# **一**

## 1 数据的存储方式

**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）**。

数据结构种类很多，甚至你也可以发明自己的数据结构，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

其基本操作无非遍历 + 访问，再具体一点就是：增删查改

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。

数组遍历框架，典型的线性迭代结构：

void traverse(int[] arr) {

for (int i = 0; i < arr.length; i++) {

// 迭代访问 arr[i]

}

}

链表遍历框架，兼具迭代和递归结构：

/\* 基本的单链表节点 \*/

class ListNode {

int val;

ListNode next;

}

void traverse(ListNode head) {

for (ListNode p = head; p != null; p = p.next) {

// 迭代访问 p.val

}

}

void traverse(ListNode head) {

// 递归访问 head.val

traverse(head.next)

}

二叉树遍历框架，典型的非线性递归遍历结构：

/\* 基本的二叉树节点 \*/

class TreeNode {

int val;

TreeNode left, right;

}

void traverse(TreeNode root) {

traverse(root.left)

traverse(root.right)

}

二叉树框架可以扩展为 N 叉树的遍历框架：

/\* 基本的 N 叉树节点 \*/

class TreeNode {

int val;

TreeNode[] children;

}

void traverse(TreeNode root) {

for (TreeNode child : root.children)

traverse(child);

}

## 算法刷题指南

直接说具体的建议：

**先刷二叉树，先刷二叉树，先刷二叉树**！

**不要小看这几行破代码，几乎所有二叉树的题目都是一套这个框架就出来了**。

void traverse(TreeNode root) {

// 前序遍历

traverse(root.left)

// 中序遍历

traverse(root.right)

// 后序遍历

}

对于一个理解二叉树的人来说，刷一道二叉树的题目花不了多长时间。那么如果你对刷题无从下手或者有畏惧心理，不妨从二叉树下手，前 10 道也许有点难受；结合框架再做 20 道，也许你就有点自己的理解了；刷完整个专题，再去做什么回溯动规分治专题，**你就会发现只要涉及递归的问题，都是树的问题**。

其实很多动态规划问题就是在遍历一棵树，你如果对树的遍历操作烂熟于心，起码知道怎么把思路转化成代码，也知道如何提取别人解法的核心思路。

再看看回溯算法，前文[回溯算法详解](/algo/suan-fa-si-wei-xi-lie/hui-su-suan-fa-xiang-jie-xiu-ding-ban)干脆直接说了，回溯算法就是个 N 叉树的前后序遍历问题，没有例外。

当然，如果细节出错，你得不到正确的答案，但是只要有框架，你再错也错不到哪去，因为你的方向是对的。

但是，你要是心中没有框架，那么你根本无法解题，给了你答案，你也不会发现这就是个树的遍历问题。

这种思维是很重要的，[动态规划详解](/algo/dong-tai-gui-hua-xi-lie/dong-tai-gui-hua-xiang-jie-jin-jie)中总结的找状态转移方程的几步流程，有时候按照流程写出解法，说实话我自己都不知道为啥是对的，反正它就是对了。。。

**这就是框架的力量，能够保证你在快睡着的时候，依然能写出正确的程序；就算你啥都不会，都能比别人高一个级别。**

## 总结

数据结构的基本存储方式就是链式和顺序两种，基本操作就是增删查改，遍历方式无非迭代和递归。

刷算法题建议从「树」分类开始刷，结合框架思维，把这几十道题刷完，对于树结构的理解应该就到位了。这时候去看回溯、动规、分治等算法专题，对思路的理解可能会更加深刻一些。