Dynamic Programming:

**动态规划**（[英语](http://zh.wikipedia.org/wiki/%E8%8B%B1%E8%AF%AD)：Dynamic programming，DP）[[1]](http://zh.wikipedia.org/wiki/%E5%8A%A8%E6%80%81%E8%A7%84%E5%88%92#cite_note-1)是一种在[数学](http://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%A6)、[计算机科学](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A7%91%E5%AD%A6)和[经济学](http://zh.wikipedia.org/wiki/%E7%BB%8F%E6%B5%8E%E5%AD%A6)中使用的，通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法。 动态规划常常适用于有[重叠子问题](http://zh.wikipedia.org/w/index.php?title=%E9%87%8D%E5%8F%A0%E5%AD%90%E9%97%AE%E9%A2%98&action=edit&redlink=1)[[2]](http://zh.wikipedia.org/wiki/%E5%8A%A8%E6%80%81%E8%A7%84%E5%88%92#cite_note-2)和[最优子结构](http://zh.wikipedia.org/w/index.php?title=%E6%9C%80%E4%BC%98%E5%AD%90%E7%BB%93%E6%9E%84&action=edit&redlink=1)性质的问题，动态规划方法所耗时间往往远少于朴素解法。

动态规划背后的基本思想非常简单。大致上，若要解一个给定问题，我们需要解其不同部分（即子问题），再合并子问题的解以得出原问题的解。 通常许多子问题非常相似，为此动态规划法试图仅仅解决每个子问题一次，从而减少计算量： 一旦某个给定子问题的解已经算出，则将其[记忆化](http://zh.wikipedia.org/w/index.php?title=%E8%AE%B0%E5%BF%86%E5%8C%96&action=edit&redlink=1)存储，以便下次需要同一个子问题解之时直接查表。 这种做法在重复子问题的数目关于输入的规模呈[指数增长](http://zh.wikipedia.org/wiki/%E6%8C%87%E6%95%B8%E5%A2%9E%E9%95%B7)时特别有用。

动态规划在查找有很多**重叠子问题**的情况的最优解时有效。它将问题重新组合成子问题。为了避免多次解决这些子问题，它们的结果都逐渐被计算并被保存，从简单的问题直到整个问题都被解决。因此，动态规划保存[递归](http://zh.wikipedia.org/wiki/%E9%80%92%E5%BD%92)时的结果，因而不会在解决同样的问题时花费时间。

动态规划只能应用于有**最优子结构**的问题。最优子结构的意思是局部最优解能决定全局最优解(对有些问题这个要求并不能完全满足，故有时需要引入一定的近似)。简单地说，问题能够分解成子问题来解决。

1. 最优子结构性质。如果问题的最优解所包含的子问题的解也是最优的，我们就称该问题具有最优子结构性质（即满足最优化原理）。最优子结构性质为动态规划算法解决问题提供了重要线索。(01背包问题)
2. 子问题重叠性质。子问题重叠性质是指在用递归算法自顶向下对问题进行求解时，每次产生的子问题并不总是新问题，有些子问题会被重复计算多次。动态规划算法正是利用了这种子问题的重叠性质，对每一个子问题只计算一次，然后将其计算结果保存在一个表格中，当再次需要计算已经计算过的子问题时，只是在表格中简单地查看一下结果，从而获得较高的效率。(斐波那契问题)

动态规划之01背包问题

<http://blog.csdn.net/mu399/article/details/7722810>