用动态规划实现导弹拦截 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/646/2021_2022__E7_94_A8_E5_8A_A8_E6_80_81_E8_c97_646773.htm 某国为了防御敌国的导弹袭击，开发出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。 输入导弹依次飞来的高度（雷达给出的高度数据是不大于30000的正整数），计算这套系统最多能拦截多少导弹，并依次输出被拦截的导弹飞来时候的高度。 SAMPLE INPUT: 389 207 155 300 299 170 158 65 SAMPLE OUTPUT: 6 389 300 299 170 158 65 因为只有一套导弹拦截系统，并且这套系统除了第一发炮弹能到达任意高度外，以后的每一发炮弹都不能高于前一发炮弹的高度；所以，被拦截的导弹应该按飞来的高度组成一个非递增序列。题目要求我们计算这套系统最多能拦截的导弹数，并依次输出被拦截导弹的高度，实际上就是要求我们在导弹依次飞来的高度序列中寻找一个最长非递增子序列。 设 X={x 1 ,x 2 ,⋯,x n } 为依次飞来的导弹序列， Y={y 1 ,y 2 ,⋯,y k } 为问题的最优解（即 X 的最长非递增子序列）， s 为问题的状态（表示导弹拦截系统当前发送炮弹能够到达的最大高度，初值为 s=∞ 第一发炮弹能够到达任意的高度）。如果 y 1 =x 1 ，即飞来的第一枚导弹被成功拦截。那么，根据题意“ 每一发炮弹都不能高于前一发的高度 ”，问题的状态将由 s=∞ 变成 s≤x 1 （ x 1 为第一枚导弹的高度）；在当前状态下，序列 Y

1={y 2,⋯,y k} 也应该是序列 X 1={x 2,⋯,x n} 的最长非递增子序列（大家用反证法很容易证明）。也就是说，在当前状态 s≤x 1 下，问题的最优解 Y 所包含的子问题（序列 X 1）的解（序列 Y 1）也是最优的。这就是拦截导弹问题的最优子结构性质。 设 D(i) 为第 i 枚导弹被拦截之后，这套系统最多还能拦截的导弹数（包含被拦截的第 i 枚）。我们可以设想，当系统拦截了第 k 枚导弹 x k，而 x k 又是序列 X={x 1,x 2,⋯,x n} 中的最小值，即第 k 枚导弹为所有飞来的导弹中高度最低的，则有 D(k)=1 ；当系统拦截了最后一枚导弹 x n，那么，系统最多也只能拦截这一枚导弹了，即 D(n)=1 ；其它情况下，也应该有 D(i)≥1。 假设系统最多能拦截的导弹数为 dmax （即问题的最优值），则 dmax = max(D(i)) 所以，要计算问题的最优值 dmax，需要分别计算出 D(1) 、 D(2) 、…… D(n) 的值，然后将它们进行比较，找出其中的最大值。根据上面分析出来的递归方程，我们完全可以设计一个递归函数，采用自顶向下的方法计算 D(i) 的值。然后，对 i 从 1 到 n 分别调用这个递归函数，就可以计算出 D(1) 、 D(2) 、…… D(n) 。但这样将会有大量的子问题被重复计算。比如在调用递归函数计算 D(1) 的时候，可能需要先计算 D(5) 的值；之后在分别调用递归函数计算 D(2) 、 D(3) 、 D(4) 的时候，都有可能需要先计算 D(5) 的值。如此一来，在整个问题的求解过程中， D(5) 可能会被重复计算很多次，从而造成了冗余，降低了程序的效率。 其实，通过以上分析，我们已经知道：D(n)=1 。如果将 n 作为阶段对问题进行划分，根据问题的动态规划递归方程，我们可以采用自底向上的方法依次计算出 D(n-1) 、 D(n-2) 、…… D(1) 的值。这样，每个 D(i) 的值只

计算一次，并在计算的同时把计算结果保存下来，从而避免了有些子问题被重复计算的情况发生，提高了程序的效率。算法代码如下：for(i=n-2.i 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com