习题

1. 在分析插入排序时，一个重要的临界值是，即输入中“反转”的个数（是A[i]左侧且大于它的元素个数）。请给出一个最坏情况下时间复杂度为的算法，从输入中计算M。

【答案】

由于M的本质是计算原始输入数列中当 时 的个数，因此考虑改写归并排序实现。

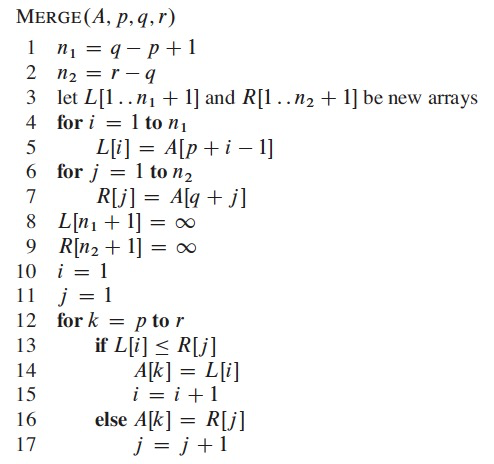
首先，定义“归并反转”为：进行Merge操作，即复制到、到后，、 同时。

由于在归并排序中，数组元素改变位置的唯一方式是在Merge过程中，同时由于合并会使 中的元素保持相同的相对顺序，相应地 中的元素也保持相同的相对顺序，因此两个元素改变其相对顺序的唯一方式就是较大的元素出现在 中，较小的元素出现在 中，因此，每一次“反转”都意味着一次“归并反转”，即它们为一一对应关系。假设我们有一个涉及值 和 的归并反转，其中 原本是 ， 原本是 。由于 位于 中，而 位于 中， 必须位于包含 的子数组之前的子数组中。因此， 一开始位于 的原始位置 之前的位置 ，所以 是一个反转。

所以，设计基于归并排序的反转计数算法如下：

让 成为 中大于 的最小值。在合并过程中的某个时刻， 和 将成为 和 中“暴露”的值，也就是说，我们将在Merge过程的第 13 行中看到 和 。此时，将出现包含 和 ，，，… 的归并反转，而这归并反转将是唯一涉及的归并反转。

因此，我们仅需要检测 和 在合并过程中的首次暴露，并将此时的 加入归并反转的总次数中。



1. 有 n 个螺母和 n 个螺栓，每个螺母正好适合一个螺栓。比较这些螺母和螺栓是否适配的唯一方法是

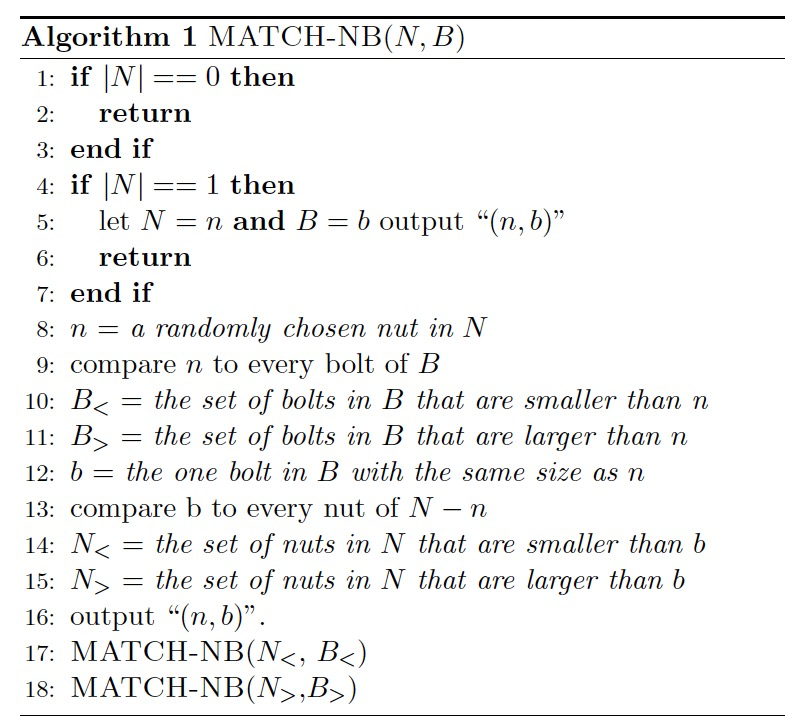
Test(x, y): x是螺母，y是螺栓；返回 "螺母太大"、"螺母太小" 或 "螺母完全合适"。

(a) 设计并分析匹配螺母和螺栓的 O(n2)算法

(b) 为同一问题设计O(nlog n)算法

【答案】

1. 写一个双层for循环嵌套的函数，内外层循环分别为n个螺栓和螺母，利用Test函数对每一个螺栓和筐子都进行比较。因为共要比较n\*n次，所以时间复杂度为
2. 假设螺栓标有数字 1、2、…、n，螺母也是如此。这数字是任意的，与其大小不对应，而只是用于指代算法描述中的螺栓和螺母。此外，输出该算法将由 n 个不同的对（i，j）组成，其中螺栓i和螺母j匹配。算法MATCH-NB将代表螺栓和待匹配螺母：N ⊆ 1,… ,n,代表螺栓,B ⊆ 1,…,n,代表螺母。



1. 考虑用最少的硬币找 n 分的问题。假设每个硬币的价值都是整数。
2. 设计一种贪心算法，用 25 分、1 分、5 分和 10 分来找零。证明你的算法产生了最优解。
3. 假设可用硬币的面值是c的幂，即面值为 c0、c1、…、ck，对于某个整数 c > 1 和 k 1，证明贪心算法总能得到最优解。
4. 给出一组贪心算法不能得到最优解的硬币面值，你的集合应包括1分硬币，这样每个n 值都有一个解。
5. 给出一个O(nk)时间复杂度的算法，可以为任意 k 种不同面值的硬币找零（假定其中一枚硬币是1分钱）。

【答案】

1. 每次我们都会挑选面值最大且小于或等于零钱的硬币。剩下的零钱数额会根据硬币的面值减少。对于一个由25分、1分硬币、5分硬币和10分组成的硬币系统，该算法表明25分硬币的数量 ，10分硬币的数量 ，5分硬币的数量为，1分硬币的数量为 。

证明正确性：

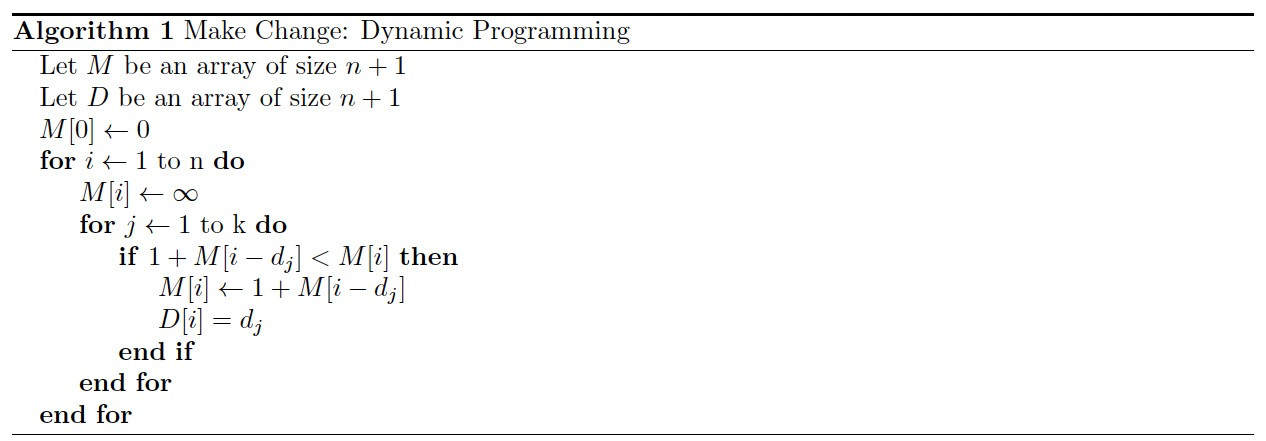
首先证明引理：在最优解中，我们最多可以有1个10分硬币、1个5分硬币和4个1分硬币；最优解中不能同时有2个10分和1个5分硬币。

利用最优策略很容易证明，在最优解中，10分、5分和1分硬币的总价值小于等于24分；5分和1分硬币的总价值小于等于9分；1分硬币的总价值小于等于5分。

基于此引理我们可以证明，假设存在一个使用更少硬币数的最优解OPT，其使用了 个25分硬币， 个10分硬币， 个5分硬币和 个1分硬币。如果，那么最优解OPT重的10分、5分、1分的总价值将大于24分，与引理相悖。按照同样的步骤，可以证明，，。即，贪心算法所给出的硬币数量与OPT相同，为最优解。

1. 假设贪心算法使用了a0枚面值为 c0 的硬币，a1枚面值为 c1 的硬币，......，ak枚面值为 ck 的硬币。假设 OPT 中对应的数字分别为 、、......、。由于我们的贪婪算法尽可能多地使用最大面值的硬币，因此 。如果 ，则 OPT 中其余硬币的总价值将大于ck。在这个硬币系统中，这意味着剩余的硬币中有一些总价值等于ck的硬币。那么我们就可以用一枚价值为ck的硬币来替换这些硬币。因此。按照同样的步骤，我们可以证明。这意味着 OPT 将使用与我们的贪婪算法相同数量的硬币。
2. 反例{1,7,10}
3. 让 表示我们用来找 个硬币的最少硬币数，让表示第 *i* 个面额。那么它的递归形式如下:

算法如下：



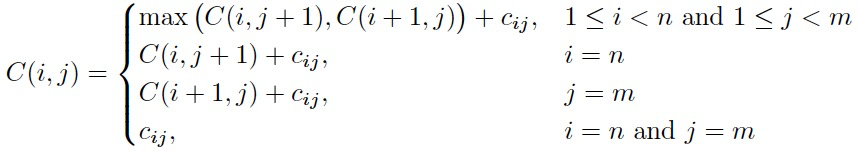
1. 在一个 n×m 的棋盘的格子上放置了不同价值的硬币。假设左上角的单元格为（1，1），右下角的单元格为（n，m）；单元格（i，j）中的硬币价值为 cij。机器人从单元格（0，0）开始，在棋盘上只能向右或向下移动。

(a) 给出一个动态规划算法，用递归方式实现，以确定机器人在棋盘上游荡时应走的路径，从而使收集到的硬币总价值最大化。分析所需时间。

(b) 给出用循环方法实现的动态规划算法。分析所需时间。

【答案】

1. 从 开始，机器人只能移动到 或。在移动之后，机器人会面临一个棋盘略小的子问题，起点分别是 或。这表明问题和子问题之间存在递归关系。如果我们认为 是机器人从 出发到时所能收集到的最大值，那么我们就可以构造出下面的递推关系：



基于递归方式实现的时间复杂度基本上等同于“暴力搜索”我们需要探索所有不同的路径，找出其中的最优路径。也就是说，时间复杂度将与机器人可能走过的路径数量成正比。现在，这变成了一个计数问题 { 机器人将做 n+m-2 次运动，其中 n-1 次是向下运动，m -1 次是向右运动。这意味着将有（或，两者相等）不同的路径可供选择。所以，时间复杂度将为。

1. 使用循环方式进行实现时，我们可以首先构建一个 的表来存储子问题的结果。然后，为了迭代解决问题，我们可以从单元格开始查表，并反向利用上述递归方法，通过迭代表中与已迭代单元格相邻的单元格来查表。迭代任何单元格都需要 O(1) 次，因此这种迭代算法的最终复杂度将与表格中的单元格数成正比，即 。