贪心算法

灯神说“你可以许3个愿望”…

七月在线 林应

2018年3月

主要内容

- 简介、思路和特性
- 实现要点
- 经典问题分析
- 作业

简介、思路和特性

- 简介
  - 贪心算法（又称贪婪算法）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，他所做出的是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，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。

简介、思路和特性

- 基本要素（关于最优解）
  - 贪心选择：指所求问题的整体最优解可以通过一系列局部最优的选择，即贪心选择来达到。这是贪心算法可行的第一个基本要素，也是贪心算法与动态规划算法的主要区别。贪心选择是采用从顶向下、以迭代的方法做出相继选择，每做一次贪心选择就将所求问题简化为一个规模更小的子问题。对于一个具体问题，要确定它是否具有贪心选择的性质，我们必须证明每一步所作的贪心选择最终能得到问题的最优解。

简介、思路和特性

- 基本要素（关于最优解）
  - 最优子结构：当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。运用贪心策略在每一次转化时都取得了最优解。问题的最优子结构性质是该问题可用贪心算法或动态规划算法求解的关键特征。贪心算法的每一次操作都对结果产生直接影响，而动态规划则不是。贪心算法对每个子问题的解决方案都做出选择，不能回退；动态规划则会根据以前的选择结果对当前进行选择，有回退功能。

简介、思路和特性

- 基本思路
  - 贪心算法的基本思路是从问题的某一个初始解出发一步一步地进行，根据某个优化测度，每一步都要确保能获得局部最优解。每一步只考虑一个数据，他的选取应该满足局部优化的条件。若下一个数据和部分最优解连在一起不再是可行解时，就不把该数据添加到部分解中，直到把所有数据枚举完，或者不能再添加算法停止。

简介、思路和特性

- 实现过程
  - 建立数学模型来描述问题
  - 把求解的问题分成若干个子问题
  - 对每一子问题求解，得到子问题的局部最优解
  - 把子问题的解局部最优解合成原来解问题的一个解

实现要点

1. 随着算法的进行，将积累起其它两个集合：一个包含已经被考虑过并被选出的候选对象，另一个包含已经被考虑过但被丢弃的候选对象。
2. 有一个函数来检查一个候选对象的集合是否提供了问题的解答。该函数不考虑此时的解决方法是否最优。

实现要点

3. 还有一个函数检查是否一个候选对象的集合是可行的，也即是否可能往该集合上添加更多的候选对象以获得一个解。和上一个函数一样，此时不考虑解决方法的最优性。
4. 选择函数可以指出哪一个剩余的候选对象最有希望构成问题的解。
5. 最后，目标函数给出解的值。

实现要点

6. 为了解决问题，需要寻找一个构成解的候选对象集合，它可以优化目标函数，贪婪算法一步一步的进行。起初，算法选出的候选对象的集合为空。接下来的每一步中，根据选择函数，算法从剩余候选对象中选出最有希望构成解的对象。如果集合中加上该对象后不可行，那么该对象就被丢弃并不再考虑；否则就加到集合里。每一次都扩充集合，并检查该集合是否构成解。

经典问题分析

- 背包问题
  - 有一个背包，背包容量是M=80kg。有7个物品，物品不可以分割成任意大小。要求尽可能让装入背包中的物品总价值最大，但不能超过总容量。
  - 物品信息
    - A(8KG/15$), B(35KG/40$), C(16KG/50$)
    - D(50KG/40$), E(40KG/35$), F(12KG/40$)
    - G(25KG/30$)

经典问题分析

- 背包问题
  - 根据贪心的策略，每次挑选价值最大的物品装入背包，得到的结果是否最优？
  - 选择： C(16KG/50$)，F(12KG/40$)，B(35KG/40$)
  - 总计：63KG/130$
  - 反例：把B换成A(8KG/15$)和E(40KG/35$)能装76KG/140$。

经典问题分析

- 背包问题
  - 每次挑选所占重量最小的物品装入是否能得到最优解？
  - 选择：A(8KG/15$)，F(12KG/40$)，C(16KG/50$)，G(25KG/30$)
  - 总计：61KG/138$
  - 反例：参考第一个方法

经典问题分析

- 背包问题
  - 每次选取单位重量价值最大的物品呢?
  - 选择：A(8KG/15$)，F(12KG/40$)，C(16KG/50$)，G(25KG/30$)
  - 总计：66KG/135$
  - 反例：参考第一个方法
- 结论：贪心法解决不了，必须依靠动态规划。

经典问题分析

- 最小生成树的Prim算法
  - 输入：一个加权连通图，其中顶点集合为V，边集合为E。
  - 初始化：Vnew = {x}，其中x为集合V中的任一节点（起始点），Enew = {}，为空。

经典问题分析

- 最小生成树的Prim算法
  - 重复以下操作（while循环）直到Vnew = V
    - 在集合E中选取权值最小的边<u, v>，其中u为集合Vnew中的元素，而v不在Vnew集合当中，并且v∈V（如果存在有多条满足前述条件即具有相同权值的边，则可任意选取其中之一）；
    - 将v加入集合Vnew中，将<u, v>边加入集合Enew中。
  - 输出：使用集合Vnew和Enew来描述所得到的最小生成树。

经典问题分析

- 跳跃游戏：给出一个非负整数数组，你最初定位在数组的第一个位置。数组中的每个元素代表你在那个位置可以跳跃的**最大**长度。判断你是否能到达数组的最后一个位置。
  - A = [2,3,1,1,4]，返回 true。
  - A = [3,2,1,0,4]，返回 false。
  - 思路：从终点前一点倒推，如果该点能到终点，剩下的问题就是从出发点能不能到该点。

经典问题分析

- 加油站：在一条环路上有N个加油站，其中第i个加油站有汽油gas[i]，并且从第_i_个加油站前往第_i_+1个加油站需要消耗汽油cost[i]。你有一辆油箱容量无限大的汽车，现在要从某一个加油站出发绕环路一周，一开始油箱为空。求可环绕环路一周时出发的加油站的编号，若不存在环绕一周的方案，则返回-1。

经典问题分析

- 加油站：如何贪心
  - 其实题目里隐含了一个很重要的推论：如果在a[i]停车加满油仍然开不到a[j]的话，那么即使在a[i]不停，在a[i+k]加满油也到不了a[j]。因为即使a[i]停了，a[i + k]还是可以停的。

经典问题分析

- 合并数字
  - 给出n个数，现在要将这n个数合并成一个数，每次只能选择两个数a,b合并，每次合并需要消耗a+b的能量，输出将这n个数合并成一个数后消耗的最小能量。

经典问题分析

- 合并数字
  - 给出一个序列[a, b, c, d]假如我们随机合并，假设任意两数和都大于已有的[a, b, c, d]
    - 合并a, b，[c, d, a+ b]，消耗能量a+b
    - 合并d, a+b, [c, a+b+d]，消耗能量a+b+d
    - 合并剩余数字，消耗能量a+b+c+d
    - 总消耗能量：a*3 + b*3 + c + d*2

经典问题分析

- 合并数字
  - 按从小到大的顺序合并：
    - 合并a, b，[c, d, a+ b]，消耗能量a+b
    - 合并c, d，[a+b, c+d]，消耗能量c+d
    - 合并剩余数字，消耗能量a+b+c+d
    - 总消耗能量：a*2+b*2+c*2+d*2，比上一种方式的消耗要小。

经典问题分析

单调递增的数字

- 给一非负整数N, 找到小于等于N的最大的单调递增数. (回想一下, 当且仅当每对相邻的数字x和y满足x <= y时, 这个整数才是单调递增数)
- 输入17699 -> 16999
- 输入12468 -> 12468
- 输入1000 -> 999
- 输入2571001 -> 2569999

经典问题分析

- 硬币排成线
  - 有 n 个硬币排成一条线。两个参赛者轮流从右边依次拿走 1 或 2 个硬币，直到没有硬币为止。拿到最后一枚硬币的人获胜。
  - 思路：
    - 如果A要拿走最后一个硬币，必须迫使B在前一步拿的时候只有3个硬币，无论拿1个还是2个都是输。
    - 为了迫使B面对3个币的局面，上一轮必须迫使B面对6个币。。。

作业

- 买卖股票的最佳时机 II
  - http://lintcode.com/zh-cn/problem/best-time-to-buy-and-sell-stock-ii/
  - 假设有一个数组，它的第i个元素是一个给定的股票在第i天的价格。设计一个算法来找到最大的利润。你可以完成尽可能多的交易(多次买卖股票)。然而,你不能同时参与多个交易(你必须在再次购买前出售股票)。

作业

- 删除数字
  - http://lintcode.com/zh-cn/problem/delete-digits/
  - 给出一个字符串A, 表示一个n位正整数, 删除其中k位数字, 使得剩余的数字仍然按照原来的顺序排列产生一个新的正整数。
  - 找到删除k个数字之后的最小正整数。

感谢大家！

恳请大家批评指正！