# 贪心算法：

（prime，，迪杰斯特拉，克鲁斯卡尔）

贪心算法是指在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，只做出在某种意义上的局部最优解。贪心算法不是对所有问题都能得到整体最优解，关键是贪心策略的选择（这个要自己想哪种最大小的可以得到整体最优解如何找到局部最优解），选择的贪心策略必须具备无后效性，即某个状态以前的过程不会影响以后的状态，只与当前状态有关。

虽然贪心算法不能对所有问题都得到整体最优解，但对许多问题它能产生整体最优解。所以就要看这个问题是怎样的

结构：

候选方案（CandidateSet）有一套可以解决问题的候选方案

选择函数（SelectionFunction）选择最优候选方案

可行性函数（FeasibilityFunction）确定是否可用某个候选方案

目标函数（ObjectiveFunction）用于判定整体的或者某个阶段的结果好坏

结果函数（SolutionFunction）用于结束

# 动态规划

主要是用动态规划表抽象出实际问题

 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）。

**（1）动态规划和分治区别：**

动态规划[**算法**](http://lib.csdn.net/base/datastructure)：它通常用于求解具有某种最优性质的问题。在这类问题中，可能会有许多可行解。每一个解都对应于一个值，我们希望找到具有[最优值](http://baike.baidu.com/view/1009693.htm)的解。动态规划算法与[分治法](http://baike.baidu.com/view/1583824.htm)类似，其基本思想也是将待求解问题分解成若干个子问题，先求解子问题，然后从[这些子](http://baike.baidu.com/view/7978438.htm)问题的解得到原问题的解。**与分治法不同的是，适合于用动态规划求解的问题，经分解得到子问题往往不是互相独立的。**

分治法：若用分治法来解这类问题，则分解得到的子问题数目太多，有些子问题被重复计算了很多次。如果我们能够保存已解决的子问题的答案，而在需要时再找出已求得的答案，这样就可以避免大量的重复计算，节省时间。我们可以用一个表来记录所有已解的子问题的答案。

注：**不管该子问题以后是否被用到，只要它被计算过，就将其结果填入表中。**这就是动态规划法的基本思路。

能采用动态规划求解的问题的一般要具有3个性质：

    (1) 最优化原理：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

**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。

    初始状态→│决策１│→│决策２│→…→│决策ｎ│→结束状态

                      图1 动态规划决策过程示意图

    (1)**划分阶段**：按照问题的时间或空间特征，把问题分为若干个阶段。在划分阶段时，注意划分后的阶段一定要是有序的或者是可排序的，否则问题就无法求解。

    (2)**确定状态和状态变量**：将问题发展到各个阶段时所处于的各种客观情况用不同的状态表示出来。当然，状态的选择要满足无后效性。

    (3)**确定决策并写出状态转移方程**：因为决策和状态转移有着天然的联系，状态转移就是根据上一阶段的状态和决策来导出本阶段的状态。所以如果确定了决策，状态转移方程也就可写出。但事实上常常是反过来做，根据相邻两个阶段的状态之间的关系来确定决策方法和状态转移方程。

    (4)**寻找边界条件**：给出的状态转移方程是一个递推式，需要一个递推的终止条件或边界条件。

    一般，只要解决问题的阶段、状态和状态转移决策确定了，就可以写出状态转移方程（包括边界条件）。

实际应用中可以按以下几个简化的步骤进行设计：

    （1）分析最优解的性质，并刻画其结构特征。

    （2）递归的定义最优解。

    （3）以自底向上或自顶向下的记忆化方式（备忘录法）计算出最优值

    （4）根据计算最优值时得到的信息，构造问题的最优解

**五、算法实现的说明**

**确定子结构**

    动态规划的主要难点在于理论上的设计，也就是上面4个步骤的确定，一旦设计完成，实现部分就会非常简单。

     使用动态规划求解问题，最重要的就是确定动态规划三要素：

    （1）问题的阶段 （2）每个阶段的状态

    （3）从前一个阶段转化到后一个阶段之间的递推关系。

     递推关系必须是从次小的问题开始到较大的问题之间的转化，从这个角度来说，动态规划往往可以用递归程序来实现，不过因为递推可以充分利用前面保存的子问题的解来减少重复计算，所以对于大规模问题来说，有递归不可比拟的优势，这也是动态规划算法的核心之处。

    确定了动态规划的这三要素，整个求解过程就可以用一个最优决策表来描述，最优决策表是一个二维表，其中行表示决策的阶段，列表示问题状态，表格需要填写的数据一般对应此问题的在某个阶段某个状态下的最优值（如最短路径，最长公共子序列，最大价值等），填表的过程就是根据递推关系，从1行1列开始，以行或者列优先的顺序，依次填写表格，最后根据整个表格的数据通过简单的取舍或者运算求得问题的最优解。

**f(n,m)=max{f(n-1,m), f(n-1,m-w[n])+P(n,m)}**

这种由后向前的递推方法，正是动态规划的寻优思想。

# 分治

2、分治法的适用条件   
分治法所能解决的问题一般具有以下几个特征：   
（1）该问题的规模缩小到一定的程度就可以容易地解决；   
（2）该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质；   
（3）利用该问题分解出的子问题的解可以合并为该问题的解；   
（4）该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子子问题。

上述的第一条特征是绝大多数问题都可以满足的，因为问题的计算复杂性一般是随着问题规模的增加而增加；第二条特征是应用分治法的前提，它也是大多数问题可 以满足的，此特征反映了递归思想的应用；第三条特征是关键，能否利用分治法完全取决于问题是否具有第三条特征，如果具备了第一条和第二条特征，而不具备第 三条特征，则可以考虑贪心法或动态规划法。第四条特征涉及到分治法的效率，如果各子问题是不独立的，则分治法要做许多不必要的工作，重复地解公共的子问 题，此时虽然可用分治法，但一般用动态规划法较好。

# 递归

# 回溯

# 贪心与动态规划

最优子结构性质：当一个问题的最优解包含子问题的最优解时，称此问题具有最优子结构性质

相同：两者所解决的问题都具有最优子结构性质

不同：动态规划法：具有互相重叠的子问题