# 贪心算法&动态规划

1.某天早上公司领导找你解决一个问题，明天公司有N个同等级的会议需要使用同一个会议室，现在给你这个N个会议的开始和结束

时间，你怎么样安排才能使会议室最大利用？即安排最多场次的会议？电影的话 那肯定是最多加票价最高的，入场率。综合算法

2.双十一马上就要来了，小C心目中的女神在购物车加了N个东西，突然她中了一个奖可以清空购物车5000元的东西（不能找零），每个东西只能买一件，那么她应该如何选择物品使之中奖的额度能最大利用呢？如果存在多种最优组合你只需要给出一种即可，嘿嘿 现在女神来问你，你该怎么办？

2000 3000 4000

1500 2000 4000

4000 2000 1500

4000 3000 2000

概念：贪心算法又叫做贪婪算法，它在求解某个问题时，总是做出眼前最大利益。

也就是说只顾眼前不顾大局，所以它是局部最优解。核心点：通过局部最优推出全局最优。

1.某天早上公司领导找你解决一个问题，明天公司有N个同等级的会议需要使用同一个会议室，现在给你这个N个会议的开始和结束时间，你怎么样安排才能使会议室最大利用？即安排最多场次的会议？

现在我们怎么去贪？也就这个我们选择的贪心策略：

选时间最短：1-3，2~4,3~5,4~6

按结束时间从小到大排序：首先把第一个加入我们可以开会的列表。之后只要开始时间是大于我们上一个的结束时间的就可以开

咱们来看思考题1：

0点~9点：9点之前开始的会议都不行了。

8点~10点

10点~12点：12点

8点~20点

思路：

按结束时间从小到大排序：首先把第一个加入我们可以开会的列表。之后只要开始时间是大于我们上一个的结束时间的就可以开

1~3：3

2~4

3~5

4~6

贪心算法的套路：一定会有一个排序。哈夫曼编码，贪心算法，压缩算法。最短路径

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，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。

贪心算法其最重要的两个点就是：

贪心策略：排序

通过局部最优解能够得到全局最优解

一般通过以下问题就可以通过贪心算法解决：

1.针对某个问题有限制值，以及有一个期望的最好结果，通常是从某些数据中选出其中一些，达到最好的结果。

2.一般会有一个排序，找出贡献最大的。

3.举例看贪心是否可以解决。

一般用在任务调度，教师排课等系统。

实际上，用贪心算法解决问题的思路，并不总能给出最优解，比如来看下一个问题：

经典问题：背包问题

小偷去某商店盗窃，背有一个背包，容量是50kg，现在有以下物品（物品不能切分,且只有一个），请问小偷应该怎么拿才能得到最大的价值？

重量 价值

物品1 10kg 60元 60 / 10 = 6

物品2 20kg 100元 100/20 = 5

物品3 40kg 120元 120/40 = 3

性价比最高：贪心的策略，按性价比排序,得到的最大价值是 60+100=160，背包装了30kg

很显然：40+10（kg）=120+60=180

遍历它：每个物品只有2个选择就是拿与不拿吧，我们就用枚举，排列组合；

3：000 111 011 010 001 100 101 110 ，有没有道理. 10个物品 有多少排列组合？10! 20!,10一下是完美的，32！

如果用贪心可以解决问题吗？

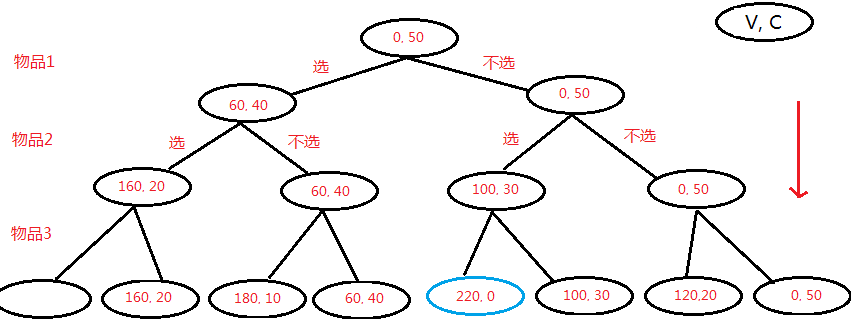
不能,如果从解决现实种问题的来说，你就按你自己的的想法拿。

通过分析，上面的背包问题通过贪心是解决不了的，很显然如果我们每次拿性价比最大的那么就只有160，而最大应该是220，这就是贪心算法的局限性，那么这个问题该如何解决呢？其实就是我们经常听到的动态规划：

价值：60 100 120

重量：10 20 30

背包是50kg：



一般用动态规划可以解决的问题：

1.局部最优解：也就是它会有一个最优子结构

2.子问题可以重复

3.状态转移方程：通过把问题分成很多小阶段一段段的转移。从而得出最优解.状态转移方程是解决动态规划的关键。如果我们能写出状态转移方程，那动态规划问题基本上就解决一大半了，翻译成代码非常简单。但是很多动态规划问题的状态本身就不好定义，状态转移方程也就更不好想到。

1.求解最值问题：最短路径经典算法

2.短字符相似性匹配：相似度、编辑距离（动态规划）两个字符串最长公共子串

3.策略问题:

4.哈夫曼编码