

654. 最大二叉树

- ① 找到给定数组的最大值作为根节点
- ② 该数左边的子数组构成左子树，右边的子数组构成右子树

时: $O(n)$

空: $O(\log n)$

14丁. 检查单词是否为句子中其他单词的前缀

- ① 将单词分隔开
- ② 用 searchWord 逐个对比，是否为其他单词的前缀

时: $\mathcal{O}(n)$

空: $\mathcal{O}(m)$

2164. 对奇偶下标分别排序

- ① 将奇偶下标元素分离
- ② 对奇下标数组逆序排序，对偶下标顺序排序
- ③ 按序将两数组合并

时: $O(n \log n)$

空: $O(1)$

1460. 通过翻转子数组使两个数组相等

① 两个哈希表统计 arr 和 target 的元素及出现次数

② 若两哈希表相等则必须通过翻转使两数组相等，否则
则返回 False

时: $O(n+n)$

空: $O(n+n)$

658. 找到 k 个最接近的元素

- ① 先用二分查找找到最靠近 x 的数 mid
- ② 再以最靠近的数往数组两边找次靠近 x 的数
- ③ 在 mid 左边则从 res 左侧添加，在 mid 右边则从 res 右侧添加
- ④ 若左右的数都一样靠近 x 的话，优先添加左侧的数

时: $O(\log n + k)$

空: $O(k)$

1464. 数组中两元素的最大乘积

① 数组排序

② 排数组最后两个数相乘

时: $O(n \log n)$

空: $O(1)$

793. 阶乘函数后 k 个零

- ① 每隔 5 个数会多一个零，每隔 5^n 会多几个零
- ② 采用二分法找出出现 k 个零的最小那个数 low
- ③ 在 $(low, low + 6)$ 之间最后确定出现 k 个零的个数

时: $O(\log^k n)$ 空: $O(1)$

66). 二叉树最大宽度

- ① 采用 bfs , 初始化 $arr = [(\text{root}, 1)]$
- ② 每次弹出首元素 , 若 tmp 有左孩子 $arr.append((\text{tmp.left}, \text{index} \times 2))$
有右孩子 $arr.append((\text{tmp.right}, \text{index} \times 2 + 1))$
- ③ 每一层算一次 $res = \max(res, arr[-1][1] - arr[0][1] + 1)$

时: $O(n)$

空: $O(\log n)$

1470. 重新排列数组

i 从 $(0, n)$, nums 的 $2 \times i + 1$ 的位置插入 $\text{nums}[n-i]$ 元素

时: $O(n)$

空: $O(1)$

998 · 最大二叉树 II

题意：本身最大树是按一定规则构造出来的。现在在原本的 nums 后加一个 val 值，用同样的构造方法构造最大树

有两种情况：
① val 值是 nums 中的最大值，将 val 节点作为根节点， root 挂在 val 节点的左子树

② 否则，在 root 的右子树中找到比 val 值小的节点，将此节点作为 val 节点的左子树

③ 找不到则将 val 节点添加到右子树叶节点的左子树

时： $O(\log n)$

空： $O(1)$

946. 验证栈序列

模拟入栈出栈的过程，若序列对得上则 True，否则 False

时: $O(n)$

空: $O(n)$

1475. 商品折扣后的最终价格

逆序遍历 prices 数组，维护单调栈 stack

① 若 stack 为空，添加 $\text{prices}[i]$

② 否则判断 $\text{stack}[-1] \leq \text{prices}[i]$ 的大小，若 $\text{stack}[-1] \leq \text{prices}[i]$ ，则
stack 添加 $\text{prices}[i]$ 且 $\text{prices}[i] = \text{stack}[-2]$

③ 否则 stack.pop() ，继续 ① ② 操作

时: $O(n)$

空: $O(n)$

687. 最长同值路径

采用后序遍历的方式，统计

① 左右子树同值路径的长度，以及其总和
②

② 返回只选择左右子树中同值路径较长的

记录 ①、② 中最长的

时： $O(n)$

空： $O(\log n)$

646. 最长数对链

- ① 根据数对的第二个数进行排序 ($\text{key} = \lambda x : x[1]$)
- ② 快慢指针，若快指针的第一个数不大于慢指针第二个数， pass

若否，慢指针 = 快指针。

- ③ 快指针 + 1

注：只有一个数对的情况也构成数对链

时： $O(n \log n + n)$

空： $O(n \log n)$

1582. 二进制矩阵中的特殊位置

- ① 统计每行、列的和
- ② 遍历矩阵，当遇到 (i, j) 个元素值为 1 时，若 i 行和 j 列的和均为 1，则为特殊位置

时： $\mathcal{O}(mn)$

空： $\mathcal{O}(m+n)$

6.52. 寻找重复的子树

① 后序遍历整棵树

② 将每个结点作为根节点的子树序列化表示出来，用哈希表统计这些序列

③ 若在统计过程中遇到一样的序列，则找到重复的子树

时: $O(n)$

空: $O(mn)$

828. 统计字符串中唯一的字符

- ① 用哈希表统计字符串中每个字符的出现的下标
- ② 对于某个字符串中的字符，其对结果的贡献为

$$(cur - pre) \times (nxt - cur)$$

③ 初始化 $pre = cur = nxt = -1$;

迭代中 $pre = cur$

$cur = v.pop(0)$

$nxt = v[0]$ if v else n

时: $O(n+n)$

空: $O(n)$

1592. 重新排列单词间的空格

- ① 用快慢指针将单词和空格数统计出来
- ② 计算出单词间空格数 以及 末尾空格数
- ③ 按② 的结果将字符串连成串

时: $O(n)$

空: $O(n)$

667. 优美的排列 II

① $\text{res} = [1, 2, \dots, n]$

② 若 $k=1$ ，直接返回 res

③ 否则， $\text{res}[k-1] = \text{res}[0]$ ，然后逆推， $\text{res}[k-2] = \text{res}[k-1] + (k-1)$

$\text{res}[k-3] = \text{res}[k-2] - (k-2)$ 这样一加一减直到 $[k-k]$

时: $O(n)$

空: $O(1)$

6.9. 修剪二叉搜索树

① 先序遍历对每个节点进行操作

② 若到某个节点值小于 low，则该节点成为其右子节点，达到修剪左子树；反之，大于 high 则成为其左子节点。

时: $\Theta(n)$

空: $\Omega(\log n)$

670. 最大交换

从头到尾对于每个数，判断后面是否有比它大的数，是的话交换二者数位，若有个相同的比它大的数，交换索引大的那个

时: $O(n^2)$

空: $O(n)$

1608. 特殊数组的特征值

- ① 将数组倒序排序
- ② 从 1 - n 遍历数组索引，又从 1 开始计算
- ③ 遍历判断 $\text{nums}[i-1]$ 与 x 大小，若 $\text{nums}[i-1] \neq \text{nums}[i]$ > x，则计 x+1；若 $\text{nums}[i-1] > x$, $\text{nums}[i] < x$. 或 $x = n$ ，则 x 为特征值。否则 没有特征值，返回 -1

时: $O(n \log n)$

空: $O(\log n)$

1619. 删 除 某 些 元 素 后 数 组 均 值

- ① 遍历一遍数组，用堆记录下最大和最小的 5% 并统计数组的和
- ② 总和减去最大和最小的 5% 后求均值

时: $O(n)$

空: $O(c \log n)$

672. 灯泡开关 II

- ① 若 $\text{presses} = 0$. 只有 1 种状态 , 否则
- ② 若 $n=1$, 有 2 种状态
- ③ 若 $n=2$, 当 $\text{presses} = 1$ 时, 有 3 种状态 ; 否则有 4 种状态.
- ④ 若 $n \geq 3$, 当 $\text{presses} = 1$ 时, 有 3 种状态 ; 否则有 4 种状态 .

时: $O(n)$

空: $O(n)$

82]. 最大人工岛

- ① 遍历一遍矩阵，如遇 $\text{grid}[i][j] = 1$ ，采用 dfs 处理
- ② dfs：遍历上下左右，将岛屿标上唯一的记号，并记录岛屿大小
- ③ 再遍历一遍矩阵，如遇 $\text{grid}[i][j] = 0$ ，则假设其变为岛屿后与上下左右的岛屿组成新大岛屿，并记录下最大的面积

时： $O(n^2)$

空： $O(n^2)$

1624. 两个相同字符之间的最长子字符串

用哈希表记录下字符第一次出现的下标，后若再遇相同的字符则
作减法，记录下标差距的最大值

时: $O(n)$

空: $O(n)$

1636. 按照频率将数组升序排序

- ① 遍历一遍数组，用哈希表统计每个元素出现次数
- ② 先按哈希表中键值降序排序（得到元组的元素的列表）
- ③ 再按频率升序排序
- ④ 最后整理出答案输出即可

时： $O(2n \log n + 2n)$ 空： $O(n)$

698. 划分为 k 个相等的子集

① 排序，算出每个子集的总和，若 $\text{nums}[i-1] > \text{pair}$ 或 pair 不为整，False

② 记忆化搜索：① cache

能凑成 pair 的尽可能的凑，当完全分成 k 个等子集 True

否则 False

时： $O(n^k)$

空： $O(n)$

854. 相似度为 K 的字符串

采用 bfs，初始化 $q = [(s1, 0)]$ ， $vis = \{s1\}$

遍历 q 中字符串若 $s1[i] \neq s2[i]$ ，则在 $[i+1, n)$ 内选择 $s1[j] = s2[i] \neq s2[j]$ 的字符交换，交换后若未在 vis 中则 $q.append$ 以及加入 vis 。

1640. 能否连接形成数组

- ① 字典记录下 arr 的每个数的下标
- ② 对 pieces 的每个组合，用切片的方式比对 arr 中有无 pieces 组合。

时: $O(2n)$ 空: $O(n)$

1652. 手炸弹

判断 k.

- ① 若 $k < 0$, 则 对 每个元素 赋上 $j \in [1, k+1] (i+n-j) \% n$ 的总和
- ② 若 $k = 0$, 则 每个元素 都 赋上 0
- ③ 若 $k > 0$, 则 每个元素 都 赋上 $j \in [1, k+1] (i+j) \% n$ 的总和.

时: $O(n^2)$

空: $O(n)$

788. 旋转数字

要求：
① 数位上不能出现 “3” “7” “4”
② 数位上必须要有至少一个“2” “5” “6” “9”

对 $1 \sim n$ 每个数都进行上述的判断。

时: $O(n \log n)$

空: $O(1)$

面试题 17.19. 消失的两个数字

- ① 算出原本数字的和 total sum 和平方和 total Square
- ② 算出消失的数字和 miss sum 和消失的平方和 miss Square

解方程

$$\frac{\text{missum} \pm \sqrt{2 \times \text{missSquare} - \text{missum}^2}}{2}$$

时: $O(n)$

空: $O(1)$

面试题 17.09 . 第 k 个数

用最小堆和哈希表唯一地记录下每个从堆中弹出的数的 3 倍，5 倍
7 倍

时: $O(n \log n)$

空: $O(n)$

剑指 offer II . 040. 矩阵中最大的矩形

- ① 统计每层的高度 $heights$
- ② 采用单调栈思想求出该层最大面积

while $stack[-1][1] > h$:

- , $oh = stack.pop()$

$ans = \max(ans, (i - stack[-1][0] - 1) \times oh)$

时: $O(mn^2)$

空: $O(n)$

面试题 01.09 . 字符串旋转

判断 s_2 是否由 s_1 旋转而来， $s_1 \times 2$ ，然后在 s_1 中找 s_2

时: $O(n)$

空: $O(n)$

面试题 01.08. 零矩阵

创建row collum矩阵记录行列的零化

遍历matrix，如遇 $\text{matrix}[i][j] == 0$, 则将 $\text{row}[i]$, $\text{collum}[j]$ 置为True

最后根据row collum情况将对应行列零化

时 : $O(mn)$

空 : $O(n)$

1784. 检查二进制字符串字段

遍历字符串，看有多少段由‘1’组成的字符串字段

时： $O(n)$

空： $O(1)$

9.21. 使括号有效的最少添加

用 left 记录已有左括号数目，right 表示缺右括号的数目

遍历字符串。若 ch 为左括号，left + 1

若 ch 为右括号，且 left > 0，则 left - 1

若 left = 0，则 right + 1

最终返回 left + right

时: $O(n)$

空: $O(1)$

8.11. 子域名的访问次数

- ① 用哈希表统计每个子域名的访问次数
- ② 再按规定格式将次数与子域名连接在一起

时: $O(mn)$

空: $O(mn)$

927. 三等分

1. 先统计数组里1的个数cnt，若不能三等分直接返回[-1,-1]
2. part = cnt / 3，找到每组开始的1的下标a，b，c，再接着向后遍历，判断是否每个都一致，全部一致则能返回[a-1,b]

时： $O(n)$

空： $O(1)$

1800. 最大升序子数组和

从头遍历数组，若 $\text{nums}[i] > \text{nums}[i-1]$ ， $\text{res}[i] += \text{res}[i-1]$ ，最后返回res中的最大值

时： $O(n)$

空： $O(n)$

870. 优势洗牌

- ① 按数组元素大小将 nums1 和 nums2 的下标排序，得 $\text{id}x1$ 和 $\text{id}x2$
- ② i 从 0 到 $n-1$ ， $\text{left} = 0$ ， $\text{right} = n-1$
若 $\text{nums1}[\text{id}x1[i]] > \text{nums2}[\text{id}x2[i]]$ ，添加到 $\text{ans}[\text{left}]$ ， $\text{left} + 1$
否则，添加到 $\text{ans}[\text{right}]$ ， $\text{right} - 1$

时: $O(n \log n)$

空: $O(n)$

856. 括号的分数

① 定义一个栈 $stack = [0]$

② 遍历字符串，若遇 “(”， $stack.append(0)$

若遇 “)” , $v = stack.pop()$

$stack[-1] += \max(2v, 1)$

时: $O(n)$

空: $O(n)$

801. 使序列递增的最小交换次数

dp : 初始化 not-swap = 0 , swap = 1

对于每个下标位置，若有 $\text{nums1}[i-1] < \text{nums1}[i]$ 且 $\text{nums2}[i-1] < \text{nums2}[i]$

说明可换可不换 ns, s = not-swap, swap + 1

接着 若有 $\text{nums1}[i-1] < \text{nums2}[i]$ 且 $\text{nums2}[i-1] < \text{nums1}[i]$

说明可交换其中一对 ns, s = min(ns, swap), min(s, notswap + 1)

最后返回 $\min(\text{not-swap}, \text{swap})$

时: $O(n)$

空: $O(1)$

1790. 仅执行一次字符串交换能否使两个字符串相等

- ① hashmap1、hashmap2 分别记录 s1 和 s2 字符出现次数
- ② cnt 记录 s1 和 s2 对应字符不同的个数
- ③ 只有 $\text{hashmap1} : \text{hashmap2}$ 且 $\text{cnt} = 0$ 或 2 才为 True

时: $O(n)$

空: $O(n)$

817. 链表组件

1 先用哈希表记录下num的每个值

2 从头遍历链表，每有一个组件res就+1

时： $O(n)$

空： $O(n)$

769. 最多能完成排序的块

初始化 $BE = 1$, $res = 0$

从头遍历 arr 数组, 若 $i > BE$, $res + 1$ 且 $BE = arr[i]$

若 $i \leq BE$ - $BE = \max(BE, arr[i])$

时: $O(n)$

空: $O(1)$

940. 不同的子序列 II

初始化 $g = [0] \times 26$, $\text{mod} = 10^9 + 7$

枚举 s 中每个字符:

$\text{total} = (1 + \text{sum}(g)) \% \text{mod}$ 该字符之前的子序列数 + 1

$g[\text{ord}(ch) - \text{ord}('a')] = \text{total}$ 赋给该字符

时: $O(n)$

空: $O(26)$

1441. 用栈操作构建数组

初始化 pos 指向 target 的下标

i从 1 ~ n+1 遍历，对于每个数都先 res.append("Push")

若 $i \neq \text{target}$ ，res.append("Pop")

否则 证明 i 是数组的元素，保留， $pos + 1$

时： $O(n)$

空： $O(1)$

886. 可能的二分法

① 用字典将所有不喜欢的组合记录下来

② 从 $1 \sim n$ 开始遍历，采用dfs思路

$\text{color}[i] = c$ ，将所有 i 不喜欢的染成 $3 - c$

若有冲突的则说明不能二分

时: $O(n+m)$

空: $O(m+n)$

904. 水果成篮

滑动窗口

当窗口内水果种类 ≤ 2 时，窗口继续扩大

否则窗口右移

最后返回窗口大小。

时: $O(n)$

空: $O(n)$

1700. 无法吃午餐 的学生数量

① 用哈希表统计喜欢圆形和方形的数量

② 从头遍历三明治，若当前三明治没人爱吃，则停止

时: $O(n)$

空: $O(1)$

719. 第 k 个语法符号

递归：若 $n=1$ 时，返回 \circ

否则， $\text{return } \text{self}.k\text{th Grammar}(n-1, \frac{k-1}{2}+1)^{\wedge}(\frac{k-1}{2})$

时： $O(n)$

字： $O(n)$

S01. 股票价格跨度

单调栈，初始化 $\text{stack}[-1, \inf]$] , idx

对于每个元素，当 $\text{price} \geq \text{stack}[-1][1]$ 时， stack.pop()
直到条件不成立， $\text{stack.append}([idx, price])$

return $\text{idx} - \text{self.stack}[-2][0]$

时: $O(n)$

空: $O(n)$

1235. 规划兼职工作

① 按工作结束时间将工作排序

② 初始化 $dp = [0]$, $pre = [0]$

③ 遍历排序后的工作，采用二分法在 pre 上找比 $start[i]$ 早结束的工作 j ，若 $dp[j] + profit[i] > dp[i-1]$ ，则选第 i 个工作

时: $O(n \log n)$

空: $O(n)$

1768. 交替合并字符串

从头遍历 word_1 和 word_2 ，并逐一添加至 res 中，当其中一个字符串遍历结束，则将另一个字符串剩余的字符添加至 res 尾部

时： $O(n)$

空： $O(n)$

9.15. 分割数组

① 初始化 stack = []

② 倒序遍历数组 nums，若 stack 为 \emptyset 或 $stack[-1] > nums[i]$ ，则将 $nums[i]$ 添加至 stack

③ $leftMax = nums[0]$

④ 倾序遍历 nums，若 $leftMax > stack[-1]$ ，说明不符合左乙右条件，
若 $nums[i] > leftMax$ ，更新 $leftMax$ ；直至 $leftMax \leq stack[-1]$

时: $O(n)$

空: $O(n)$

862. 和至少为 k 的最短子数组

① presum 数组统计前缀和

② 初始化两端队列 q

③ 枚举 presum，若出现 $s[i] - s[j] \geq k$ ($j < i$)，那么往左不可能有比 i 更短的子数组。弹出 j ；
若出现 $s[j] \geq s[i]$ ($j < i$)，往后若 $s[i+x] - s[j]$ 满足条件，则 $s[i+x] - s[i]$ 肯定也满足条件。弹出 j

时: $O(n)$

空: $O(n)$

934. 最短的桥

- ① 遍历数组，当出现第一个 "1" 则采用 bfs 找出第一个岛屿
- ② 采用 bfs 每次将岛屿扩张一格，直至与第二个岛屿接壤

时: $O(n^2)$

空: $O(n^2)$

1822. 数组元素积的符号

① 初始化 $res = 1$

② 遍历数组，对于每个元素都进行 $res \times num[i]$

③ 若 $res = 0$ ，直接返回 0

若 $res > 0$ ，将 $res = 1$

否则 将 $res = -1$

时： $O(n)$

空： $O(1)$

90). + 数组的最小值之和 .

① 单调栈确定某个值的影响范围 (找左右边界)

② 遍历数组 $ans += (i - left[i]) \times (right[i] - i) \times arr[i]$

时: $O(n)$

空: $O(n)$

481. 神奇字符串

① 若 $n \leq 3$ ，此时字符串为 "122"，直接返回。

② 否则，双指针 $i = 1, j = 2$

当 $s[i] = 1$ ，则在 j 后添 1 个字符；

$s[i] = 2$ 则在 j 后添 2 个字符；

当 $s[j] = 1$ 时，则此次添 2，反之添 1

时： $O(n)$

空： $O(n)$

1662 . 检查两个字符串数组是否相等

将两数组的字符串连起来作比较

时: $O(n)$

空: $O(n)$

1620. 网络信号最好的坐标

- ① 因为信号塔的范围在 $[0, 50]$ ，所以坐标 x, y 限制在 $[0, 50]$
- ② 从 $(0, 0)$ 到 $(50, 50)$ 依次计算每个坐标的信号，只有当前信号大于先前统计的才更新

时: $O(n^2)$

空: $O(1)$

1668. 最大重复子字符串

初始化 $nword = word$

若 $nword$ 能在 sequence 中找到，即 $nword$ 仍是子串。

则 $nword += word$ ，直到在 sequence 中找不到 $nword$ 。

时： $O(n)$

空： $O(n)$

754. 到达终点数字

① target 为正为负结果不变

② 初始化 $k = 1$

③ 循环执行 $target -= k$, $k += 1$ 直到 $target \% 2 = 0$

(1) 当 $target = 0$, 直接返回 k

(2) 当 $target$ 为偶数, 将 $\ln k - 1$ 步其中一步倒转即可

(3) 当 $target$ 为奇数, $k + 1$ 步或 $k + 2$ 步必能使 $target$ 变为偶数

时: $O(\log n)$

空: $O(1)$

11.06. 解析布尔表达式

- ① 从头遍历表达式，用栈分别存操作符和布尔值
- ② 在得到一个完整表达式时，先运算得出布尔值再进行下去
 - (1) 若操作符为：“!”，直接对布尔值取反
 - (2) 若操作符为：“|”，若布尔值中有“t”则“t”，否则“f”
 - (3) 若操作符为：“&”，若布尔值中有“f”则“f”，否则“t”

时间： $O(n)$

空间： $O(n)$

1678. 设计 Goal 解析器

双指针遍历，将 command 中 “G” 变成 ‘G’，“()” 变成 ‘o’，

“(a)” 变成 ‘a’

时: $O(n)$

空: $O(n)$

816. 模糊坐标

直接模拟构造 所有合理的坐标

① 若一个字符串首不为“0”，可直接以整数添加

② 不可添加小数点的情况，串尾为：“0”，串首为：“0”且下标不为1

时: $O(n^2)$

空: $O(n)$

764. 最大加号标志

dp : 对每个元素判断上下左右最大连续个数，并从上下左右中选连续个数最少的作为该元素的 score

最后再从所有元素的值中挑选最大的

时: $\mathcal{O}(n^2)$

空: $\mathcal{O}(n^2)$

1684. 统计一致字符串的数目

- ① 用哈希表将 allowed 串中每个字符记录下来
- ② 再遍历 words 列表中每个 word 的每个字符，看是否都在哈希表中

时: $O(mn)$

空: $O(1)$

1704. 判断字符串的两半是否相似

统计前一半的元素数目与后一半的是否相同

时: $O(n)$

空: $O(1)$

791. 自定义字符串排序

① 哈希表 1 记录 order 的字符

② 遍历 s，用哈希表 2 记录在哈希表 1 中的字符数目。
不在哈希表 1 中的字符添加到 res 中

③ 遍历 order，若字符出现在哈希表 2 中则添加对应数目
的字符到 res 中

时: $O(\max(m, n))$

空: $O: (\max(m, n))$

805. 数组的均值分割

① 对每个数 $\text{nums}[i] = \text{nums}[i] \times n - s$

② 将数组分成两半

① 在左半部分找有无子数组和为 0

② 在右半部分有无子数组和为 0

③ 两边有无子数组和为相反数

时: $O(n \cdot 2^{n/2})$ 空: $O(2^{n/2})$

17.10. 卡车上的最大单元数

- ① 先将 boxTypes 按单元数量大小排序
- ② 按序将箱子装上卡车，直到 $\text{truckSize} = 0$ 或 boxTypes 为空。

时: $O(n \log n)$

空: $O(n \log n)$

775. 全局倒置和局部倒置

- ① 初始化 $before = -1$ 用于记录最大已遍历值
- ② 从 $0 \sim n-1$ 遍历，若 $before > \text{nums}[i+1]$ 则返回 False
- ③ $before = \max(\text{before}, \text{nums}[i])$

时: $O(n)$

空: $O(1)$

792. 匹配子序列的单词数

- ① 用列表嵌套字典记录 s 中每个字符 的 后缀 以及下标
- ② 遍历 $words$ 的 每个字符，在 ① 中找后缀，若 找不到后缀
说 明 不是 s 的 子序列

时: $O(n)$

空: $O(nm)$

891. 子序列宽度之和

- ① 元素顺序不影响最终结果，`nums.sort()`
- ② 每个元素对结果的贡献

以该元素为最大
值的序列数

$$(2^i - 2^{n-1-i}) \times \text{nums}[i]$$

以该元素为最小
值的序列数

时: $\mathcal{O}(n \log n)$

空: $\mathcal{O}(n)$

1732 · 找到最高海拔

① 初始化 $cur = res = 0$

② 遍历 gain， $cur += gain[i]$ ，并 $res = \max(res, cur)$

时: $O(n)$

空: $O(1)$

799. 香槟塔

- ① 先假设将 poured 的酒全部装在最顶上的杯子
- ② 一层层往下模拟溢出状态，若某个杯子的内容大于 1 时。
则向杯子两边平均溢出
- ③ 直到找到 query-row 行的 query-glass 个杯子。

时: $O(100^2)$

空: $O(100)$

808. 分汤

① 四种操作都是 25 的倍数， $n = \lceil n / 25 \rceil$

② 当 $n > 193$ 时，A 花完 + A+B 同时分完概率 $/2$ 已经无限接近 1

③ $dp[i][j] = dp[\max(0, i-4)][j] + dp[\max(0, i-3)][j-1] +$

$$\underbrace{dp[\max(0, i-2)][\max(0, j-2)] + dp[i-1][\max(0, j-3)]}_4$$

时： $\mathcal{O}(193^2)$

空： $\mathcal{O}(193^2)$

1742. 盒子中小球的最大数量

- ① 将每个小球的编号 转为 盒子 编号，并用 哈希表 统计 盒子 编号 的 数量
- ② 从 哈希表 中 返回 最大的 盒子 编号 的 数量

时: $O(n)$

空: $O(n)$

795. 区间子数组个数

- ① 遍历 nums 数组，并维护 last1 ， last2 两个变量
- ② 若 $\text{left} \leq \text{nums}[i] \leq \text{right}$ ， $\text{last1} = i$
若 $\text{nums}[i] > \text{right}$ ， $\text{last1} = -1$ ， $\text{last2} = i$
- ③ 若 $\text{last1} \neq -1$ ， $\text{res} += (\text{last1} - \text{last2})$

时: $O(n)$

空: $O(1)$

809. 情感丰富的文字

对每个 words 的单词 采用双指针与 s 作比较

curi, nxti 遍历 s , curj, nxtj 遍历 word

① 若 $s[curi] \neq word[curj]$ \rightarrow False

② 若 同一个字母 , $nxti - curi \neq nxtj - curj$ 且 < 2
 \rightarrow False

时: $O(mn)$

空: $O(n)$

1752. 检查数组是否经排序和轮转得到

① $\text{nums} = \text{nums} + \text{nums}$

② 遍历 nums ，若 $\text{nums}[i] > \text{nums}[i-1]$ 则 $\text{cnt} += 1$

当 $\text{cnt} = \text{len}(\text{nums}) / 2$ 时 True

否则 False

时: $O(n)$

空: $O(n)$

813. 最大平均值和的分组

① 平均值和最大分组的子数组数目是 k

转移方程

$$dp[i][j] = \begin{cases} \text{sum(nums)} / i & , j = 1 \\ \max \left\{ dp[i-x][j-1] + \sum_{r=x}^{i-1} \text{nums}[r] / (i-x), j > 1 \right\} & \end{cases}$$

时间: $O(k \cdot n^2)$

空间: $O(n)$

17.8. 生成交替二进制字符串的最少操作数

形成交替字符串最终只有2种格式 “0101…”，“1010…”

① 统计形式 “0101…” 需要的操作数

② 统计形式 “1010…” 需要的操作数

返回较小的

时: $O(n)$

空: $O(1)$

1779. 找到最近的有相同 X 或 Y 坐标的点

遍历 points ，若有 $\text{points}[i][0] = X$ 或 $\text{points}[i][1] = Y$ 。

则记录该点与 (X, Y) 的曼哈顿距离，返回曼哈顿距离最小点的下标。

时: $O(n)$

空: $O(1)$

1769. 移动所有球到每个盒子所需要的最小操作数

① 初始化 $\text{cnt} = 0$, $\text{ans}[] = [0] \times n$

② 分别从左 遍历， 每遍历一步， $\text{ans}[\text{i}] = \text{ans}[\text{i}-1] + \text{cnt}$,

且若 $\text{boxes}[\text{i}] = "1"$ 时， $\text{cnt} += 1$ ，则可得到只往右移，将所有球移到每个盒子 的操作数。

③ 从右到左遍历 亦然

时： $O(n)$

空： $O(n)$

1796. 字符串中的第二大的数字

初始化 $\text{res} = [-1, -1]$

遍历字符串，若出现数字且不在 res 中且不比 res 中数字小，

则替换 res 中较小的

最后返回 $\min(\text{res})$

时: $O(n)$

空: $O(1)$

1774. 最接近目标价格的甜点成本

④ 深溯法将所有可能的成本列出，返回最接近目标价格的

时: $O(n \cdot 3^m)$ 空: $O(n)$

1805.字符串中不同整数的数目

- 1 字符串word末尾先添加一个非数字字符，确保字符串最后一个字符不是数字
- 2 双指针获取字符串中的每一串连续的数字子串，并去掉前导0
- 3 用哈希表统计获取到的字符串，确保统计的整数是唯一的

时： $O(n)$

空： $O(n)$

1775. 通过最少操作数使数组的和相等

- ① 先求两数组的和 sum1, sum2
- ② 确保数组和大的为 nums1, 小的为 nums2
- ③ nums1, nums2 排序
- ④ nums1 从尾到头遍历, nums2 从头到尾遍历
- ⑤ 先操作改变大的数
- ⑥ 若遍历到操作不变的时候, 就说明不可能相等

时: $O(n \log n + n)$

空: $O(n \log n)$

1812. 判断国际象棋棋盘中一个格子的颜色

- ① 将字符串中第一个字符的 ASCII 码与 'a' 作差值 + | + 第二个字符
- ② 若为偶数则黑，奇数则白

时: $O(1)$

空: $O(1)$

1780. 判断一个数是可以表示成三的幂的和

- ① 将 $3^0 \sim 3^{15}$ 记录在一个数组中 (members)
- ② 倒序遍历 members，当 $\text{members}[i] \leq n$ 时，则进行③
- ③ $n = n - \text{members}[i]$
- ④ 当遍历结束后，若 $n = 0$ ，则 True，否则 False

时间: $O(1)$

空间: $O(1)$

1691.堆叠长方体的最大高度

①先将每个长方体的长宽高按大小排序

②将所有长方体按最长边顺序排序

③动态规划+贪心：

对于每次 $dp[i]$ ，遍历 j 属于 $[0,i)$ ， $dp[i] = \max(dp[i], dp[j] + co[i][2])$

时： $O(n^2)$

空： $O(n)$

1781. 所有子字符串美丽值之和

- ① 遍历一遍子字符串，每次遍历统计从 i 开始所有子串美丽值
- ② 哈希表记录子串频率，再算出美丽值

时: $O(n^2)$

空: $O(26)$

1832. 判断句子是否为全字母句

① 初始化 $a = [0] \times 26$

② 遍历句子，将 $a[ord(ch) - ord('a')] = 1$

③ 最后求 a 的和

时: $O(n)$

空: $O(26)$

1697. 检查边长度限制的路径是否存在

并查集

- ① 所有查询按 limit 排序，所有边按边权排序
- ② 对于每个查询，从边权最小的边开始，将小于 limit 的边加入并查集
- ③ 最后查询两点是否连通即可

$$时间: O(m + \log m + q + \log q)$$

$$空间: O(\max(m, q))$$

1945. 字符串转化后的各位数字之和

① 将字符串逐个转化为字符串数字

② 将字符串数字逐位相加求和

时: $O(k \cdot n)$

空: $O(n)$

1785. 构成特定和需要添加的最少元素.

- ① 数组求和 cur
- ② 计算 cur 与 特定和 的差距 gap
- ③ 若 $gap \div limit$ 整除 则 返回 , 否则 $gap \div limit + 1$

时: $O(n)$

空: $O(1)$

1764. 通过连接另一个数组的子数组得到一个数组

① 按序遍历 groups，从头一一比对 nums 中是唯一且不相交地
出现 groups 中所有子数组

时: $O(n)$

空: $O(n)$

1971. 寻找图中是否存在路径

- ① 用哈希表记录每个顶点相邻的顶点.
- ② 广度优先遍历寻找从源到目的的路径

时: $O(n)$

空: $O(n)$

1760. 袋子里最少数目的球

题目可转换为：在 maxOperations 次操作后每个袋中球能否少于 x ？

二分法： $l = 1, r = \max(\text{nums}), \text{mid} = \frac{l+r}{2}$

每次的操作数 $\text{ops} = \sum ((x-1) // \text{mid} \text{ for } x \text{ in } \text{nums})$

时： $O(\log n)$

空： $O(1)$

1753. 移除石子的最大得分

每次从三堆石子中较大的两堆中取即可
采用堆排序维持石堆大小顺序

时: $O(n \log n)$

空: $O(1)$

2011. 执行操作后的变量值

顺着operations的顺序对变量进行操作，有‘+’则变量+1，否则变量-1，返回最终的变量值。

时： $O(n)$

空： $O(1)$

1754. 构造字典序最大的合并字符串

不断对比 word₁ 与 word₂ 的大小，将大的首字符添加到 res 末尾。

时: $O(n^2)$

空: $O(n)$

1739. 放置盒子

接触地面的盒子数 $i * (i+1) / 2$

盒子上限数 $i * (i+1) * (i+2) / 6$

在此基础上增加1个与地面接触的盒子需要增加盒子数 $j * (j+1) / 2$

计算让 $i * (i+1) * (i+2) / 6 \leq n$ 的最大 i ，以及 $i * (i+1) * (i+2) / 6 + j * (j+1) / 2 \geq n$ 的最 j

时： $O(\sqrt{n})$

空： $O(1)$

1759. 统计同构子字符串的数目

- ① 双指针 将 连续 字符串 选出
- ② 对于每个 连续 字符串 计算 其 同构子字符串 数目
- ③ res 需对 $10^9 + 7$ 取模

时: $O(n)$

空: $O(1)$

227. 转换字符串的最少操作数

一次遍历，遇上 'X' 则将连续 3 个字符变为 'O'

时: $O(n)$

空: $O(1)$

1750. 删除字符串两端相同字符后的最短长度

双指针遍历， f, b

① 若 $*(\text{f} + i) = *b$ 且 $\text{f} + i \neq b$, $i + 1$

② 若 $*(\text{b} - j) = *f$, $j + 1$

时: $O(n)$ 完: $O(1)$

2032 至少在两个数组中出现的值

- ① 遍历 nums1 , 若某个值在 nums2 或 nums3 中出现且不在 res 中
 - a) 加入 res 中;
- ② 遍历 nums2 , 若某个值在 nums3 中出现且不在 res 中. 则加入 res 中

时: $O(n^2)$

空: $O(1)$

2037.使每位同学都有座位的最少移动次数

seats和students都进行排序一次，两数组
顺序取绝对差值，res += 绝对差值就是最
终答案

时： $O(n \log n)$

空： $O(n)$

2351.第一个出现两遍的字母

- ① 初始化一个长度为26的数组，alphabet
- ② 遍历字符串s，并将每个字母出现的次数都记录在alphabet上
- ③ 当有字母出现次数为2时，则返回该字母

时： $O(n)$

空： $O(26)$

1801.积压订单中的订单总数

堆

① 采购订单在sell中选价格最低的，sell用小堆

② 销售订单在buy中选价格最高的，buy用大堆

③ heappush和heappop连用

时： $O(n \log n)$

空： $O(n)$

检查句子中的数字是否递增

- ① s 以空格分开
- ② 遍历 s 中每个数字
- ③ 若不是严格递增则 False

时: $O(n)$

空: $O(n)$

1802. 有界数组中指定下标的最大值

二分查找

- ① 在 index 处，认为 mid 为最大值，在 index 往左，mid 依次递减，直到减为 1，右边同样的操作，这样便可统计出指定 index 下标最大值为 mid 的最小数组和
- ② 若最小数组和 $\leq \text{maxsum}$, $l = \text{mid}$ 否则 $r = \text{mid} - 1$

时: $O_c(\log n)$

空: $O(1)$

2180. 统计各位数字之和为偶数的整数个数.

① 从 2 到 num 逐个数字检查 各位数字之和 是否为偶数

② $\text{curnum} \% 10$ 取出 个位数

③ $\text{curnum} / 10$ 得到去掉个位数后的数字.

时: $O(m \cdot n)$

空: $O(1)$

1658. 将 x 减到 0 的最小操作

滑动窗口，初始化 $l = -1$, $r = 0$, $lsum = 0$, $rsum = sum(nums)$

若 $lsum + rsum = x$, 则 记录 操作数

若 $lsum + rsum < x$, 则 l 左移

若 $lsum + rsum > x$, 则 r 右移

时: $O(n)$

空: $O(1)$

2185. 统计给定前缀的字符串

遍历 words 中每个 word，逐个字符比对 pre。若每个字符都对得上

则 res +1，否则 continue

时： $O(n^2)$

空： $O(1)$

1806. 还原排列的最少操作数

直接模拟，ori，pre，cur 三个数组按题目要求模拟。

时: $O(n)$

空: $O(n)$

2283. 判断一个数的数字计数是否等于数位的值

- ① 哈希表 统计字符串每个数字 的出现次数
- ② 确认 每个数字的出现字数 是否等于数位 的值

时: $O(n)$

空: $O(n)$

1807. 替换字符串中的括号内容

- 1 用unordered_map映射每一个键值对
- 2 快慢指针找到每一个括号
- 3 若括号内容是map中的键，则替换其值，否则替换为‘?’

时： $O(n)$

空： $O(n)$

2287. 重排字符串形成目标字符串

- ① 用哈希表₁统计目标字符串各字符的次数 — count
- ② 哈希表₂统计 s 字符串中目标字符的次数 — total
- ③ $res = \min(res, total / count)$

时: $O(n)$

空: $O(n)$

1819. 序列中不同最大公约数的数目

- 1 vector<bool>记录序列中各数字
- 2 从1–maxnum枚举可能的最大公约数i
- 3 从i到maxnum翻倍，看是否存在与序列中j
- 4 若存在，则计算之前的最大公约数与当前数的最大公约数

时： $O(n \log n)$

空： $O(n)$

2293. 极大极小游戏

1 确定总循环 $\text{nums.size()} / 2 \sim 0$

2 每次循环都从 $0 \sim i$ ，奇数位置则 $\text{nums}[j] = \max(\text{nums}[j*2], \text{nums}[j*2+1])$ ，偶数位置则 $\text{nums}[j] = \min(\text{nums}[j*2], \text{nums}[j*2+1])$

时： $O(n^2)$

空： $O(1)$

1814. 统计一个数组中好对子的数目

- ① 先算出数组中每个数的 rev 版本，并计算 $\text{num} - \text{rev}(\text{num})$
存放于 revVec 中
- ② 将 revVec 中每个数的出现次数用 map 记录下
- ③ 统计好对子的数目

时: $O(n)$

空: $O(n)$

1813. 句子相似性川

- ① 将两句子的单词全部分隔出来
- ② 指针 i 从头往后检查一样的单词
- ③ 指针 j 从后往前检查一样的单词
- ④ 若某个句子所有单词都检查得一样的话，True

时: $O(n)$

空: $O(n)$

2299. 强密码检验器II

- 1 六个条件对应6个bool类型
- 2 遍历password，逐个对应上6个条件，满足则为true，不满足则为false
- 3 最后将6个bool类型做与操作返回

时： $O(n)$

空： $O(1)$

1817. 查找用户活跃分钟数

- 1 用`map<int, set<int>>`统计每个id的分钟数
- 2 再统计同一活跃分钟数的总用户数

时 : $O(n)$

空 : $O(n)$

164. 生成平衡数组的方案数

- ① 先遍历一遍数组，统计奇、偶下标的和 odd₁，even₁
- ② 再遍历一遍数组，odd₁ 和 even₁ 减去对应奇偶下标的数后判断 $odd_1 + even_2 \stackrel{?}{=} odd_2 + even_1$
- ③ ②后再 $odd_2, even_2$ 加上对应奇偶下标的数

时: $O(n)$

空: $O(n)$

2315. 统计星号

- ① 初始化 flag = false, res = 0
- ② 遍历字符串 s, 若 flag = false 且 ch = '*' , res + 1
若 flag = false 且 ch = '1' , flag = true.
若 flag = true 且 ch = '1' , flag = false

时: $O(n)$

空: $O(1)$

2331.计算布尔二叉树的值

后序遍历二叉树，计算每个节点 的值，最后返回根结点的布尔值

1 若非叶节点值为2，则左右孩子做或运算

2 若非叶节点值为3，则左右孩子做与运算

时： $O(n)$

空： $O(\log n)$

1604. 警告一小時內使用相同員工卡大於等於三次的人

- ① 用哈希表把所有員工的打卡時間記錄下來
- ② 看哪個員工在一小時內打卡大於等於三次

時: $O(n \log n)$ 空: $O(n)$

1233. 删除子文件夹

- ① folder 排序
- ② 遍历 folder，对每个文件夹路径进行检索，查看是否是子文件夹
- ③ 若不是子文件夹，添加到 vector<string> res 中

时: $O(n \log n)$

空: $O(n)$

1797. 设计一个验证系统

- ① 用哈希表记录下每个验证码的过期时间
- ② 统计未过期的验证码只需遍历哈希表逐一检查即可

时: $O(n)$

空: $O(n)$

2335. 装满杯子需要的最短总时长

每次选取较大的两个需求装水，直到没有需求数大于0

时: $O(n)$

空: $O(1)$

1138. 字母板上的路径

模拟 board 进行移动

注：当 cur 向下移动后大于 'z' 时，需先将 cur 向左移

时： $O(n)$

空： $O(1)$

1395. 统计作战单位数

从下标 $1 \sim n-1$ 遍历中间单位 j ，每次遍历统计左边严格比单位 j 大以及比单位 j 小的单位数，同时统计右边的单位数， $\text{res} += (\text{left} * \text{right}) + (\text{left1} * \text{right1})$

时： $O(n^2)$

空： $O(1)$

1140. 石子游戏II

动态规划，要求一开始的答案，需要从后往前推

对于一个堆，如果当前可取的堆数 $2*M$ 大于总堆数，则将所有堆取完

若否，则 $sum -$ 假设对手能取最多堆数

时： $O(n^2)$

空： $O(n^2)$

2357. 使数组中所有元素都等于零

循环执行

- ① 排选 nums 中非零的最小元素. min
- ② 所有非零的元素都减去 min

直到所有元素都等于零

时: $O(n^2)$

空: $O(1)$

1247. 交换字符使得字符串相同

- ① 遍历两字符串，统计 "xy" 和 "yx" 对的数目
- ② 若 "xy" + "yx" 为奇 \rightarrow False
- ③ 返回 $xy/2 + yx/2 + xy \% 2 + yx \% 2$

时: $O(n)$

空: $O(1)$

1255. 得分最高的单词合集

- 1 先统计letters中每个字母出现的次数
- 2 统计每个words的单词合集组合所需要的字母总数（用位运算进行统计）
- 3 统计每个单词合集组合的最高得分

时： $O(n + m + 2^m)$

空： $O(m)$

1144. 递减元素使数组呈锯齿状

- 1 若只有一个元素，返回0
- 2 分别计算偶数下标大和偶数下标小两种情况的操作次数
- 3 只需要对小的情况进行递减操作即可

时： $O(n)$ 空： $O(1)$

2363. 合并相似的物品

① 先用 map 把相似物品合并（带排序）

② 遍历 map 将 pair 转移到 ret 二维数组中

时: $O(n)$

空: $O(n)$

2373. 矩阵中的局部最大值

- 1 遍历二维矩阵grid (从1~n-1行，1~n-1列)
- 2 对每个位置都进行 3×3 的矩阵搜索，找到最大值，并且赋给res对应位置上

时： $O(n^2)$

空： $O(n^2)$

面试题05.02. 二进制数转字符串

- 1 初始化res=“0.”
- 2 每一位小数位开始试，如果该小数位置1仍然小于目标数num，则置1；否则置0

时： $O(n)$

空： $O(1)$

1467.保证文件名唯一

用到set记录目前已经有的文件名，用map记录当前文件名的后缀

- 1 当检查到一个文件名没有出现在set上，可以直接添加该文件名
- 2 当检查到在set上，则需要添加后缀，从map中调取合适的后缀再添加文件名

时： $O(n^m)$ 空： $O(n)$

982.按位与为零的三元组

- 1 用map统计每两个按位与结果出现的次数，当二元组为0的时候，三元组也必定会为0，所以可以不用添加到map中，res结果直接增加n，减小复杂度
- 2 再遍历一遍nums，与map中的键值做与运算，若结果为0，则res结果增加当前键所对应的值

时： $O(n^2)$

空： $O(n)$

1599. 经营摩天轮的最大利润

- ① 每轮顾客排队之时 同时服务顾客
- ② 当没有顾客来排队之后 仍需为剩余的顾客服务
- ③ 以上两过程记录利润最大的轮次

时: $O(n)$

空: $O(1)$

1653.使字符串平衡的最少删除次数

- 1 遍历字符串s，统计出a的数目righta， $res = righta$
- 2 再次遍历字符串，若此时 $ch==a$ ， $righta--$ ，说明此处右边还有 $righta$ 个a；
若此时 $ch==b$ ， $leftb++$ ，说明此处左边（包括此处）有 $leftb$ 个b；
每个位置都判断 $res = \min(res, leftb + righta)$

时： $O(n)$

空： $O(1)$

1096.花括号展开II

dfs :

S.find('x') 返回一个int指示下标，返回string::npos表示找不到

S.find('x', j)表示从j下标开始查找'x'

stringstream ss(S, b, ',')表示遍历S，以','为分隔符将S分隔开并且赋给b

时 : O (n^2)

剑指offerII 47. 礼物的最大价值

dp

1 第一行和第一列先累加起来

2 从 $(1, 1)$ 开始遍历， $(i, j) = \max((i-1, j), (i, j-1)) + (i, j)$

时： $O(mn)$

空： $O(1)$

10. 正则表达式匹配

$dp[i][j]$ 表示 s 的前 i 个字符是否与 p 的前 j 个字符匹配， $dp[0][0] = \text{true}$

① 初始化第一行 $dp[0]$ ，也就是 s 为空， $p[j-1]$ 需为 * 才可匹配

② 第一种情况， $p[j-1]$ 为普通字符，则需要判断 s 的前 $i-1$ 个字符与 p 的前 $j-1$ 个字符是否匹配，再判断 $s[i-1]$ 与 $p[-1]$ 是否匹配

③ 第二种情况， $p[j-1]$ 为 *，若 s 的前 i 个字符与 p 的前 $j-2$ 个字符已经匹配了，那也就没有必要再添加字符了；若不匹配，则需要判断 s 的前 $i-1$ 个字符与 p 的前 $j-2$ 个字符是否匹配，再判断 $s[i-1]$ 与 $p[j-2]$ 是否匹配

时： $O(mn)$

空： $O(mn)$

2379. 得到 k 个黑块的最少涂色次数

① 先统计下标 $0 \sim k-1$ 的白块个数， cnt

② 双指针 $i=0, j=k$ ，若 $\text{blocks}[i] = W$ ， $\text{cnt} - 1$

若 $\text{blocks}[j] = W$ ， $\text{cnt} + 1$ ， $i+1, j+1$

时： $O(n)$

空： $O(1)$

1590.使数组和能被P整除

- 1 采用 $x = (x+num) \% p$ 的方法求出整个数组和模p的结果，若 $x==0$ ，则不需要移除子数组
- 2 用前缀和的思想，结合哈希表iim。当前前缀和模p后的结果cur，如果目标值 $target=(cur-x+p)\%p$ 出现过，则 $ans = \min(ans, iim[target])$ ，否则*iim[cur] = i*

时： $O(n)$

空： $O(n)$

面试题17.05. 字母与数字

前缀和+哈希表

- 1 根据当前遍历的是否为字母计算前缀和sum
- 2 若sum并未出现过，则添加到哈希表中，映射当前的下标
- 3 若sum出现过，则判断当前下标*i*-hash[sum]是否是最大的，若是，则最长子数组为hash[sum]开始，并且长度为*i*-hash[sum]

时： $O(n)$

空： $O(n)$

2383.赢得比赛的最少训练时长

- 1 先计算energy数组求和energy_sum，若大于initialEnergy，则需要先训练提升initialEnergy到energy_sum+1
- 2 遍历与每个对手进行比拼，若此时的initialExperience > exp，直接加上exp
否则需要提前训练至exp+1，再加上exp

时： $O(n)$

空： $O(1)$

1605.给定行和列的和求可行矩阵

- 1 初始化二维数组，并且 $\text{res}[i][0] = \text{rowSum}[i]$ ，即把每一行的和都放在第一列
- 2 对列进行分配，从上至下，如果当前列已经满足列的和，则把多余的数值挪到下一列

时： $O(n^2)$

空： $O(n)$

题目：一圈数（可正可负），求最大的连续和

- 1 先求普通数组的最大连续和（假设最大连续和不需要经过首尾）
- 2 求除了首尾元素的最小连续和，再用总和减去最小连续和（假设需要经过首尾）

16.15. 最大网络秩

① 用 set 统计出每座城市所连的城市

② 枚举两两城市找到相连城市的最大值

时: $O(n^2)$

空: $O(n)$

2488.统计中位数为k的子数组

前缀和+哈希表

- 1 先找到k的下标idx，用map统计好对应下标的前缀和，初始化um[0] = 1
- 2 遍历数组，若当前数小于k，sum-=1,若当前数大于k，sum+=1，否则sum不变
- 3 若当前下标i<idx，直接um[sum]++，否则4
- 4 计算pre0 = um[sum]，pre1=um[sum-1]，res+=pre0+pre1

时： $O(n)$

空： $O(n)$

2389.和有限的最长子序列

- 1 用map统计每个数字与其出现次数
- 2 对于每个query，采用贪心法，从最小的数字开始凑子序列，直到子序列和大于query

时： $O(nm)$

空： $O(n)$

1616. 分割两个字符串得到回文串

- 1 双指针，指针i指向a头，指针j指向b尾，若 $[i] == [j]$ ，则 $++i, --j$
- 2 遇到不一样的地方，则要检查两字符串内部 $[i,j]$ 是否为回文串

时： $O(n)$

空： $O(1)$

6319.奇偶位数

1 << cnt++ 与 n 相与，若相与结果为1，偶数位则even++，奇数位则odd++

最后返回{even, odd}

时 : O (logn)

空 : O (1)

6322.检查骑士巡视方案

- 1 先初始化骑士的8个移动方向
- 2 模拟一遍骑士的移动，看看是否能与题目给的grid的顺序匹配的上

时： $O(n^2)$

空： $O(1)$

6352. 美丽子集的数目

- 1 dfs，每个下标的元素都会进行选或不选的操作
- 2 若要选择当前元素，还需要判断已选子集当中是否存在 $\text{nums}[\text{idx}] - k$ 和 $\text{nums}[\text{idx}] + k$

注意：在选择了当前元素后，回溯回来的时候要恢复现场，即 $\text{rec}[x]++$ 后要恢复 $\text{rec}[x]--$

时： $O(n^2)$

空： $O(n^2)$

6321. 执行操作后的最大MEX (minimum excluded)

- 1 对数组中每个数都对value取模得到非负数
- 2 采用map记录当前数cur出现的次数，将 $cur + value * \text{map}[cur]$ 添加到set中
- 3 遍历set，得到第一个缺失的数即为MEX

时： $O(n)$

空： $O(n)$

72. 距离编辑

dp

- ① 若两字符串有空串，那么返回两字符串较长者的长度
- ② 初始化二维数组 $dp[0][i]$ 以及 $dp[i][0]$ ，意思为当其中一个字符串为空时需要多少次操作
- ③ 统计三个操作，一是从 $A[:i-1]$ 操作成 $B[:j]$ 的次数，记为left，二是从 $B[:j-1]$ 操作成 $A[:i]$ 的次数，记为down，三是直接继承 $A[:i-1]$ 和 $B[:j-1]$ 的次数，判断 $A[i]$ 和 $B[j]$ 是否一致，记为leftdown
- ④ $dp[i][j]$ 选则③中较小者

时： $O(n^2)$

空： $O(n^2)$

1626.无矛盾的最佳球队

dp

- 1 用vector<pair<int,int>>装球员的分数以及年龄，按分数大小排序
- 2 对于位置i，其之前的位置j，如果年龄大于等于j者，取dp最大者，最后dp[i]+i的分数
- 3 在处理2的时候保存dp最大者

时： $O(n^2)$

空： $O(n)$

76.最小覆盖子串

滑动窗口

- ① 双指针i，j，map cur 和target
- ② j往右遍历，当cur已经包含了target时，需要右移i使得cur刚好包含target窗口和不包含target
- ③ 在处理②的时候记录下最小的窗口

时： $O(m*n)$

空： $O(m+n)$

1630.等差子数组

- 1 遍历l和r数组取出子数组范围
- 2 截出子数组之后排序，再判断是否满足等差的特性

时： $O(mn)$

空： $O(n)$

84.柱状图中的最大矩形

单调栈原理：当前遍历的高度heights[i]可达的左右边界

- ① 采用单调栈找出每个高度的左边界
- ② 采用单调栈找出每个高度的右边界
- ③ 算出当前高度的最大面积($\text{right}[i] - \text{left}[i] - 1$) * heights[i]

时： $O(n)$

空： $O(n)$

1574.删除最短的子数组使数组有序

- 1 倒序遍历j看是否数组本身就是非递减的，如果是则不需要删除子数组
- 2 顺序遍历i看 $\text{arr}[i]$ 是否能和 $\text{arr}[j]$ 连接上，若有 $\text{arr}[i] > \text{arr}[j]$ 说明连接不上，需要删除中间部分
若能连接上， $\text{res} = \min(\text{res}, j - i - 1)$
- 3 若出现 $\text{arr}[i] > \text{arr}[i-1]$ ，说明此时已找到要删除的子数组开头了

时： $O(n)$

空： $O(1)$

98.验证二叉搜索树

- 1 初始化一个pre = 最小值
- 2 中序遍历二叉树，每个值都不能必须大于pre，并且结束后将该节点值赋给pre

二叉搜索树的中序遍历是一个严格递增序列

时： $O(n)$

空： $O(\log N)$

2395.和相等的子数组

- 1 用unordered_set记录相邻的两个数的和
- 2 若当此时的和已经在set中出现过时，则返回true
- 3 否则false

时： $O(n)$

空： $O(n)$

1638.统计只差一个字符的子串数目

- 1 长度范围从1到s.size()，每次截取相同长度的子串
- 2 ——比对两子串的每个字符，看是否只差一个字符

时： $O(n^2 * m)$

空： $O(n+m)$

1092.最短公共超序列

1 先用dp列出两序列的公共子序列长度

2 逆序遍历dp，若 $dp[i][j] = dp[i-1][j]$ ，说明当前i字符不在公共子序列当中，是超序列需要的
若 $dp[i][j] = dp[i][j-1]$ 同理，若 $dp[i][j] \neq dp[i-1][j-1]$ 且 $dp[i][j] \neq dp[i-1][j]$ ，说明当前字符是公共子序列中的，只需要添加一个即可

时： $O(n^2)$

空： $O(n^2)$

1641.统计元音字典序字符串数目

- 1 初始化 $a=1, e=1, l=1, o=1, u=1$
- 2 每一轮以a结尾的字符串会产生5个新字符串，e结尾的会产生4个。。。
- 3 即 $u+=(a+e+l+o), o+=(a+e+l) \dots$

时： $O(n)$

空： $O(1)$

1637.两点之间不包含任何点的最宽垂直区域

- 1 按点的横坐标排序
- 2 找相邻下标的两个点的横坐标差值的最大值

时： $O(n \log n)$

空： $O(1)$

2367. 算数三元组的数目

- 1 用set统计nums每个数字
- 2 遍历nums，若当前num+diff和num+2*diff存在于set中，则算是一个算数三元组

时： $O(n)$

空： $O(n)$

1039.多边形三角剖分的最低得分

dp

1 i倒序遍历n-3到0

2 j遍历从i+2到n

3 k遍历从i+1到j

4 i，j，k构成三角形的三个顶点， $dp[i][j] = \min(dp[i][j], dp[i][k] + dp[k][j] + v[i] * v[j] * v[k])$
 $Dp[i][k]$ 表示点i与点k之间的点构成的最低得分， $dp[k][j]$ 同理

时： $O(n^3)$

空： $O(n^2)$

1053. 交换一次的先前排列

- 1 倒序遍历arr，若有 $\text{arr}[i] > \text{arr}[i+1]$ 情况则进行2
- 2 再倒序遍历arr，若 $\text{arr}[j] \geq \text{arr}[i] \text{ } || \text{ } \text{arr}[j] == \text{arr}[j-1]$ ，则 $--j$

时： $O(n)$

空： $O(1)$

2427. 公因子的数目

- 1 $\text{mid} = \min(a, b) / 2$
- 2 遍历i从1到mid，若i同时为a和b的公因子， $\text{res}++$
- 3 若a和b本身就有公因子关系， $\text{res}++$

时： $O(n)$

空： $O(1)$

1017.负二进制转换

- 1 若 $n=1$ 或 $n=0$ ，直接转换成字符串返回
- 2 若 $n \neq 0$
- 3 计算 $rem = n \& 1$ ，作为res的最高位， $n -= rem$ ， $n /= -2$ ，重复判断2

时： $O(\log n)$

空： $O(n)$

2399.检查相同字母的距离

- 1 遍历字符串s，拿到每个字符以及其位置i，判断其位置前后距离distance[ch]的位置是否有相同字符

时： $O(n)$

空： $O(1)$

1019.链表中的下一个更大节点

- 1 遍历一遍链表求出有多少个节点n
- 2 创建结果数组vector<int> (n, 0)， 创建栈
- 3 再次遍历链表，用单调栈思想，若栈空或栈顶元素大于当前节点值，则入栈，
否则，将所有比当前节点值小的元素弹出，并且在数组对应位置记录下一个更大节点为当前节点

时 : O (n)

空 : O (n)

1041. 困于环中的机器人

- 1 先按照指示完整走一遍，得出走完之后的方向与坐标
- 2 若坐标仍是 $(0, 0)$ ，则 true
- 3 若坐标不是 $(0, 0)$ ，则判断方向有没有改变，如果有改变则被困住了，没有改变就没有困住

时： $O(n)$

空： $O(1)$

1147. 段式回文

- 1 指针i从前往后遍历，且 $\text{left} += \text{text}[i]$ ，指针j从后往前遍历，且 $\text{right} = \text{text}[i] + \text{right}$
- 2 对比 $\text{left} == \text{right}$? $\text{left} = "", \text{right} = "", \text{res} += 2$? 继续
- 3 最后查看 $\text{left} != "" \parallel \text{i} == \text{j}$? $\text{Res} + 1 : \text{res}$

时： $O(n^2)$

空： $O(1)$

1043.partition array for maximum sum

采用dp算法

1 先计算数组开头的k个元素的最大和，即截止目前元素最大值*元素个数

2 从k到n-1， $dp[i] = \max(dp[i-j] + \text{getMax}(arr, i-j+1, i) + j, dp[i])$
含义为尝试每种不同的区分方法，取和最大者

时： $O(n*k)$

空： $O(n)$

1027.Longest Arithmetic Subsequence

dp

- 1 定义 $dp[n][1001]$ (每个num范围时0–500，所以相邻两数之差范围是–500 – 500)
- 2 遍历i 从1到n
- 3 遍历j 从0到i，计算出差值 $tmp = nums[i] - nums[j]$ ，更新 $dp[i][tmp]$
- 4 ②③过程中记录最大的dp

时： $O(nm)$

空： $O(nm)$

1105. Filling Bookcase Shelves

Dp

① 由于输入范围书本最多1000本，每本书最高1000高度，所以初始化dp (n+1, 1000001)

② 遍历每本书i，首先设定 $dp[i] = dp[i+1] + h$ ，即该书另起一行，放在原本之上

③ 逆序遍历j : 0-i，找到目前最高的高度 $h = \max(h, books[j])$ ，再统计当前总宽
 $w += books[j][0]$ ，倘若未超过限宽，则更新 $dp[i] = \min(dp[i], dp[j-1] + h)$ ，否则结束循环

时： $O(n^2)$

空： $O(n)$

239. Sliding Window Maximum

Duque

- 1 先遍历 $0-k-1$ ，始终维护dequeue最后一个下标对应的值大等于 $\text{nums}[i]$
- 2 遍历 $k-n$ ，先按序弹出最旧的值，同样维护dequeue最后一个下标对应的值大等于 $\text{nums}[i]$ ，
 $\text{res}[i-k+1] = \text{nums}[\text{q.front}()]$ ，q最前面的下标对应的值是目前窗口的最大值

时： $O(n)$

空： $O(n)$

297. Serialize and Deserialize Binary Tree

BFS

- 1 采用bfs将二叉树转换成字符串，所有非空节点都有两个子节点，不管是空

注意：第一个节点前不需要添加空格

- 2 采用根左右的顺序，用queue把当前非空节点的子节点添加到queue中

注意：使用substr时要小心超出string的长度

时： $O(n)$

空： $O(n)$

1139. Last Substring of Lexicographical Order

双指针

1 初始化指针i指向s首，j指向s[1]

2 逐个字符对比 $s[i+k]$ 与 $s[j+k]$ ，如果相等，则 $++k$ ，否则到3

3 判断 $s[i+k]$ 与 $s[j+k]$ 的大小，若小于，则i要指向j的位置，而j指向 $j+k+1$ ，否则到4

4 j往后挪

时： $O(n)$

空： $O(1)$

1031.Maximum Sum of Two Non-Overlapping Subarrays

DP+滑动窗口

- 1 sum_l 初始化为 idx0–firstlen–1 的和，sum_r 初始化为 idxfirlen–secondlen–1 的和， max_l 初始化为 sum_l， i = firstlen， j = firstlen + secondlen
- 2 每次 i 往后移，sum_l 也往后移，若新的 sum_l 更大则更新 max_l，sum_r 往后移
- 3 2 的意义为统计两个滑动窗口的最大和
- 4 两个滑动窗口的位置更换再进行一次 1 2 3

时 : O (n)

空 : O (k)

2106. Maximum Fruits Harvested After at Most K Steps

- 1 双指针i，j代表一个范围
- 2 遍历j 从 0 – n，如果 $pj - fruits[i][0] + \min(\text{abs}(\text{startPos} - fruits[i][0]), \text{abs}(\text{startPos} - fruits[j][0])) > k$, 则窗口左侧i右移，直到没有窗口或者上面的距离小于等于k
- 3 统计窗口内收获的值的最大者

时 : $O(n)$

空 : $O(1)$

2432. The Employee worked on the Longest Task

- 1 遍历tasks，通过 $\text{tasks}[i][1] - \text{tasks}[i-1][1]$ 找到工作时间最长的员工
- 2 若工作时间相同者则选取id号更小者

时： $O(n)$

空：(1)

1016. Binary String of Substring representing 1 to N

- 1 遍历 i 从 $1 \sim N$ ，将每个 i 转化为32位二进制
- 2 将二进制数转化为字符串 str ，并去除前导0
- 3 在给定的字符串 s 中找 str ，如果找不到则返回false

时： $O(n)$

空： $O(1)$

2446. Determine if Two Events Have Conflict

- 1 将四个时间都转换为数字
- 2 对比两个时间段是否有交集

时 : O (1)

空 : O (1)

2605. 从两个数字数组里生成最小数字

- 1 遍历nums1和nums2，使用vector记录下nums1和nums2中出现的数字
- 2 若两数组有重复出现的数字，选择重复出现的数字中最小的，这个数字就是答案
- 3 否则，在两数组中选择最小的数字组成一个两位数，这个两位数就是答案
- 4 选择出的两个数字，让更小的充当十位，这样能使组成的数字更小

时： $O(n)$

空： $O(n)$

$UE1$ 天线 + $UE1$ 接收功率 + $UE1$ 电量 + $UE2$ 接收功率 + $UE2$ 电量

① 电量管理应在基站端 (为了手动控制充电)

$\hookrightarrow UE1$ 总电 10mW

保证用户端同步不出错

$UE2$ 总电 20mW

② reward 需要考虑接收功率与电量需求

① 电量最大化系统的接收功率

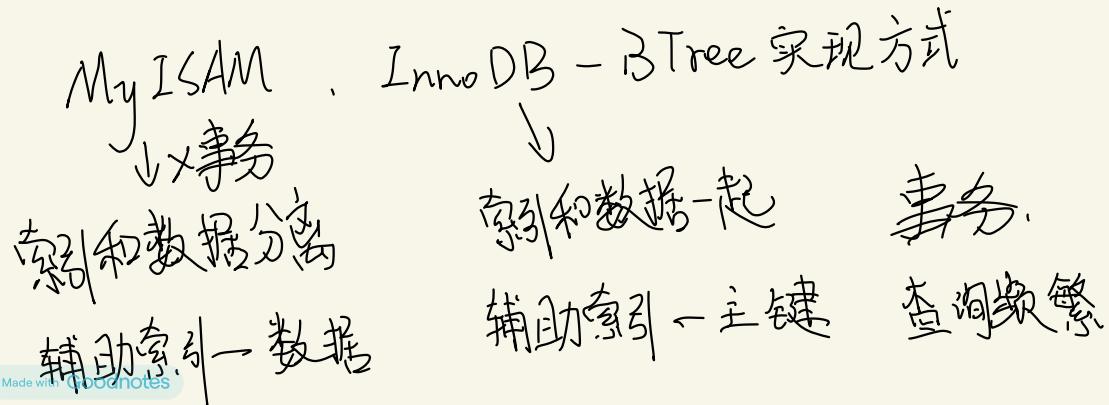
$\hookrightarrow UE1 = UE2 = 10mW$

② 电量饱和了之后对电量需求会变小

$$Q(s, a) = (1 - \alpha) Q(s, a) + \alpha (\text{reward} + \gamma^{\max} | Q(s_{\text{next}}) |)$$

$$Q(\{16, 23\}, 0, -26, 0), x) = (1 - \alpha) \cdot Q(\{16, 23\}, 0, -26, 0), x) \\ + \alpha (\text{reward} + \gamma^{\max} | Q(\{x, 23, 0, -26, 0\}) |))$$

多版本并发控制 (WVCC) → InnoDB，快照，多事务
间隙锁 (Next-Key Locking) — 索引间隙，×插入
聚簇索引 — 物理排序，定位快，查询频繁，记录顺序 = 索引顺序
支持事务与不支持的区别 — 原子性，一致性，隔离性，持久性
在线热备份 — 运行期间备份



四种索引 - FULLTEXT、HASH、BTREE、RTREE

MyISAM 一次定位 黑色默认，常用
MySQL 5.6 ``=`` 和 ``in`` 高效 树形结构 geometry

CHAR, VARCHAR X 范围、排序

TEXT

视图 - 虚拟的表，简化操作 X 索引 X 更新

叶树 - 磁盘读写代价低，效率稳定，遍历叶子节点。

索引 - 频繁使用的排序，
提高 where 速度 X 很少用 X 重复列较多 X TEXT
列上 X 函数 X 运算

覆盖索引 - 查询所需的列都在索引中，避免回表

第一范式：无重复的列 第二范式：非主属性完全依赖于主键 第三范式：属性不依赖非主属性

创建索引：① 非空字段 ② 取值之间差异程度大 ③ 唯一、不为空、常用的字段

逻辑主键：与业务无关，如自增 ID

业务主键：与业务相关，如商品编码

limit - offset：相同查询每次结果可能不同，解决：不同分页请求产生不同的缓存键

InnoDB 的行锁：通过索引加锁，若无索引则加表锁。

交叉连接：两个表的每一行都进行组合。

Redis - 事务脚本化 . LUA 脚本 . LRU 驱动事件 . 多种集群方案
RDB . AOF 组合 Redis 命令 内存淘汰机制

MurmurHash : 即使有序输入也有很好的随机性

每个字典带有 2 个哈希表 : 一个平时用, 另一个 rehash 用

Redis : 高性能 . 高并发 - 都是利用缓存

分布式缓存 , 多实例共享
在内存中

why 快过 Memcached : ① 纯内存 ② 单线程 ③ 非阻塞多路 I/O

why not 多线程 : ① 简单高效 ② 线程竞争 ③ I/O 多路亦有并发性

过期 : ① 定期删除 (随机) ② 随性删除 (使用前检查)

内存淘汰机制 : LRU . Random . no . eviction

RDB 故化：快照持久化、默认方式，利用主从服务器

AOF 故化：每次更改都写入磁盘

AOF 重写：创建更小的 AOF 文件，只记录当前数据库状态。每次操作将新 AOF 替换旧的
(避免 AOF 文件过大)

哨兵：高可用，主服务器倒了，从服务器顶上

集群：分布式，单个 Redis 内存不足，可以分片存储（主写从读）

并发竞争 key：

预留缓存模式：①读时先查缓存 再查数据库 ②写时先删缓存 再改数据库

缓存淘汰法：

异步淘汰缓存：

脑裂：由于网络问题，slaves 认为 master 倒机了，实际并没有，最后选出 2 个 master

解决：长时间有 slave 未同步，则不接收请求

主观宕机 $sdown$ ，客观宕机 $odown$

