**LeetCode刷题基础指南**

精选编号400以内 medium和easy为主

1. **位操作**

基本操作：

<https://leetcode.com/problems/number-of-1-bits/description/>

<https://leetcode.com/problems/power-of-two/description/>

用bits去做状态压缩：<https://leetcode.com/problems/repeated-dna-sequences/description/>

经典题：找不重复数字 <https://leetcode.com/problems/single-number-ii/description/> （如果不用位操作 怎么解决？）

经典题：斐波那契数列的O(logN)解法（高次幂的解法）

<https://leetcode.com/problems/powx-n/>

1. **链表**

基本操作：元素插入 删除 翻转 分割 归并 排序。

插入

<https://leetcode.com/problems/insertion-sort-list/description/> 思考O(NlogN)如何实现

翻转

<https://leetcode.com/problems/reverse-linked-list/description/>

<https://leetcode.com/problems/swap-nodes-in-pairs/description/>

翻转操作中包含了元素的插入，

额外增加ListNode dummy\_head以及ListNode\* cur1, cur2等多个pointer

归并

<https://leetcode.com/problems/add-two-numbers/description/>

<https://leetcode.com/problems/merge-two-sorted-lists/description/>

<https://leetcode.com/problems/merge-k-sorted-lists/description/>

多路归并需要用K大小的堆 用priority\_queue<pair<value, index>>实现pop掉top值后将对应的index的链表中头部元素入堆。

交叉点和循环

<https://leetcode.com/problems/linked-list-cycle/description/>

两个游标 一个步长1 一个步长2 求相遇

<https://leetcode.com/problems/insertion-sort-list/description/>

两个游标 从不同起点出发 到NULL之后交换起点 求相遇

链表有交点的问题，无脑建立多个cursor 用不同步长不同起点游走，最终相交

1. **二分搜索/分治**

通常题目中**条件为一个有序数组**，要求时间复杂度为O(logN)级别

基本二分搜索，**start/end节点性质在每一步要满足不变性，同时注意初始条件和循环终止条件**，注意搜索的值不存在时的处理方式。

进阶：找lowbound==第一个可以插入target的位置 和

Upperbound == 最后一个可以插入target的位置

注意以array大小为1 为2 为3时考虑边界条件。

<https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/description>

<https://leetcode.com/problems/search-insert-position/description/>

<https://leetcode.com/problems/search-a-2d-matrix/description/>

找数组的峰值和低谷

以array[i] – array[i-1]作为梯度，然后求start/end/mid的梯度后分情况递归。

梯度分情况讨论：<https://leetcode.com/problems/find-peak-element/description/>

<https://leetcode.com/problems/search-in-rotated-sorted-array/description/>

<https://leetcode.com/problems/first-bad-version/description/>

基本二分的变形

经典：只读数组O(1)空间O(N)时间找重复数字问题 <https://leetcode.com/problems/find-the-duplicate-number/description/>

经典题：两个重复数字：<https://leetcode.com/problems/single-number-iii/description/>

<https://leetcode.com/problems/h-index-ii/description/>

经典：有序矩阵中查询某个数（左下右上） <https://leetcode.com/problems/search-a-2d-matrix-ii/description/>

经典二分：有序矩阵中查第k大的数 <https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/description/>

*从数组中随机挑选一个数m，令l,s,g分别表示大于等于m的第一个数下标，大于m的第一个数下标以及大于m的最后一个数下标，然后遍历数组进行类似快排的交换，将数组分为三类，再分情况讨论。*

*ls = s; // end of <; not include*

*eq = s; // end of ==; not include*

*gr = e; // end of >; not include e = array.size() 都不包含*

*v = nums[s]; // 可以用快排的小trick防止最差情况出现*

*while(eq < gr){*

*if (nums[eq] < v){*

*swap(nums, ls, eq);*

*ls ++; eq ++;*

*}*

*else if (nums[eq] == v){ eq ++; }*

*else{*

*swap(nums, gr – 1, eq);*

*gr--;*

*}*

*} // end for*

*while(eq < gr){*

*if (nums[eq] < v){swap(nums, ls++, eq++); }*

*else if (nums[eq] == v){ eq ++; }*

*else{swap(nums, --gr , eq);}*

*} // end for*

partition的使用：<https://leetcode.com/problems/majority-element-ii/description> ←如何证明平均O(N)时间能完成？

最长上升子序列 用a[i]记录到i为止的数组中最大上升子序列的长度 用b[i]记录该最大上升子序列最后一个元素的值，可以证明b[i]具有递增的特性；对array[j]二分搜索小于等于array[j]的最大b[i]即可，设为d， a[j] = d + 1 时间复杂度为O(NlogN)

<https://leetcode.com/problems/longest-increasing-subsequence/description/>

1. **bruteforce暴力搜索，排列组合，回溯，剪枝**

面试很少问，但考验基本功

全子集问题（组合）：用二进制表示，一个循环搞定

<https://leetcode.com/problems/subsets/description/>

带限制条件的子集：要注意去重，及时剪枝减少搜索规模

<https://leetcode.com/problems/4sum/description/> 此题O(N^3)

<https://leetcode.com/problems/combination-sum/description/>

全排列 不递归：（如果有重复数字怎么解？）

<https://leetcode.com/problems/permutations/description/>

<https://leetcode.com/problems/permutation-sequence/description/>

<https://leetcode.com/problems/next-permutation/> 🡨经典题如何制定交换策略？

假设从小往大全排列，从右边找到第一个非逆序元素a，将其与右侧逆序部分中大于a的最小元素b交换，然后对原逆序部分进行升序排序。

有序排列 去重

经典题：<https://leetcode.com/problems/ugly-number-ii/description/>

N皇后，数独问题：记录暴力搜索状态，每步判断，及时剪枝。

<https://leetcode.com/problems/n-queens/description/>

<https://leetcode.com/problems/valid-sudoku/description/>

广度优先搜索

两个队列同时进行，注意无向图的<https://leetcode.com/problems/clone-graph/description/>

深搜+dp+递归过程中记录节点值

经典矩阵上升序列问题：<https://leetcode.com/problems/longest-increasing-path-in-a-matrix/description/> 首先转化为有向图，再用拓扑排序+dp做。

1. **STL+数据结构Heap Map Set Hash Stack Queue**

Std:: prority\_queue, map, set, multi\_set, multi\_map, pair, unordered\_set, unordered\_map, stack, deque, vector

基本vector string

<https://leetcode.com/problems/longest-substring-without-repeating-characters/description/>

基本Hash

各类N-SUM题用HASH解：<https://leetcode.com/problems/two-sum/description/>

经典题：找不重复数字：<https://leetcode.com/problems/single-number/description/> 思考题中只重复两次的条件如何利用？

经典题：链表复制+指针操作：<https://leetcode.com/problems/copy-list-with-random-pointer/description/>

set map和sort/unique的使用

<https://leetcode.com/problems/group-anagrams/description/>

<https://leetcode.com/problems/largest-number/description/>

重载<符号或者使用函数对象： <https://leetcode.com/problems/merge-intervals/description/> 第一个元素升序 第二个元素降序排列

<https://leetcode.com/problems/largest-number/description/> 如何确定两个字符串排序？

小于号重载（必须在全局）

*template<T>*

*bool operator < (T & a， T & b) {…}*

函数对象

*template<T>*

*class MyLess{*

*public:*

*Bool operator () (T & a, T & b){…}*

*};*

使用 sort(container.begin(), container.end(), MyLess<MyType>());

基数排序，数据分桶

<https://leetcode.com/problems/maximum-gap/description/> O(C \* N)时间内排序，取桶个数为2时，C最差为logN

priority\_queue使用

经典题：topK和topK频率：<https://leetcode.com/problems/top-k-frequent-elements/description/> 思考：如果数组长度导致不能全部读入到内存，如何做？

<https://leetcode.com/problems/find-k-pairs-with-smallest-sums/description/>

stack使用

<https://leetcode.com/problems/simplify-path/description/>

**经典盛水问题**：<https://leetcode.com/problems/trapping-rain-water/submissions/> 用stack记录历史，用water\_level记录当前水位，当梯度为正时，迭代stack中水位以上收集水量

*res += (min(height[i], h.top().second) - water\_level) \* (i - h.top().first - 1);*

*water\_level = min(height[i], h.top().second);*

经典题：逆波兰表达式（与树的遍历有关）<https://leetcode.com/problems/simplify-path/description/>

经典题，计算器（双stack解决）：<https://leetcode.com/problems/basic-calculator/description/>

基本数据结构：树的序列化表示 <https://leetcode.com/problems/verify-preorder-serialization-of-a-binary-tree/description/>

<https://leetcode.com/problems/flatten-nested-list-iterator/description/>

1. **二叉树**

二叉树主要使用递归的思想！（写成循环的话就是stack和queue）

概念：二叉搜索树（BST），平衡二叉树，完全树，字典树，线段树；主要算法：递归，增删改查，深度优先遍历（DFS）广度优先遍历（BFS），先序遍历（PreOrder）中序遍历（InOrder）后序遍历（PostOrder），树的序列化表示和转换等。

递归 +贪心/动态规划

经典题，树的个数（卡特兰数 递推公式理解）：<https://leetcode.com/problems/unique-binary-search-trees/description/>

<https://leetcode.com/problems/unique-binary-search-trees-ii/description/> ！

<https://leetcode.com/problems/different-ways-to-add-parentheses/description/> 运算可以用树表示

经典题，判断树相等：<https://leetcode.com/problems/same-tree/description/>

经典题，判断对称：<https://leetcode.com/problems/symmetric-tree/description/>

<https://leetcode.com/problems/balanced-binary-tree/description/>

<https://leetcode.com/problems/maximum-depth-of-binary-tree/description/>

经典题，最小深度：<https://leetcode.com/problems/minimum-depth-of-binary-tree/description/>

经典题，BST是否成立：<https://leetcode.com/problems/validate-binary-search-tree/description/>

<https://leetcode.com/problems/balanced-binary-tree/description/>

递归+动态规划：<https://leetcode.com/problems/binary-tree-maximum-path-sum/description/>

经典题，树翻转：<https://leetcode.com/problems/invert-binary-tree/description/>

经典题，共同最小祖先：<https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/description/>

广度优先遍历：

<https://leetcode.com/problems/binary-tree-level-order-traversal/description/>

<https://leetcode.com/problems/binary-tree-right-side-view/description/>

先中后序遍历：

<https://leetcode.com/problems/binary-tree-preorder-traversal/description/>

<https://leetcode.com/problems/binary-tree-postorder-traversal/description/>

<https://leetcode.com/problems/binary-tree-inorder-traversal/description/>

序列化与二叉树：

经典题：<https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/>

<https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/description/>（不要求写，理解）

遍历过程中的历史节点记录：

经典题：<https://leetcode.com/problems/flatten-binary-tree-to-linked-list/description/>

<https://leetcode.com/problems/path-sum/description/>

<https://leetcode.com/problems/binary-tree-paths/description/>

BST的增删改查

<https://leetcode.com/problems/delete-node-in-a-bst/description/>

<https://leetcode.com/problems/recover-binary-search-tree/description/>

进阶：红黑树实现

进阶：前缀字典树 Trie

<https://leetcode.com/problems/implement-trie-prefix-tree/description/>

<https://leetcode.com/problems/top-k-frequent-words/description/>

进阶：区间树（线段树）

<https://leetcode.com/problems/count-of-smaller-numbers-after-self/description/>

进阶 Huffman树的实现: <https://blog.csdn.net/google19890102/article/details/54848262>

1. **树/图相关算法，联通区域，拓扑排序，最短路，最小生成树**

图的表示：有/无向图，边的集合，邻接链表，邻接矩阵；图的遍历：深度优先/广度优先，找环；最短路：单点最短路Dijstra（优先队列实现）, 任意两点最短路Floyd（动态规划实现）；并查集。

拓扑排序：

经典排课问题：<https://leetcode.com/problems/course-schedule-ii/description/> 这里注意维护两个表：<课程编号，依赖课程数量> , <课程编号，后序课程编号的vector>，不断循环找依赖课程数量为0的节点即可，整体时间复杂度为O(V+E)，注意完全独立的课程不要遗漏。

图的最短路：

任意两点最短路：<https://leetcode.com/problems/minimum-height-trees/description>

并查集：数组a[i]表示i节点的父节点下标，若a[i] == i则表示已经是root节点；b[i]表示类别i的个数；查询类别：找到root为止，同时带路径压缩；判断i，j是否相等时，找到各自的root节点为止，同时启用路径压缩：将到root路径的所有节点都指向root下标；合并i，j集合时候，不妨令类别i的个数小于j的个数：将i的root节点指向j的root节点即可；由于合并时候都是节点数少的一半改变父节点，假设最后只合并到一个类，那么总的合并时间复杂度为O(n)，查询的时间复杂度平均为O（1）

图的联通区域：<https://leetcode.com/problems/surrounded-regions/description/> (并查集实现？)

经典题：最大相邻序列：<https://leetcode.com/problems/longest-consecutive-sequence/description/>

经典题：并查集个数：<https://leetcode.com/problems/friend-circles/description/>

1. **贪心/动规**

贪心，一般比较难想到

两个指针类型题：<https://leetcode.com/problems/container-with-most-water/description/> 两边往中间靠

经典题：<https://leetcode.com/problems/majority-element/description/> 维护集合性质不变

<https://leetcode.com/problems/gas-station/description/>

已经知道节点j开始绕一圈过程中最多缺多少油，则j-1节点会缺多少油？

<https://leetcode.com/problems/candy/description/> 找到连续向下的坡长，记录左右两边最大的坡长即可

动态规划

多画图，多判断，从而推得递推公式；注意初始条件，特殊条件判断；使用滚动数组减少内存开销；如果需要给出最佳结果的路基，需要额外数组记录每步决策的顺序

一维动态规划，定义a[i]到i为止最佳结果，视i+1情况讨论得到a[i+1]

经典题: <https://leetcode.com/problems/maximum-subarray/description/>

滚动数组使用<https://leetcode.com/problems/triangle/description/>

经典题：最长上升子序列<https://leetcode.com/problems/longest-increasing-subsequence/>

记录多个状态：<https://leetcode.com/problems/longest-valid-parentheses/description/>

a[i]记录上一个未匹配的左括号的位置，b[i]记录以i个括号为结尾的最佳结果。

1. a[0] = -1 表示没有未匹配的左括号
2. S[i]为右括号，b[i] = 2 + b[i-1] + b[a[i] - 1]; b[i-1]表示已有匹配数，2表示当前右括号匹配到的a[i]位置左括号,b[a[i] - 1]表示因为a[i]位置左括号被匹配，可以再往前叠加匹配数量；a[i+1] = a[a[i]]
3. S[i]为左括号b[i] = 0; a[i+1] = i

<https://leetcode.com/problems/decode-ways/description/> 类似最大子序列和的递归方式，注意讨论’0’字符的情况

二维动态规划，定义a[i,j]为最佳结果，由a[:, j-1]/a[i-1, j-1]/a[i-1, j]….等递推得到（LCS）

CRF/HMM中计算partition function的BW算法（前向后向算法）和Viterbi等；

二维经典：<https://leetcode.com/problems/minimum-path-sum/description/>

字符串DP：LCS，编辑距离

<https://leetcode.com/problems/edit-distance/>

<https://leetcode.com/problems/regular-expression-matching/> 对LCS规则稍加改动

<https://leetcode.com/problems/distinct-subsequences/description/>

改动LCS的规则，注意初始化

<https://leetcode.com/problems/maximal-square/> 注意正方形的条件怎么用于DP?

1. **字符串相关**

（部分在动态规划中）

最长不重复子串：<https://leetcode.com/problems/longest-substring-without-repeating-characters/description/> O(N^2)解决是naïve的， 若字典是固定的，如何在O(N)时间内解决？