理论

动态规划为算法的重要基础之一。这类算法适用的问题通常有两个特点（或最少有其中一个）：

理论

动态规划为算法的重要基础之一。这类算法适用的问题通常有两个特点（或最少有其中一个）：

- 问题可以分解为较小规划的同类问题

理论

动态规划为算法的重要基础之一。这类算法适用的问题通常有两个特点（或最少有其中一个）：

- 问题可以分解为较小规划的同类问题
- 子问题多次重复出现

理论

动态规划为算法的重要基础之一。这类算法适用的问题通常有两个特点（或最少有其中一个）：

- 问题可以分解为较小规划的同类问题
- 子问题多次重复出现

算法的核心概念是将大问题分成多个小问题，然后解决各个小问题。其中每一个小问题只解决一次。

理论（续）

动态规划的实现有两种方法：

- 从上到下：基本上跟递归很像。不同的地方就是把中间计算的结果都存起来，从而避免多次解决同一个小问题。
- 从下到上：把有可能相关的子问题一个个解决，最后计算出要的结果。

例子一：斐波那契数列

首先，斐波那契数列定义为 $\{F_n\}_{n=1}^{\infty}$，其中

$$F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 3.$$

例子一：斐波那契数列

首先，斐波那契数列定义为 $\{F_n\}_{n=1}^{\infty}$，其中

$$F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 3.$$

如果要计算 F_{41}，那就需要使用上述定义：

$$F_{41} = F_{40} + F_{39}.$$

例子一：斐波那契数列

首先，斐波那契数列定义为 $\{F_n\}_{n=1}^{\infty}$，其中

$$F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 3.$$

如果要计算 F_{41}，那就需要使用上述定义：

$$F_{41} = F_{40} + F_{39}.$$

而要计算 F_{40}，我们则需知道 F_{39} 和 F_{38}。在这里可以看见我们把较大的 n 分割成较小的数字 $(n-1$ 和 $n-2)$。而且小一点的问题多次需要解决。所以这个情况符合使用动态规划的前提。

例子二：最长递增子序列

例子：给出 $\{5, 2, 8, 6, 3, 6, 9, 7\}$。那么最长的递增子序列为：$\{2, 3, 6, 9\}$。
子问题：看前 j 个数字的最长递增子序列。

例子三：找零问题

假定给定的币组为 $T = \{C_1, C_2, \ldots, C_k\}$，而要找零的钱为 X，最小币数的方法为 $W(T, X)$。如果找不了就记成 $W(T, X) = -1$。那我们可以看到下列关系：

例子三：找零问题

假定给定的币组为 $T = \{C_1, C_2, \ldots, C_k\}$，而要找零的钱为 X，最小币数的方法为 $W(T, X)$。如果找不了就记成 $W(T, X) = -1$。那我们可以看到下列关系：

$$W(T, 0) = 0$$

$$W(T, X) = \begin{cases} -1, \text{ 如果 } \{i | W(T, X - C_i) \geq 0, i = 1, \ldots, k\} = \phi \\ 1 + \min\limits_{\substack{i=1,\ldots,k \\ X - C_i \geq 0 \\ W(T, X - C_i) \geq 0}} W(T, X - C_i), \text{ 如不是上述情况} \end{cases}$$

例子三：找零问题

假定给定的币组为 $T = \{C_1, C_2, \ldots, C_k\}$，而要找零的钱为 X，最小币数的方法为 $W(T, X)$。如果找不了就记成 $W(T, X) = -1$。那我们可以看到下列关系：

$$W(T, 0) = 0$$

$$W(T, X) = \begin{cases} -1, \text{ 如果 } \{i | W(T, X - C_i) \geq 0, i = 1, \ldots, k\} = \phi \\ 1 + \min\limits_{\substack{i=1,\ldots,k \\ X - C_i \geq 0 \\ W(T, X - C_i) \geq 0}} W(T, X - C_i), \text{ 如不是上述情况} \end{cases}$$

这里可以看见如何把这个问题分割成较小的问题。

其他例子

- Dijkstra 距离计算方法
- Floyd 距离计算方法
- ...

例子四：Knapsack

假设在一个游戏节目中，参加者被赋予一个能装 W 公斤的袋子，以及 n 类物件，重量为 $w_1, \ldots, w_n$，价值为 $v_1, \ldots, v_n$。参加者应该拿哪些物件让总价值最大化？（这里有两个版本：可重复拿同一类物件、或是不可以）