贪心

crazy_cloud

2023 年 7 月 31 日

贪心

- 贪心指一类遵循某种规则，不断选取当前最优策略的办法。
- 贪心法能正确解决问题，要求问题具有最优子结构，即该问题最优解包含其子问题的最优解。
- 当我们设计出一个贪心算法，我们可以试图找反例来测试其正确性。只要反例存在那么算法就是错误的。
- 当然，想要证明贪心算法的正确性，需要一个严格的证明。

与动态规划比较

- 决策：贪心算法只选择当前最优的决策；动态规划则对所有感兴趣的状态的最优解做了记录。
- 最优子结构：两者都有。
- 复杂度：一般而言，贪心算法复杂度远低于动态规划。
- 正确性：贪心需要证明；而动态规划只要状态和转移正确，一定是正确的。

装载问题

- 有 n 个物品，第 i 个物品重量为 w_i，选择尽量多的物品使得总重量不超过 C。

装载问题

- 目标是物品数量尽量多，显然选轻的更划算。
- 将所有物品按 w_i 从小到大排序，我们的策略是选择尽可能长的前缀。

部分背包问题

- 有 n 个物品，第 i 个物品重量为 w_i，价值为 v_i，选择尽量多的物品使得总重量不超过 C。
- 物品可以只取一部分，价值和重量按比例计算。

部分背包问题

- 在装载问题基础上增加了价值。
- 不能先拿轻的，因为它价值可能更小；不能先拿贵的，因为它重量可能也大。
- 按“性价比”即 v_i/w_i 进行贪心
- 性价比高的尽量取，最后一个部分取。
- 如果不允许只取一部分，则会变成真正意义上的部分背包问题，必须使用动态规划之类的其它算法解决。

乘船问题

- n 个人，第 i 个人重量为 w_i，每艘船载重量均为 C，最多乘两个人。
- 求最少几艘船能载所有人。

乘船问题

- 考虑最轻的人 i，若他不能与任何一人同船，那么剩下的所有人只能每人单独一船。
- 否则选尽可能重的和 i 同船。
- 剩下的人同理。

区间问题

- 给定 n 个区间，每个区间左右端点分别为 l_i, r_i，现在要求选出尽量多的区间使得它们两两不相交（不包括端点），问最多能选出几个区间。

区间问题

- 尝试：在可选区间中选左端点最小的。
- 反例：$[1,6],[2,3],[4,5]$。
- 尝试：在可选区间中选长度最小的。
- 反例：$[3,6],[1,5],[5,9]$。

区间问题

- 在可选区间中选右端点最小的。
- 直观：右端点越小，之后可选的区间越多。
- 所有区间按 r_i 排序。
- 去除区间包含的情况（只留小区间），那么剩下的 l_i 也是有序的。
- 不选第一个除了压缩了之后的区间选取的范围以外，没有造成任何好处。所以尽量取前面的。

,

选点问题

- 给定 n 个区间，每个区间左右端点分别为 l_i, r_i。取尽量少的点使得所有区间内部至少有一个点。

选点问题

- 我们同样先去掉区间包含的情况（只留小区间）。
- 考虑从左往右贪心放点，只有不得不放的时候才放点。换句话说，就是放点位置一定在某些 r_i 处。
- 从小到大考虑 r_i，如果当前区间 $[l_i, r_i]$ 没有包括上一个放置的点，则在 r_i 处放一个点。
- 在从左往右考虑的贪心中，如果一个点提前放（不在 r_i 处），将其后移既不会影响当前区间的覆盖情况，还会增加潜在的与后面的区间重合的情况。

顺序问题 I

- 给定 n 个数 a_i，再给出 n 个数 b_i。
- 现在要求你重新排列 b 的顺序，使得 $\sum_{i=1}^{n} a_i \cdot b_i$ 最小。

顺序问题 I

排序不等式

对于两个不下降序列 $a_1, a_2, \ldots, a_n$ 和 $b_1, b_2, \ldots, b_n$，如下不等式成立：

$$\begin{aligned} & a_1 b_1 + a_2 b_2 + \cdots + a_n b_n && (\text{同序}) \\ \geq & a_1 b_{j_1} + a_2 b_{j_2} + \cdots + a_n b_{j_n} && (\text{乱序}) \\ \geq & a_1 b_n + a_2 b_{n-1} + \cdots + a_n b_1 && (\text{逆序}) \end{aligned}$$

顺序问题 I

排序不等式 · 证明

- 这里只展示同序最大的证明，逆序最小只需要取相反数即可。
- 采用调整法和反证法。假设序列 $b_1, b_2, \ldots, b_{i-1}, b_j, \ldots$ 能使得结果最大，其中 i 是首个乱序位置。如果有多个同样大小的取 i 最大的。
- 令 b_i 在该序列中的位置是 k。注意，有 $i < j$ 且 $i < k$。下面我们证明，交换 b_i 和 b_j 能够构造一个结果更大，或者结果相等但是首个乱序位置更大的序列。
- 交换后与交换前的差为

$$(a_i b_i + a_k b_j) - (a_i b_j + a_k b_i) = (a_k - a_i)(b_j - b_i) \geq 0.$$

- 与假设矛盾，故不存在乱序位置，即同序为最大。

顺序问题 II

- 顺序问题中有一类特殊的问题，可以仅仅比较相邻项交换的代价来决定贪心选择。
- 具体来说，当计算发现交换相邻项的代价只由与这两个下标有关的变量，和一个固定常量组成的时候。我们可以直接计算得出最优顺序的排序依据。

顺序问题 II

例 (POJ3045 Cow Acrobats)

- 有 n 头牛，每头牛有一个重量 w_i 和力量 s_i。
- 要求把 n 头牛像叠罗汉一样叠起来。每头牛的风险值就是它顶上的牛的重量之和减去它的力量。
- 一个叠罗汉序列的风险值就是所有牛中最大的风险值。
- 计算风险值最小的序列。

顺序问题 II

例 (POJ3045 Cow Acrobats · 题解)

- 考虑叠罗汉序列相邻的两头牛 i 和 j，后者在上面。令第 j 头牛上面所有牛重量之和为 W。
- 那么 i 风险值是 $W + w_j - s_i$，j 的风险值是 $W - s_j$。如果交换这两头牛的位置，那么 i 的风险值是 $W - s_i$，j 的风险值是 $W + w_i - s_j$。
- 我们只需要比较 $\max\{W + w_j - s_i, W - s_j\}$ 与 $\max\{W - s_i, W + w_i - s_j\}$。
- 注意到 $W + w_j - s_i > W - s_i$ 且 $W + w_i - s_j > W - s_j$，故只需要比较 $W + w_j - s_i$ 与 $W + w_i - s_j$。
- 即 j 在 i 上面风险值不大于交换后当且仅当 $W + w_j - s_i \le W + w_i - s_j$，即 $w_j + s_j \le w_i + s_i$。
- 因此，$w + s$ 越小的越应该排在上面。

字典序最小问题

- 给定长度为 n 的字符串 S，现要构造一个长度为 n 的字符串 T。
- 初始时 T 为空串，随后可以从 S 头部删除一个字符加到 T 的尾部，或是从 S 尾部删除一个字符加到 T 的尾部，求字典序最小的 T。

字典序最小问题

- 由字典序比较方式知，T 越靠前的位置越小越好。
- 操作是从前到后一位位构造 T，因此每次贪心取 S 头尾两个字符中字典序较小那个即可。
- 若头尾字典序一样就继续比较下一位。