# 1、动态规划简介

动态规划的英文是**dynamic programming**，我们一般简称dp，需要注意的是，这里面的“programming”指的是一种表格法[1]，而不是编程，后面会具体讨论。动态规划通常用来求解**最优化问题(optimization problem)**。这类问题可以有很多的可行解，每个解都有一个值，我们希望找到具有最优值（最大值或最小值）的解，我们称这样的解为该问题的一个最优解(an optimal solution)，而不是最优解(the optimal solution)，因为可能有多个解都具有最优值。

# 2、动态规划问题描述

- **问题描述：** 一个系统，有若干状态，每个状态下有若干合法的操作，称为决策，决策会改变系统的状态，也会带来收益（或费用）；
- **问题目标：** 在初始状态下，求最终状态下收益或者费用的最优值（一般是总收益最大或总费用最小）；
- **问题特点：** 在每个阶段，选择一些决策，状态会随之改变。但是，收益（或费用）只取决与当前状态和决策，和到达当前状态的路径无关，也就是说和之前的决策无关，这一点称为**无后效性**。在所求的最优解中，总收益（费用）一般指各阶段收益（费用）的总和。

# 3、动态规划步骤

在用动态规划解决问题的时候，需要依次解决以下内容

> - 确定状态集合和收益（或费用）
> - 初始状态、终止状态
> - 确定决策集合
> - 是否无后效，无后效才能使用动态规划
> - 收益（费用）如何表示

这其中，状态的确定是一个难点，因为这个“状态”往往不是常规思维考虑到的，常规思维往往是一个一个枚举各种情况，而状态是对所枚举的一系列情况的概括与抽象，需要一定的思考和经验才能确定一个比较合适的状态选择。所以，动态规划也是枚举，它枚举的是状态，一个状态代表着原来的一类解，是一个解集，然后在这个解集中去选择最优解。

# 4、动态规划与其他算法的比较

## 4.1、与分治比较

动态规划和分治法比较相似，都是通过组合子问题的解来求解原问题。分治法的思路是，将原问题划分为若干互不相交的子问题，然后递归地求解子问题，最后把它们的解组合起来，求出原问题的解[1]。需要注意的是，分治法在递归求解子问题的时候，可能会出现大量子问题重叠的现象，也就是某个子子问题可能是好几个子问题的公共子问题。这种情况下，分治法会做很多重复的计算。与之相反，动态规划对每个子问题只求一次，将其解保存在一个表格中，以打表的方式来避免重复计算。表格记录的是每个状态下解的最优值，需要的时候直接查表即可。这就是第一节解释的**dynamic programming**中的programming。

## 4.2、与贪心比较

动态规划比贪心难，因为贪心在决策的时候，只选择当前最优的一个决策（当然，为什么选这个决策是需要证明的，这是贪心的关键），而动态规划需要枚举到达当前状态的所有决策，从中选择一个最优的。因

此，贪心的正确性需要数学证明，而动态规划由于是枚举，所以自然是正确的。

## 5、动态规划经典例题

上面说的比较抽象，动态规划还是需要多做题去慢慢理解，这类题目也非常多，这里就先举一个入门级的动态规划问题：**数塔**，题目描述点这。

这个题目用暴力方法肯定是不可以的，若塔有n+1层，则枚举的路径条数就是2^n，如果n比较大的话.....所以我们拒绝暴力，倡导和谐（虽然，有些问题暴力是可以ac的）。

换个角度，我们思考一下，从顶点出发时到底向左走还是向右走应取决于是从左走能取到最大值还是从右走能取到最大值，只要左右两道路径上的最大值求出来了就可以作出决策。同样，下一层的走向又要取决于再下一层上的最大值是否已经求出才能决策。这样一层一层推下去，直到倒数第二层时就非常明了了。如数字2，只要选择它下面较大值的结点19前进就可以了。所以实际求解时，可从底层开始，层层递进，最后得到最大值。**自顶向下分析，自底向上计算**[2]。代码参考这里。

## 6、References


- [1]《算法导论》15 动态规划
- [2] HDU ACM程序设计 ppt05动态规划