tushar Roy 讲 leetcode
- Irfan Baqui 讲 leetcode
- Algorithm Live（偏难）
- 数据结构和算法可视化

刷题

1. 一开始先忽略 hard 题，实习面试一般考 medium/easy 题，全职面试可能考 hard 题
2. 每个 topic 不必全刷，可以挑几个题目做一下，了解思路和套路
3. 可以先刷 `Linked-List, Binary Tree, Binary Search Tree`，这个过程可以练习 `指针`， `DFS` 和 `BFS`

    1. `Linked-List` 主要考察 `指针` 操作
    2. `Tree` 一般就是 `DFS` 或者 `BFS`，写法也是有套路的

数据结构

- `数据结构` 是数据的组织结构。我们经常需要对数据进行 `查增删改` (search, insert, delete, modify)。不同 `数据结构` 的 `查增删改` 效率不同。
- `数据结构` 的种类，以及在 `python3` 中的实现

```
< 基础 >
List（顺序表）：Array，String，Matrix
Hash（哈希表）：Dictionary（HashMap），Set（HashSet）
Queue（队列）：先进先出（first in first out）
Stack（栈）：先进后出（fisrt in last out）
Heap（堆）：又称 Priority-Queue（优先队列），始终保持最小值（或最大值）在堆顶

< 进阶 >
Linked-List（链表）：有 next 指针，可认为是一叉树
Binary Tree（二叉树）：有 left 和 right 指针
Binary Search Tree（二叉搜索树）：便于搜索
Graph（图）

< 高阶 >
UnionFind（并查集）：针对连通域（connected component）问题
Trie（前缀树/字典树）：

< 疯狂 >
Binary Indexed Tree（树状数组）：
Segment Tree（线段树）：
```

```python
# list
array = [1, 2, 3, 4]   # array[i]
string = 'abcd'        # string[i]
matrix = [[1,2,3], [4,5,6], [7,8,9]]  # matrix[i][j]

# hash map
D = dict()
D[key] = value
del D[key]

# hash set
S = set()
S.add(value)
S.remove(value)

# queue
from collections import deque
queue = deque()
queue.append(value) # push
queue.popleft()     # pop

# stack
```

```
stack = []
stack.append(value) # push
stack.pop()         # pop

# heap (version 1)
from heapq import heapify, heappush, heappop
heap = []
heappush(heap, (0, 'a'))
heappush(heap, (2, 'c'))
heappush(heap, (1, 'b'))
print(heappop(heap), heappop(heap), heappop(heap))

# heap (version 2)
from queue import PriorityQueue
heap = PriorityQueue()
heap.put((0, 'a'))
heap.put((2, 'c'))
heap.put((1, 'b'))
print(heap.get(), heap.get(), heap.get())

# Linked-List
node.next  # 下一个节点

# Binary Tree
node.left  # 左子节点
node.right # 右子节点

# Graph
node.neighbors # 相邻节点
```

算法

```
< 搜索: Medium >
DFS（深度优先搜索）: 树和图的问题一般用 DFS/BFS 解决
BFS（宽度优先搜索）:
Binary Search（二分搜索）:
检验 mid 是否满足条件，通过排除法，改变left/right，找最左元素，or，找最右元素
(判断口诀: 元素越左越正确，找最右；元素越右越正确，找最左)

< 搜索: Hard >
Backtracking（回溯搜索）: 暴力枚举

< 排序 >
Merge Sort（归并排序）:
Quick Sort（快排）:
Quick Select（快选）:

< 思路 >
Greedy（贪婪）:
Divide & Conquer（分治）:
Dynamic Programming（动态规划）: 存储子问题答案，避免重复运算

< 双指针: Medium >
Meeting Pointers: 左指针指起始，右指针指末尾，通过排除法选择，走左指针或右指针
Tail Pointer:（洗牌问题）左右指针从起始出发，右指针遍历数组，左指针指向一类牌的末尾

< 双指针: Hard >
Sliding Window（滑动窗口）:
左右指针从起始出发，左右指针形成窗口，根据窗口状态选择，移动左指针或右指针。
（判断口诀: 序列越短越正确，找最长；序列越长越正确，找最短）
```

套路

`套路题` 是指有很多变体的类型题目，例如 `回文，括号，区间，子串子序列，矩形，随机` 等等。
把一个类型的题目放在一起讨论，更便于对比和复习。