

贪心算法

核心思想：局部最优→全局最优

如果这种思想找不到反例，题目就可以用贪心算法

分发饼干

思路：把大饼干分给大胃口的孩子（**局部最优**），从而把饼干喂给更多的孩子（**全局最优**）

*如果大饼干分给较小胃口的孩子，那么小饼干就很可能分不出去

实现：先对g和s排序（Arrays.sort），然后对这两个数组从后往前遍历，**胃口是固定遍历**（for循环），**饼干是条件遍历**（当前饼干大于等于胃口则将饼干往前一位）

*不能固定遍历饼干，条件遍历胃口，这样有可能一个符合条件的分发都没有

题目：

455. 分发饼干（简单）

摆动序列

一定要把坡度图画出来分析！

局部最优：删除单调坡度上的节点（不包括底和顶），这样这个坡上就有两个局部峰值

全局最优：序列有最多数量的局部峰值

大体思路：

考虑遍历节点的prediff (nums[i]-nums[i-1]) 和curdiff (nums[i+1]-nums[i])，如果prediff>0且curdiff<0，或prediff<0且curdiff>0，就可以纳入最长序列的统计

遍历的方式是从数组头元素开始，一直到倒数第二个元素，每次计算curdiff，当前prediff为循环上一次的curdiff

然而有以下**三种特殊情况**：

(1) 上下坡中有平坡：统一删除左边的所有平坡元素，仅保留最右边的元素，此时判断条件对应修改成prediff≥0且curdiff<0，或prediff≤0且curdiff>0

(2) 首尾两端：

- a. **首元素**：可以视作在它之前虚拟给一个同等值的平坡，此时符合（1）中的判断，将首元素纳入长度的统计，具体体现在**初始化prediff=0**
- b. **尾元素**：由于遍历的尽头是数组倒数第二个元素，而尾元素根据题目规则又需要算进去，因此默认尾元素是峰值，体现在**初始化序列长度count=1**

（3）单调上坡/下坡中有平坡：如果每次循环都进行prediff和curdiff的传递，那么在这种情况下就会出问题（平坡最右边的元素会被算入结果）。为规避这种情况，仅在判断条件成立时才进行prediff=curdiff的传递

*数组元素长度为0或1时，需要返回数组长度

题目：

376. 摆动序列（中等）

最大子数组和

局部最优：当遍历到子数组的和为负数时，立刻抛弃整个子数组，以下一个元素为起点继续遍历

（如果子数组和为负数，对下一元素而言，整体的和一定会被拉低）

*换言之，保证数组里的每个元素，它前面子数组的和一定为正

全局最优：根据局部最优的方式，最终可以得到最大子数组和

实现：将结果res初始化为无穷小，子数组和count初始为0，每次遍历将count与数组元素相加，如果count比res大则将res变为count，如果count ≤ 0 则将count置为0

细节：

（1）将res初始化为无穷小，可以保证如果数组元素全为负（如[-2, -2, -2, ...]）时结果正确

（2）count比res大才更新res，可以保证如果数组出现[4, -1]时结果正确，如果-1后面还有正数则正常更新res。总而言之，这样可以保证**遍历全过程中对最大子数组和的求解都是正确的**

（3）注意区分概念：是不能让“连续和”为负数的时候加上下一个元素，而不是不让“连续和”加上一个负数

题目：

53. 最大子数组和（中等）

买卖股票的最佳时机II

问题拆解：假如说第0天购买，第3天卖出，利润为 $\text{nums}[3] - \text{nums}[0]$ ，但实际上可以拆分成 $(\text{nums}[3] - \text{nums}[2]) + (\text{nums}[2] - \text{nums}[1]) + (\text{nums}[1] - \text{nums}[0])$ ，这样便于我们确认局部最优和全局最优

局部最优：选取 nums 前后元素之差为正的

全局最优：选取所有 nums 前后元素之差为正的进行累加

*注意：“问题拆解”并不反映真实情况，并不是说买3次卖3次，而是第0天买第3天卖
题目：

122. 买卖股票的最佳时机II（中等）

跳跃游戏

跳跃游戏的两个题都可以用贪心算法解决

跳跃游戏I：局部贪心，每次跳到最远距离，即跳到当前元素值的步数。这个距离用 rightmost 记载，它是可以到达的最右边位置。具体表现为：**当遍历位置小于等于 rightmost 时**，不断更新 rightmost 使其越来越右，若最后 rightmost 能到达数组最右边甚至更远，则能到终点，反之则不能。

（遍历位置小于等于 rightmost 这个条件很重要，遍历位置大于 rightmost 的统计和值的更新是无效的，因为根本就不能到达这个位置）

跳跃游戏II：局部贪心，获取当前最远覆盖位置 cur 和下一元素最远覆盖位置 next ，初始化它们为0。遍历数组元素（ $\text{nums.length}-2$ ，不包括尾元素，否则结果会多1）：遍历位置小于等于 next 时，不断更新 next 到更远覆盖位置；当遍历位置等于 cur 时，就把结果加1，并且把 cur 变为 next 的位置，最后返回结果，其实返回的就是所有覆盖长度段（包括第一个元素也有一个长度段）的间隔数。

***这两个题都要对数组长度为1的情况进行单独说明**

题目：

55. 跳跃游戏（中等）

45. 跳跃游戏II（中等）

K次取反后最大化的数组和

局部最优：把绝对值大的负数取反变为正

全局最优：把尽可能多的负数变为正，实现和最大

步骤：

1、把数组排序

2、遍历数组进行**第一次变化，仅对负数操作**（遇到正数直接break循环），每次对负数元素取反后k要减1，第一次变化后有以下几种情况——

（1）不够把所有负数都变为正，所以需要设置遍历条件为i小于数组长度且k大于0

（2）所有负数变为正后k仍大于0（包括遍历已到尽头或绝对值最小的负数后面还有正数两种情况）

3、把更新的数组再排序（不影响步骤2的情况（1），因为此时k=0，对应下面情况b），排序后**进行第二次变化，有两种情况：**

a. k为奇数，此时需要将排序后最小的数取反

b. k为偶数，此时不做变化（相当于对同一个数操作）

4、把进行第二次变化后的数组进行for循环加和，得到最终结果

题目：

1005. K次取反后最大化的数组和（简单）

加油站

（和“最大子数组和”题目非常像）

局部最优：考虑 $gas[i]$ 和 $cost[i]$ 的差值，从 $i=0$ 开始遍历，如果从起点位置start开始的各差值之和 $cursum$ 小于0，说明从 $cursum$ 开始算的起点到i这个区间任何一个位置出发，都不可能走完一圈，此时把车的**起点start设为 $i+1$ 且 $cursum$ 置0**

全局最优：数组内**所有差值totalsum**加起来小于0，则不可能走完一圈，返回-1，否则返回start

（start, cursum, totalsum初始化为0）

题目：

134. 加油站（中等）

分发糖果

关键：右边比左边分数高，和左边比右边分数高要分开考虑

局部最优：相邻元素谁更大，则它对应的res数组值就多1

全局最优：按照全局最优的思路，获得最少分发的糖果数量

具体实现：

- a. **考虑右边比左边分数高**：需要从第二个元素（ratings[1]）开始**从前往后遍历** ratings数组，先把结果数组res头元素全部置1，遇到ratings数组元素比前一个元素大时，则将res对应位置的值加1，否则不变
- b. **考虑左边比右边分数高**：此时需要**从后往前遍历**（因为要依赖后面元素的值），从倒数第二个元素开始，遇到rating数组元素比它后一个元素大时，由于需要同时满足a、b两种情况，因此要取res后一个元素的值+1与自身res的最大值
- c. 遍历res数组把结果累加

题目：

135. 分发糖果（困难）

柠檬水找零

局部最优：收到5块，不找零；收到10块，返回5块；收到20块，**优先返回10块+5块**（把更多的5块留给找零顾客的10块），再考虑返回3个5块

全局最优：按照局部最优的思路获得题目答案

具体实现：需要维护5块钱的张数sum5和10块钱的张数sum10（20块钱的张数没必要统计，派不上找零的用场），有以下几种情况：

- (1) **遇到5块**，则把sum5加1；
- (2) **遇到10块**，则查看sum5的张数：如果 <1 ，则无法找零，返回false；否则把sum5减1，sum10加1；
- (3) **遇到20块**，先看sum10的张数：
 - a. **如果sum10张数 ≥ 1** ：在这情况下**sum5的张数如果 <1** ，则返回false，否则sum5和sum10均减1；
 - b. **如果sum10张数 < 1** ：需要3个5块去找零，**如果sum5 < 3** ，则返回false，否则sum5减3；
- (4) 遍历循环没有返回false，则说明能按题目条件找零，循环结束后返回true

题目：

860. 柠檬水找零（简单）

根据身高重建队列

关键：本题涉及身高h和人数k两个维度，需要先确定一个维度，再确定另一个维度

- a. 如果按人数k排序，会发现k的排列并不符合条件，h的排列也不符条件且无规律，无法确定维度
- b. 如果**按身高h排序**，如果**从高到低排序**，且**相同身高下按k的升序排序**，那么在重建队列时，把某个身高-人数（h-k）对**插入到索引k的位置**，能保证他前面的人比他高且数量正确

（也就是说，需要让矮个子的人掌握选择位置的主动权）

局部最优：优先按身高更高的people的k来插入

全局最优：做完所有插入操作后满足题目条件

实现细节：由于涉及插入操作，用**链表**的数据结构更加方便，最后再把链表转为数组

***注意排序和重建队列的写法，详情看leetcode代码**

题目：

406. 根据身高重建队列（中等）

用最少数量的箭引爆气球

关键：重叠区间问题（此题以及往下3个题都是区间问题）

局部最优：往**重叠区间**射，可以保证数量最少，且找不出反例

全局最优：按局部最优的方式，能得到正确答案

具体实现：

- (1) 将气球区间**升序排序**（按左边界和右边界排都可以，本题按**左边界**）
- (2) 初始化结果res为1（至少需要一支箭）
- (3) 从第二个区间（i=1）开始遍历，如果它的**左边界大于上一个区间的右边界**，此时需要多加一支箭，res加1
- *注意一定是“大于”不是“大于等于”，因为两个气球挨着没有交集（如[1,2]，[2,3]），一支箭也能搞定**
- (4) 否则就把当前区间的右边界更新为**自己右边界和上一区间右边界的最小值**（可以反证法讨论更新在该位置的左边和右边，会出现什么问题）

细节：为防止溢出，在左边界升序排序时用**Integer内置比较方法**，详情看Leetcode代码

题目：

452. 用最少数量的箭引爆气球（中等）

无重叠区间

局部最优：保证每个区间往其它区间渗入尽可能小，这样更不容易重叠

全局最优：按局部最优的方式，保证删除的区间数量最少

***数值区间问题：通常需要对区间排序（常按左边界排）**

具体实现：

- （1）将区间按左边界升序排序（注意采用Integer内置比较方法）
- （2）初始化结果res为0
- （3）从第二个区间（ $i=1$ ）开始遍历，如果它的左边界 \leq 上一区间的右边界，**就把该区间的新右边界设置为它原来的右边界和上一区间右边界的最小值**，此时代表要删除一个区间，res加1
(用这个程序，同时实现了左边界 $>$ 上一区间右边界时不进行删除操作)
- （4）最后返回结果

题目：

435. 无重叠区间（中等）

合并区间

（此题和贪心算法关系不大，但和“无重叠区间”的方法类似，可以对比总结）

采取策略：对合并结果的数组最后一个区间不断进行操作

具体实现：

- （1）将区间按左边界升序排序（注意采用Integer内置比较方法）
- （2）初始化将第一个区间加入结果链表res中
- （3）从第二个区间（ $i=1$ ）开始遍历，如果左边界 \leq 上一区间的右边界，**就把res末尾区间的新右边界改为它的右边界和遍历区间右边界的最大值**，然后移除末尾区间，把合并后的新区间加入res（注意写法，参考leetcode代码）
- （4）左边界 $>$ 上一区间的右边界，说明不需要合并，**直接把当前区间加入res中**
- （5）最后把res转为数组形式返回结果

题目：

56. 合并区间（中等）

划分字母区间

思路：两次遍历——

第一次：统计每个字母最后一次出现的位置，利用数组型哈希表实现；

第二次：统计每个区间的最右边界（取自己和当前遍历字母最后一次出现位置的最大值），直到遍历位置和这个最右边界的值相等，即可记录答案

（注意：记录答案需要记下区间的始末位置，才能得到长度，且需要不断更新区间初始位置）

***通过以上思路，可以保证划分的区间尽可能多且符合题意**

题目：

763. 划分字母区间（中等）

单调递增的数字

局部最优：当遇到当前数字高一位的数字比自己大时，要把高一位的数字减1，从最后一次减1的地方开始把它后面的数全部变为9

全局最优：按局部最优的方式，可以得到最大的单调递增数字

具体实现：

- （1）把整数转成字符串形式，字符串每一位代表一个数字
- （2）设定把后面数字全部变为9的起始位置start，初始化为字符串的长度
- （3）从后往前遍历字符串，如果遇到当前数字高一位的数字比自己大，则将高一位的数字减1，且start变为当前数字的位置
- （4）以start为起点向后遍历字符串，把后面所有位全部置为9
- （5）最后把字符串转为整数

***遍历顺序一定是从后往前**，如果从前往后遍历会出问题（学会举例，如332）

***注意把整数转为字符串，把字符串转回整数以及对字符串的数字减1的写法**（见leetcode代码）

题目：

738. 单调递增的数字（中等）

监控二叉树

局部最优：尽可能在叶节点的父节点上加监控，不要在叶节点上加监控，要求我们从下往上看

全局最优：按局部最优的方式，可以得到最少数量的摄像头

难点：

(1) 二叉树的遍历——优先考虑叶节点的情况，因而采用**后序遍历**（左右中）

(2) 安置摄像头的具体操作——

a. 分析每个节点存在的**所有状态**（无摄像头覆盖、自身有摄像头、有摄像头覆盖）

b. 善用**数字标记**（如上面三种状态分别代表0、1、2）

具体实现：参看leetcode代码，写得很详细，注意分情况讨论，以及递归函数注意对应递归三部曲

题目：

968. 监控二叉树（困难）