

极客时间算法训练营

第十课

贪心

李煜东

《算法竞赛进阶指南》作者



目录

1. 贪心理论与常见的证明方法
2. 贪心题目实战

贪心算法

贪心算法（Greedy Algorithm）是一种

(1) 在每一步选择中都采取在当前状态下的最优决策（局部最优）

(2) 并希望由此导致的最终结果也是全局最优

的算法

贪心算法与一般的搜索，以及后面要讲的动态规划相比，不同之处在于：它不对整个状态空间进行遍历或计算，而是始终按照局部最优选择执行下去，不再回头。

因为这个特性，贪心算法不一定能得到正确的结果

除非可以证明，按照适当的方法做出局部最优选择，依然可以得到全局最优结果

能用贪心求解的题目，也都可以用搜索或动态规划求解，但贪心一般是最高效的

引入：零钱兑换问题

零钱兑换

<https://leetcode-cn.com/problems/coin-change/>

给一个硬币面额的可选集合 coins，求拼成金额 amount 最少需要多少枚硬币。

例：coins = [20, 10, 5, 1], amount = 46

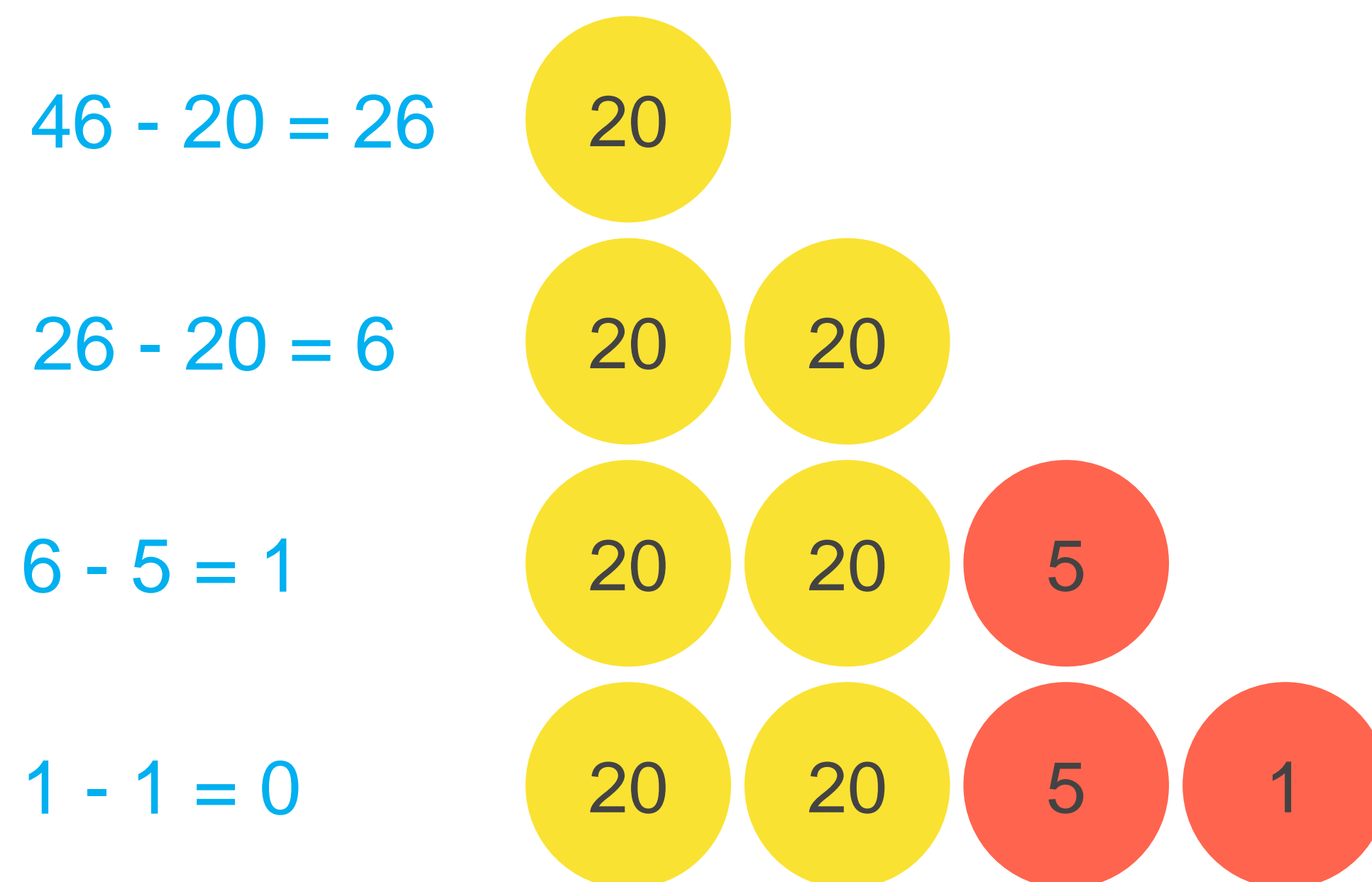
答案：46 = 20 + 20 + 5 + 1

我们将用搜索、贪心、动态规划（下一节课）三种思路来探讨这个问题

零钱兑换：贪心

根据我们平时找钱的思路，一般我们会先考虑面值大的，零钱再用面值小的凑齐

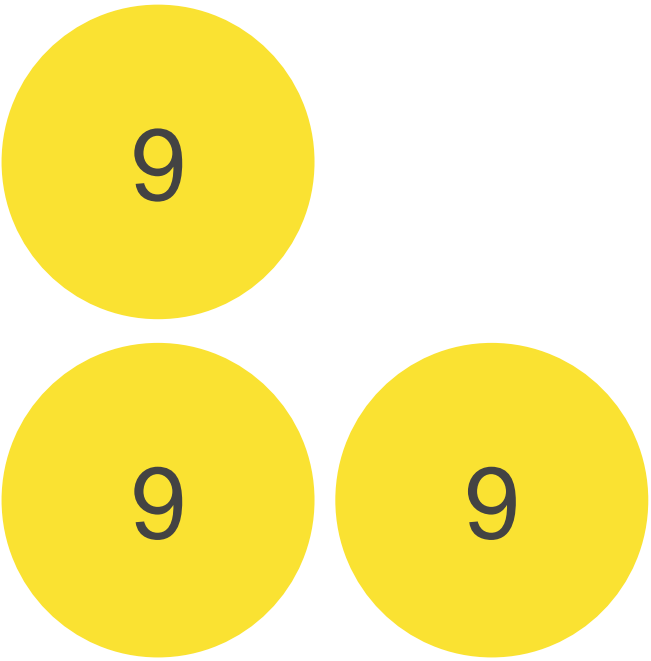
“每次都选尽量大的面值” 就是一个贪心思想



零钱兑换：贪心法的反例

coins = [10, 9, 1], amount = 18

正解： $18 - 9 = 9$



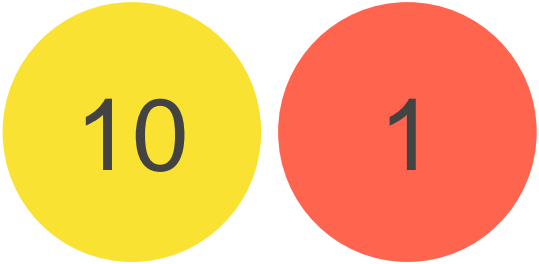
贪心解：

$9 - 9 = 0$

$18 - 10 = 8$



$8 - 1 = 7$



$7 - 1 = 6$



.....

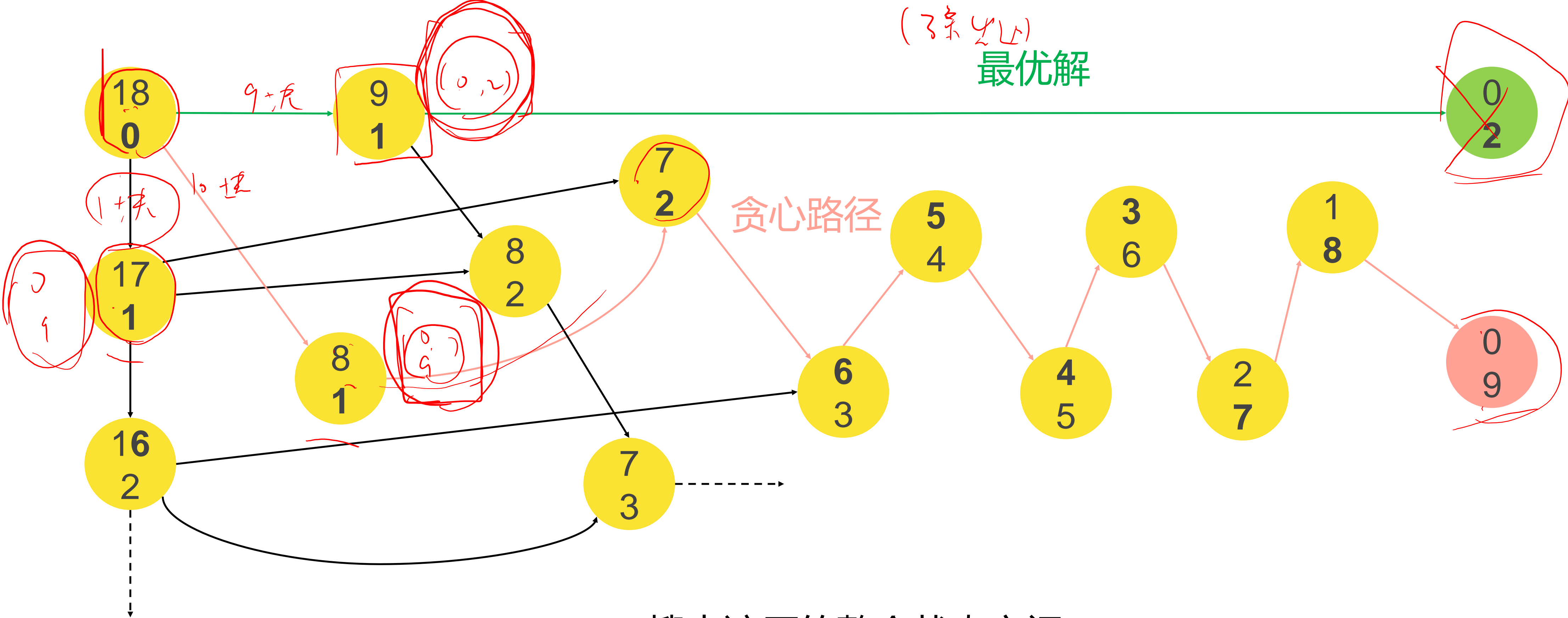
$1 - 1 = 0$



零钱兑换：搜索

状态：剩余金额、已用硬币枚数

决策空间
↑
(3条边)
最优解



搜索遍历的整个状态空间

贪心算法的证明

由“零钱兑换”问题可以看出，贪心算法实际上是在状态空间中按局部最优策略找了一条路径。

贪心算法不一定能得出正确的解，在大部分题目上不能随便使用。

遇到题目，一般先想搜索、动态规划等基于全局的解法，若时间复杂度太高，再考虑贪心。

若要使用贪心算法，必须先证明其正确性。

除了反证法、数学归纳法等大家熟悉的方法外，我们再介绍几种常用的手法。

实战

柠檬水找零

<https://leetcode-cn.com/problems/lemonade-change/description/>

“零钱兑换”类问题在面值互相构成倍数时，贪心算法成立

在本题中，面值为[5, 10, 20]

如果能用1个10块找零，那用2个5块必然也可以

也就是说，做出“用10块找零”这个决策，未来的可能性包含了“用2个5块找零”以后未来的可能性——决策包容性

所以本题可以贪心：优先使用面值较大的找零

实战

分发饼干

<https://leetcode-cn.com/problems/assign-cookies/description/>

两种思路

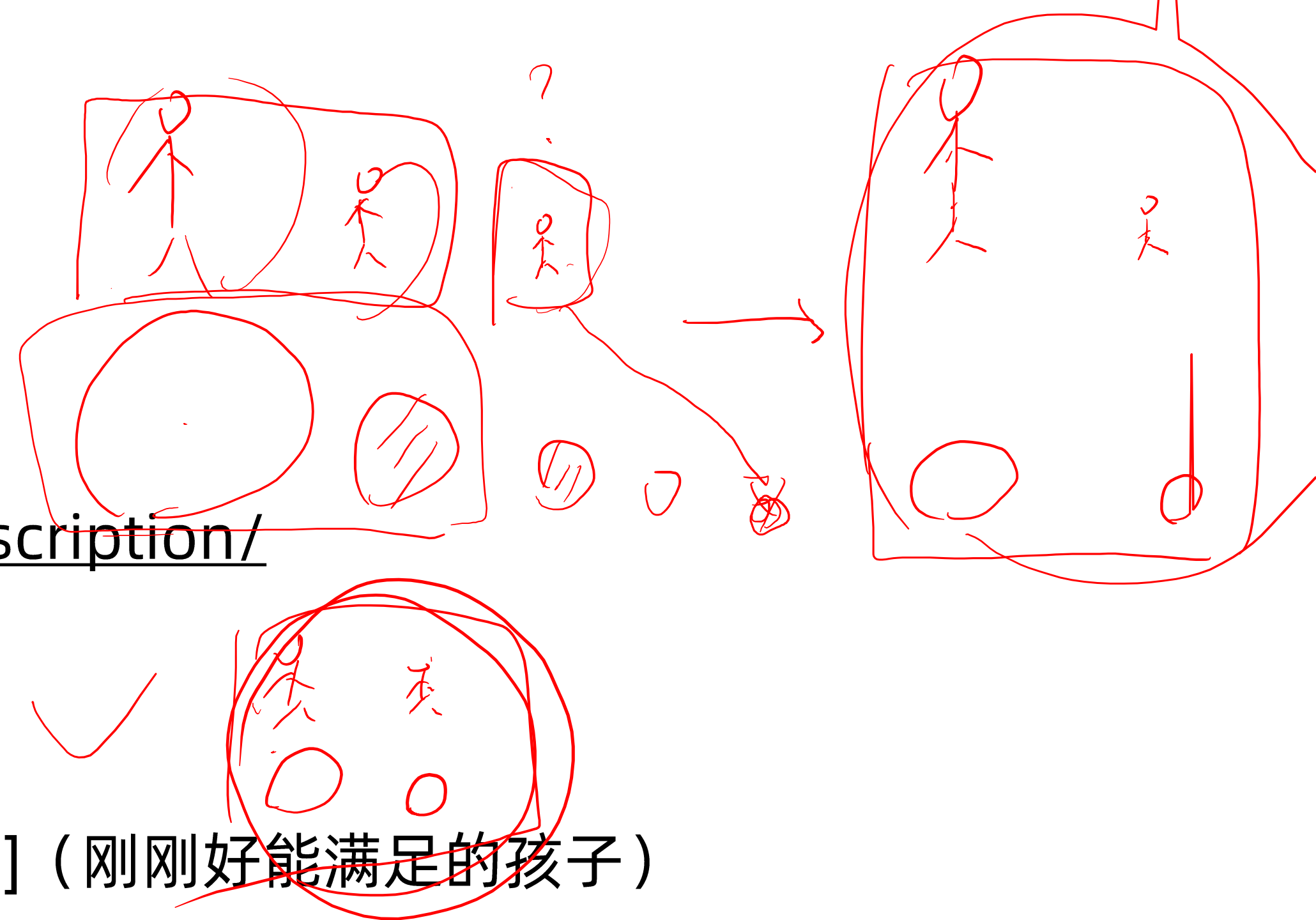
一块饼干（尺寸 $s[j]$ ）发给谁？发给满足 $g[i] \leq s[j]$ 的最大 $g[i]$ （刚刚好能满足的孩子）

一个孩子（胃口 $g[i]$ ）吃哪块饼干？吃满足 $s[j] \geq g[i]$ 的最小 $s[j]$ ，不存在就别吃了

通俗点讲就是小饼干给小孩子，大饼干给大孩子，不能满足就不给了

决策包容性：一块饼干总是想要满足一个孩子的，满足胃口更大的孩子，未来的可能性包含了满足胃口更小孩子可能性

把饼干和孩子排序，代码更容易实现



贪心算法的证明

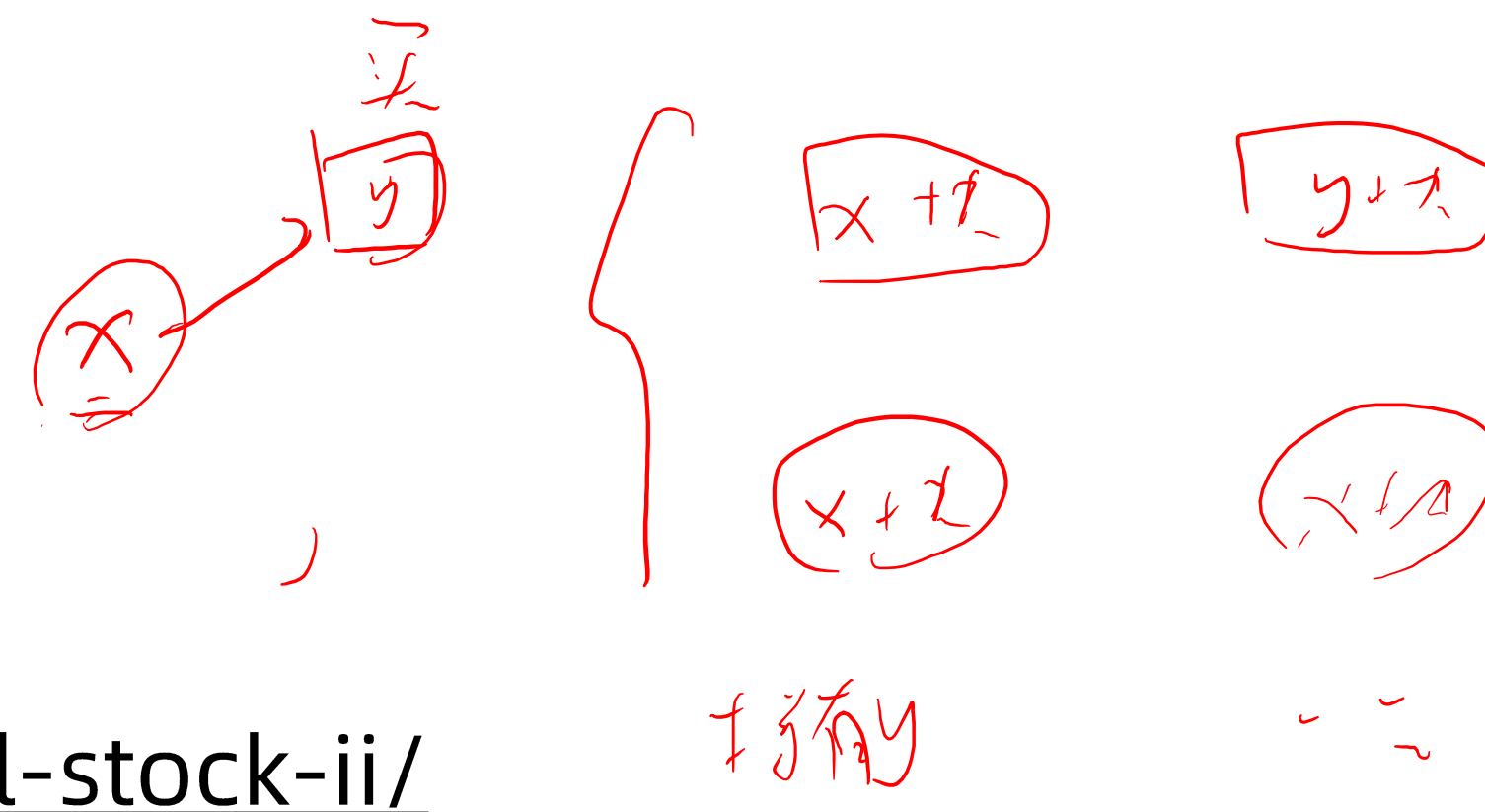
决策范围扩展

在思考贪心算法时，有时候不容易直接证明局部最优决策的正确性
此时可以往后多扩展一步，有助于对当前的决策进行验证

实战

买卖股票的最佳时机 II

<https://leetcode-cn.com/problems/best-time-to-buy-and-sell-stock-ii/>



这是一个预言家的炒股状态——知道未来每天的价格

当前持有股票，卖不卖？往后看一天，明天还涨那肯定不卖，明天跌那肯定卖啊！

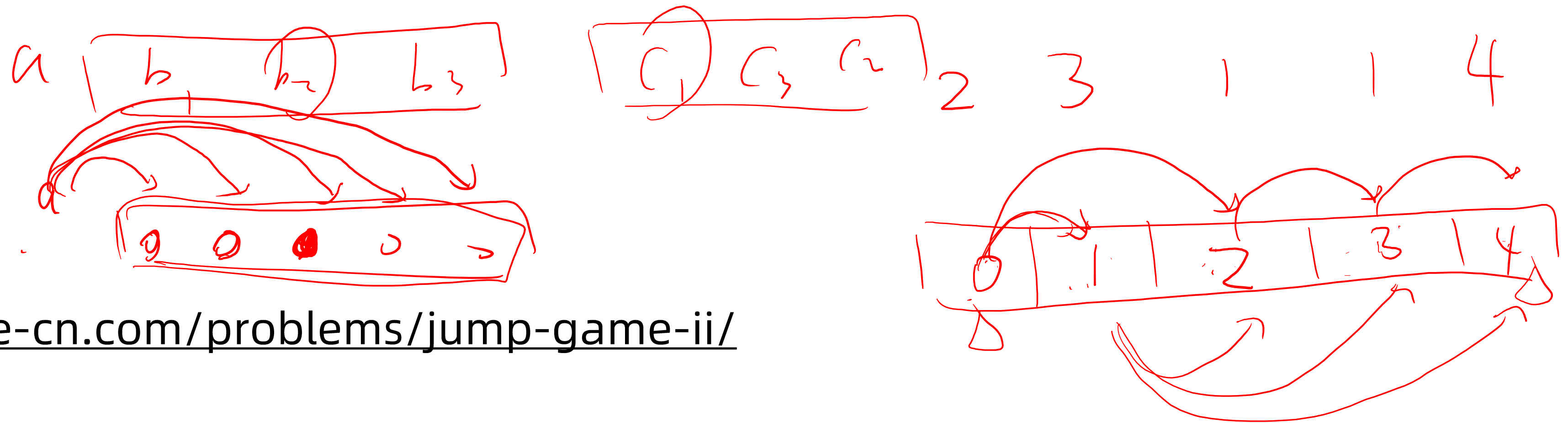
当前没有股票，买不买？往后看一天，明天涨那肯定买，明天跌那肯定先不买！

最后就是完美结果——获得所有 $prices[i] - prices[i - 1] > 0$ 区间的收益

实战

跳跃游戏 II

<https://leetcode-cn.com/problems/jump-game-ii/>



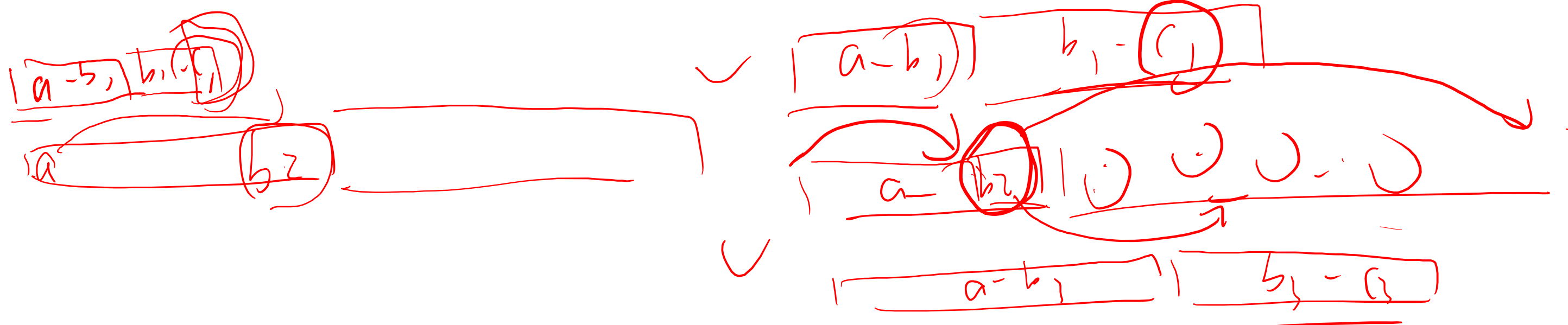
看一下往后跳两步的所有可能性

贪心策略：若 a 能到 b_1, b_2, b_3 ，而 b_1, b_2, b_3 能到的最远位置分别为 c_1, c_2, c_3 ，那应该从 a 跳到 c_1, c_2, c_3 中最大的一个对应的 b 。

Leetcode 示意图



决策包容性：同样都是跳 1 步，从 a 跳到“能跳得更远”的 b ，未来的可达集合包含了跳到其他 b 的可达集合，所以这个局部最优决策是正确的。



贪心算法的证明

邻项交换

经常用于以某种顺序“排序”为贪心策略的证明

证明在任意局面下，任何局部的逆序改变都会造成整体结果变差

实战

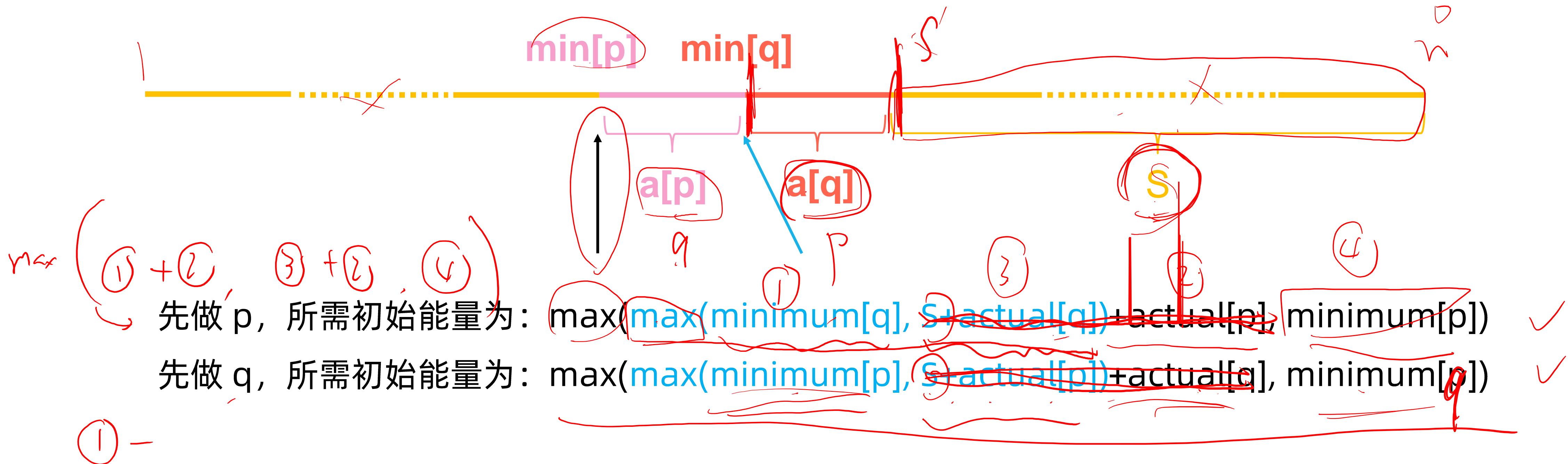


完成所有任务的最少初始能量

<https://leetcode-cn.com/problems/minimum-initial-energy-to-finish-tasks/>

考虑任意一种做任务的顺序，设做完第 $i+2$ 到 n 个任务所需的初始能量最少为 S

对于两个相邻任务：设第 i 个和第 $i+1$ 个完成的任务分别是 p 和 q



实战

完成所有任务的最少初始能量

<https://leetcode-cn.com/problems/minimum-initial-energy-to-finish-tasks/>

① 拆 max

② 消去

③ 比大小

若先做 p 比较优, 则应满足 (拆括号, 消去一样的项)

$$\max(\text{minimum}[q] + \text{actual}[p], \text{minimum}[p]) < \max(\text{minimum}[p] + \text{actual}[q], \text{minimum}[q])$$

因为必定有 $\text{minimum}[q] + \text{actual}[p] > \text{minimum}[q]$

所以上式等价于 $\text{minimum}[q] + \text{actual}[p] < \text{minimum}[p] + \text{actual}[q]$

即 $\text{actual}[p] - \text{minimum}[p] < \text{actual}[q] - \text{minimum}[q]$

贪心策略: 按照 $\text{actual} - \text{minimum}$ 升序排序, 以此顺序完成任务

THANKS

 极客时间 | 训练营