学习算法和刷题的框架思维

笔记本:	labuladong		
创建时间:	2022/1/5 11:28	**更新时间:**	2022/1/5 17:24
作者:	mdq1357@163.com		
URL:	https://labuladong.gitee.io/algo/1/2/		

学习算法和刷题的框架思维

- 数据结构的存储方式

1. 数据结构的存储方式只有两种：数组（顺序存储）和链表（链式存储）。数组和链表是「结构基础」。因为那些多样化的数据结构，究其源头，都是在链表或者数组上的特殊操作，API 不同而已。
2. 数据结构种类很多，但是底层存储无非数组或者链表，二者的优缺点如下：

数组由于是紧凑连续存储,可以随机访问，通过索引快速找到对应元素，而且相对节约存储空间。但正因为连续存储，内存空间必须一次性分配够，所以说数组如果要扩容，需要重新分配一块更大的空间，再把数据全部复制过去，时间复杂度 O(N)；而且你如果想在数组中间进行插入和删除，每次必须搬移后面的所有数据以保持连续，时间复杂度 O(N)。

链表因为元素不连续，而是靠指针指向下一个元素的位置，所以不存在数组的扩容问题；如果知道某一元素的前驱和后驱，操作指针即可删除该元素或者插入新元素，时间复杂度 O(1)。但是正因为存储空间不连续，你无法根据一个索引算出对应元素的地址，所以不能随机访问；而且由于每个元素必须存储指向前后元素位置的指针，会消耗相对更多的储存空间。

- 数据结构的基本操作

1. 各种数据结构的遍历 + 访问无非两种形式：线性的和非线性的。线性就是 for/while 迭代为代表，非线性就是递归为代表。

- 算法刷题指南

```
void traverse(TreeNode root) {
// 前序遍历代码位置
traverse(root.left);
// 中序遍历代码位置
traverse(root.right);
// 后序遍历代码位置
}
```

动态规划解题套路框架

首先，动态规划问题的一般形式就是求最值。求解动态规划的核心问题是穷举。
动态规划的三要素：重叠子问题、最优子结构、状态转移方程。写出状态转移方程是最困难的，一个思维框架如下：

明确 base case -> 明确「状态」-> 明确「选择」 -> 定义 dp 数组/函数的含义

```
# 初始化 base case
dp[0][0][...] = base
# 进行状态转移
for 状态1 in 状态1的所有取值：
    for 状态2 in 状态2的所有取值：
        for ...
```

```
dp[状态1][状态2][...] = 求最值(选择1，选择2...)
```

- 斐波那契数列

但凡遇到需要递归的问题，最好都画出递归树，这对分析算法的复杂度，寻找算法低效的原因都有巨大帮助。