刷题算法总结

1 查找

1.1 二分查找

对应题目：

- 33. 搜索**旋转**排序数组
- **34. 在排序数组中查找元素第一个和最后一个出现位置**
- 74. 搜索二维矩阵
- 153. 寻找旋转排序数组中的最小值
- **162. 寻找峰值（涉及 Lambda 函数技巧）**

涉及到的标准库：binary_search()，upper_bound（大于），以及 lower_bound（不小于）
注意细节，ans 永远取有可能是最后结果的那个，举个例子

```
int binarySearch(vector<int>& nums, int target, bool lower) {
        int left = 0, right = (int)nums.size() - 1, ans = (int)nums.size();
        while (left <= right) {
            int mid = (left + right) / 2;
            if (nums[mid] > target || (lower && nums[mid] >= target)) {
                right = mid - 1;
                ans = mid;
            } else {
                left = mid + 1;
            }
        }
        return ans;
    }
```

另外，不一定有序数组才可以使用二分算法，例如寻找数组旋转点、寻找峰值。

2 双指针

双指针是指在遍历对象时，使用两个或多个指针进行遍历及相应的操作。大多用于数组操作，这利用了数组连序性的特点。双指针常用来降低算法的时间复杂度，因为使用两个指针可以避免多层循环。通常有对撞指针和快慢指针。

对应题目：

- **82. 删除排序链表重复元素**
- 15. 三数之和
- **844. 比较含退格字符**
- 986. 区间列表交集
- **11. 盛最多水的容器**

双指针的重点在于如何设置双指针，指针每一步该如何移动，包括方向和速度。

2.1 滑动窗口

滑动窗口就是暴力解法的优化，是在给定特定大小的字符串或者数组上操作，而不在整个数组上操作。进而减少了问题的复杂度和嵌套深度。

对应题目：

- **438. 找到字符串中所有异位子字符**
- **713. 乘积小于 k 的子数组**
- 209. 长度最小的子数组

3 BFS/DFS

3.1 岛屿网格类问题

可以与二叉树的 DFS 进行类比，可以通过额外数组进行辅助，例如存储是否已经被遍历过、改变网格值表示是否为同一个岛屿等等。

对应题目：

- 200. 岛屿数量

- 547. 省份数量
- 827. 填海造陆
- **1091. 二进制矩阵中的最短路径**注意当把它加入到 queue 的时候就直接 visited=1。
- **130. 被围绕的区域**注意 dfs 的时候首先判断要不要跳过它，跳过要注意齐全，是否满足条件，是否越界，是否以及 visited。

3.2 二叉树上的操作

BFS 的话使用 queue, 需要逐层操作的时候 while 内部要嵌套一个 for 循环
对应题目：

- 117. 填充每个节点的下一个右侧节点指针
- 572. 另一棵树的子树（有高端解法）(KMP 算法对比字符串)

4 递归/回溯

4.1 回溯

回溯的步骤：

- 寻找出递归树以及结束条件
- 找选择列表
- 判断是否剪枝
- 做出选择
- 撤销选择

```
void backtracking(参数) {
    if (终止条件) {
        存放结果;
        return;
    }

    for (选择：本层集合中元素 (树中节点孩子的数量就是集合的大小)) {
```

```
        处理节点;
        backtracking(路径, 选择列表); // 递归
        回溯, 撤销处理结果
    }
}
```

有时候也要用到 visited 避免重复搜索，注意遇到重复的时候下一步该做什么，而不是直接 return。另外，回溯时只有遍历到最后或者条件达成的时候才将结果放入到 vector 中。

对应题目：

- **797. 所有可能的路径**
- 78. 子集
- 90. 子集 II（包含重复元素）
- **46. 全排列**
- **47. 全排列 II**
- 39. 组合求和
- **40. 组合求和 II**
- 17. 电话号码
- 22. 括号生成
- 79. 单词搜索

5 动态规划

动态规划分为以下几步：

1. 确定 dp 数组及其下标的含义
2. 确定递推公式
3. dp 数组如何初始化
4. 确定遍历顺序
5. 举例推导 dp 数组

5.1 基础题目

着重注意使用滚动数组技巧。

- 509. 斐波那契数列
- **70. 爬楼梯**
- 746. 最小花费爬楼梯
- 62. 不同路径，63. 不同路径 II
- 343. 整数拆分
- **96. 不同的二叉搜索树**

5.2 01 背包问题

每个物品只能放一次，始终注意如何将一个问题转化为 01 背包问题，滚动数组要倒序遍历。

-
- **416. 分割等和**
- **1049. 最后一块石头**
- **494. 目标和**
- **474. 一和零**

5.3 完全背包问题

每个物品可以放无数次，滚动数组要正序遍历。**着重注意初始化以及递推!!!**
关于方法个数的问题，先遍历背包是排列数，先遍历物品是组合数。

- **518. 零钱兑换 II**
- **377. 组合总和 IV**
- **70. 爬楼梯加入一次可以爬 1-m 个台阶呢？排列数问题**
- 322. 零钱兑换
- **139. 单词拆分**也可以用回溯算法
- 279. 完全平方数

5.4 打家劫舍

- 198. 打家劫舍
- 213. 打家劫舍 II
- 337. 打家劫舍 III **树状 dp** dp 返回两个数，一个是不偷该节点的最大利润，一个是偷该节点的最大利润

5.5 股票

每天有多种状态，每一天每种状态所获得的最大利润都要记录下来

- **121. 买卖股票最佳时机，**只能购买一次
- **122. 买卖股票最佳时机 II，**可以无数次购买
- **123. 买卖股票最佳时机 III，**最多购买两次（4 种状态）
- **188. 买卖股票最佳时机 IV，**最多购买 k 次（2k 种状态）
- **309. 买卖股票最佳时机含冰冻期，**可以无数次购买但有一天冰冻期（4 种状态：持有股票、未持有股票但已经过了冰冻期、刚卖、冰冻期）
- **714. 买卖股票最佳时机含手续费，**和 122 很像

5.6 子序列问题

5.6.1 不连续子序列

注意 dp 的下标含义，思考为什么有时候需要记录以 i 结尾的子序列有时候不需要

- 最长上升子序列
- 最长公共子序列
- 不相交的线

5.6.2 连续子序列

注意 dp 的下标是以 i 结尾的满足要求的子序列长度

- 674. 最长连续递增序列
- 718. 最长重复子数组
- 53. 最大子序和

5.7 子序列问题

注意 dp 的下标含义

- 392. 判断子序列
- **115. 不同的子序列**
- 583. 两个字符串的删除操作
- **72. 编辑距离**

5.8 回文

dp[i][j] 表示 [i,j] 范围是否是回文或者长度

- **647. 回文子串**
- **516. 最长回文子序列长度**