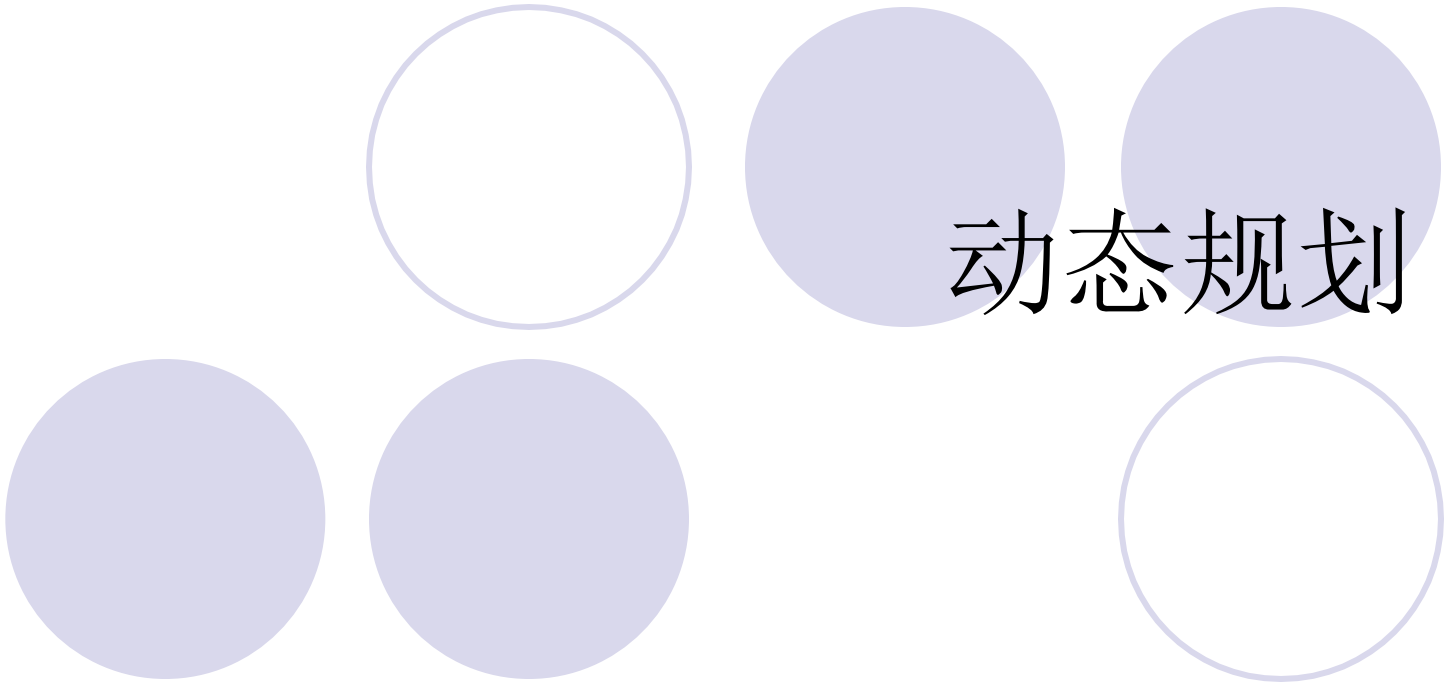

动态规划

什么是动态规划

- 在学习动态规划之前你一定学过搜索.那么搜索与动态规划有什么关系呢?我们来下面的一个例子.

数字三角形

- 给你一个数字三角形, 形式如下：

```
    1
   2  3
  4  5  6
 7  8  9  10
```

找出从第一层到最后一层的一条路,使得所经过的权值之和最小或者最大.

数字三角形

- 无论对与新手还是老手，这都是再熟悉不过的题了，很容易地，我们写出状态转移方程：**f(i, j)=a[i, j] + min{f(i-1, j)+f(i-1, j + 1)}**
- 对于动态规划算法解决这个问题，我们根据状态转移方程和状态转移方向，比较容易地写出动态规划的循环表示方法。但是，当状态和状态转移非常复杂的时候，也许写出循环式的动态规划就不是那么简单了。
- 解决方法：

记忆化搜索

记忆化搜索

- 我们尝试从正面的思路去分析问题，如上例，不难得出一个非常简单的递归过程：
- f1:=f(i-1,j+1); f2:=f(i-1,j);
- if f1>f2 then f:=f1+a[i,j] else f:=f2+a[i,j];
- 显而易见，这个算法就是最简单的搜索算法。时间复杂度为2^n，明显是会超时的。分析一下搜索的过程，实际上，很多调用都是不必要的，也就是把产生过的最优状态，又产生了一次。为了避免浪费，很显然，我们存放一个opt数组：

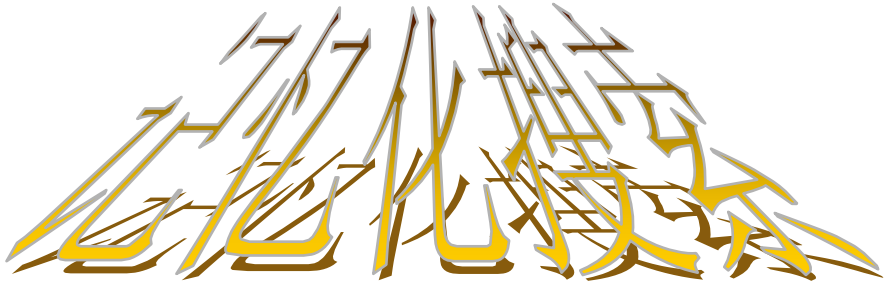
记忆化搜索

- **Opt[i, j] -** 每产生一个**f(i, j)**，将**f(i, j)**的值放入**opt**中，以后再次调用到**f(i, j)**的时候，直接从**opt[i, j]**来取就可以了。
- 于是动态规划的状态转移方程被直观地表示出来了，这样节省了思维的难度，减少了编程的技巧，而运行时间只是相差常数的复杂度，而且在相当多的情况下，递归算法能更好地避免浪费，在比赛中是非常实用的.

记忆化的功效

动态规划的实质

- 可以看出动态规划的实质就是



- 这也就是为什么我们常说动态规划必须满足重叠子问题的原因.记忆化,正符合了这个要求.

状态 阶段 决策

- 或许有一种对动态规划的简单称法,叫分阶段决策.其实我认为这个称法并不是很能让人理解.那么下面我们来看看阶段,状态,决策这三者间得关系吧.

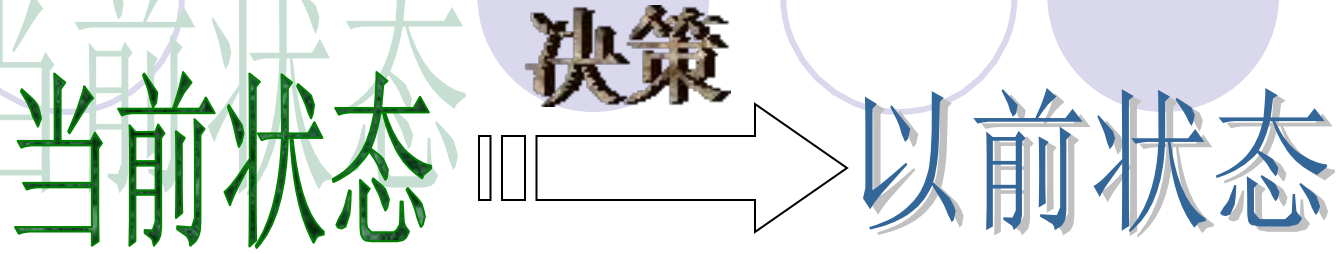
back

状态 阶段 决策

- 状态是表现出动态规划核心思想的一个东西.而分阶段决策这个东西有似乎没有提到状态,这是不科学的.
- 阶段,有些题目并不一定表现出一定的阶段性.数字三角形的阶段就是每一层.这里我们引入一个概念---以前状态.但阶段不是以前状态,状态是阶段的表现形式.数字三角形的以前状态就是当前层的前一层.
- 那什么是决策呢?我们看看下面一张图就知道了.

back

决策



显然, 从上图可以看出, 当前状态通过决策, 回到了以前状态. 可见决策其实就是状态之间的桥梁。而以前状态也就决定了当前状态的情况。

数字三角形的决策就是选择相邻的两个以前状态的最优值。

back

动规的要诀—状态

- 我们一般在动规的时候所用到的一些数组，也就是用来存储每个状态的最优值的。
- 我们就从动态规划的要诀，也就是核心部分“状态”开始，来逐步了解动态规划。

back

一、基本概念

- 动态规划过程是：每次决策依赖于当前状态，又随即引起状态的转移。一个决策序列就是在变化的状态中产生出来的，所以，这种多阶段最优化决策解决问题的过程就称为动态规划。

二、基本思想与策略

- 基本思想与分治法类似，也是将待求解的问题分解为若干个子问题（阶段），按顺序求解子阶段，前一子问题的解，为后一子问题的求解提供了有用的信息。在求解任一子问题时，列出各种可能的局部解，通过决策保留那些有可能达到最优的局部解，丢弃其他局部解。依次解决各子问题，最后一个子问题就是初始问题的解。

- 由于动态规划解决的问题多数有重叠子问题这个特点，为减少重复计算，对每一个子问题只解一次，将其不同阶段的不同状态保存在一个二维数组中。

- 与分治法最大的差别是：适合于用动态规划法求解的问题，经分解后得到的子问题往往不是互相独立的（即下一个子阶段的求解是建立在上一个子阶段的解的基础上，进行进一步的求解）。

三、适用的情况

能采用动态规划求解的问题一般要具有3个性质：

(1) 最优子结构：如果问题的最优解所包含的子问题的解也是最优的，就称该问题具有最优子结构，即满足最优化原理。

(2) 无后效性：即某阶段状态一旦确定，就不受这个状态以后决策的影响。也就是说，某状态以后的过程不会影响以前的状态，只与当前状态有关。

（3）有重叠子问题：即子问题之间是不独立的，一个子问题在下一阶段决策中可能被多次使用到。（该性质并不是动态规划适用的必要条件，但是如果没有这条性质，动态规划算法同其他算法相比就不具备优势）

最优子结构

- 最短路径：如果某条路径是起点到终点的最短路径，则起点到该路径上的每一点都是最短路径。（最长路径？）

无后效性

- 最短路径：当前选择任何一条边都不会影响以后选择其他边。
- 有费用的最短路径：设经过任何一条边时都要耗费一定的费用，总费用一定的情况下，当前选择某一条特定的边可能导致某些其他的边无法被选择。

四、求解的基本步骤

- 动态规划所处理的问题是一个多阶段决策问题，一般由初始状态开始，通过对中间阶段决策的选择，达到结束状态。这些决策形成了一个决策序列，同时确定了完成整个过程的一条活动路线(通常是求最优的活动路线)。如图所示。动态规划的设计都有着一定的模式，一般要经历以下几个步骤。