	数据结构	空间
DFS (回溯,剪枝)	stack	O(h) 和高度正比
BFS	queue	$O(2^h)$ "最短路"

1,2,3　1,3,2
2,1,3　2,3,1
3,1,2　3,2,1

1 + [2,3] 的全排列
2 + [1,3] 的全排列
3 + [1,2] 的全排列

递归结构体

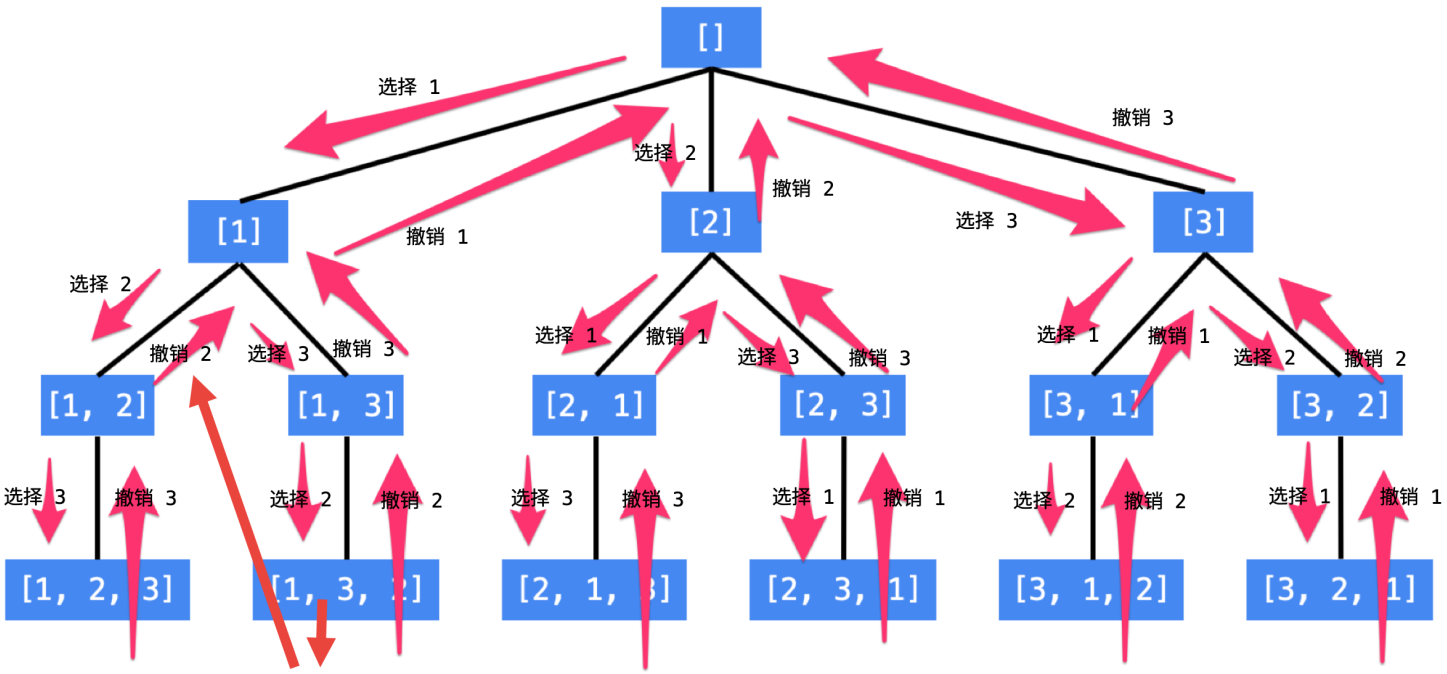
DFS (回溯)

看 leetcode 全排列

状态：每一个结点表示了求解问题的不同阶段

深度优先遍历在回到上一层结点时需"状态重置"

状态变量：① depth 递归到了第几层
② Path 已经选了哪些数
③ used 标记数组 (空间换时间)



正是因为在上一步撤销了对 2 的选择，在这一步才能选择 2，这是在深度优先遍历的过程中，需要状态重置的意义。
其它地方也是一样的，就不标注了。

回溯要 恢复现场

① DFS、递归、栈 它们背后统一的逻辑"后进先出"

② 回溯常用最简单的递归实现

③ 回溯 Vs DFS — 强调一种遍历的思想 与 BFS 对应

强调了 DFS 思想的用途

④ 很多教程把"回溯算法"称为爆搜 (暴力解法)

因此回溯算法用于搜索一个问题的所有的解，通过深度优先遍历的思想实现

DFS Vs DP

不同点：DP适合求最优解

DFS 所有解 (本质是遍历算法，时间复杂度高)