OpenSource Computing

1. 记录优秀的技术细节
2. 谈谈对于科技走势的想法
3. 共享学术研究的成果
4. 增进同行的交流和互识

Welcome~~~

私信 归档 RSS

06

05

贪心算法小结

贪心算法是一个快速的、不稳定的算法。贪心算法总是作出在当前看来最好的选择。也就是说贪心算法并不从整体最优考虑，它所作出的选择只是在某种意义上的局部最优选择。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。如单源最短路经问题，最小生成树问题等。在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的近似（具有不稳定性）。当“贪心序列”中的每项互异且当问题没有重叠性时，看起来总能通过贪心算法取得（近似）最优解的。

贪心算法一般是求“最优解”这类问题的。最优解问题可描述为：有n个输入，它的解是由这n个输入的某个子集组成，并且这个子集必须满足事先给定的条件。这个条件称为约束条件。而把满足约束条件的子集称为该问题的可行解。这些可行解可能有多个。为了衡量可行解的优劣，事先给了一个关于可行解的函数，称为目标函数。目标函数最大（或最小）的可行解，称为最优解。

a)求“最优解”最原始的方法为搜索枚举方案法（一般为回溯法）

除了极简单的问题，一般用深度优先搜索或宽度优先搜索。通常优化方法为利用约束条件进行可行性判断剪枝；或利用目标函数下界（或上界），根据当前最优解进行分枝定界。

b)动态规划（需要满足阶段无后效性原则）。

动态规划主要是利用最最优子问题的确定性，从后向前（即从小规模向大规模）得到当前最优策略，从而避免了重复的搜索。

一、贪心问题一般性质

1、贪心选择性质

所谓贪心选择性质是指所求问题的整体最优解可以通过一
心选择来达到。这是贪心算法可行的第一个基本要素，也
的主要区别。

动态规划算法通常以自底向上的方式解各子问题，而贪心
式进行，以迭代的方式作出相继的贪心选择，每作一次贪
规模更小的子问题。对于一个具体问题，要确定它是否具
每一步所作的贪心选择最终导致问题的整体最优解。

在动态规划中，每一个父问题结果的得出需要它的子问题作为条件；但是“贪心选择性”则不需要；贪心选择性所做的是一个非线性的子问题处理过程，即一个子问题并不依赖于另一个子问题，但是子问题间有严格的顺序性。

该算法存在问题：

1、不能保证求得的最后解是最佳的；

2、不能用来求最大或最小解问题；

3、只能求满足某些约束条件的可行解的范围。

2、最优子结构性质

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。问题的最优子结构性质是该问题可用动态规划算法或贪心算法求解的关键特征。不能采用分治法解决的问题，是理论上是不能使用贪心算法的，而且，必须拥有最优子结构性质，才能保证贪心算法返回最优解。

二、贪心算法与回溯法及动态规划算法的差异

我们以例子来说明这个问题。

1、最短路径

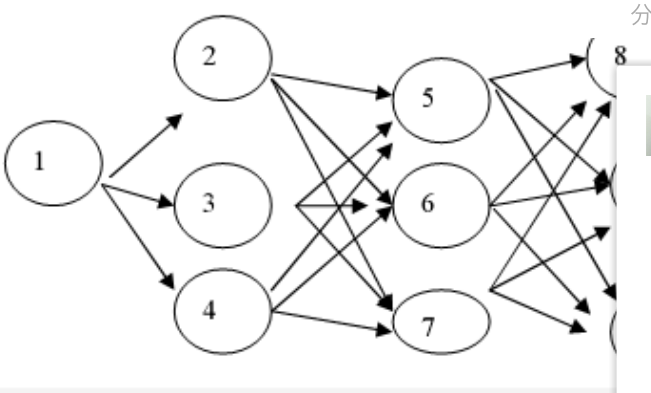

图 1

如图所示，我们如果用回溯法，则会搜索过程中，会产生如下搜索树：

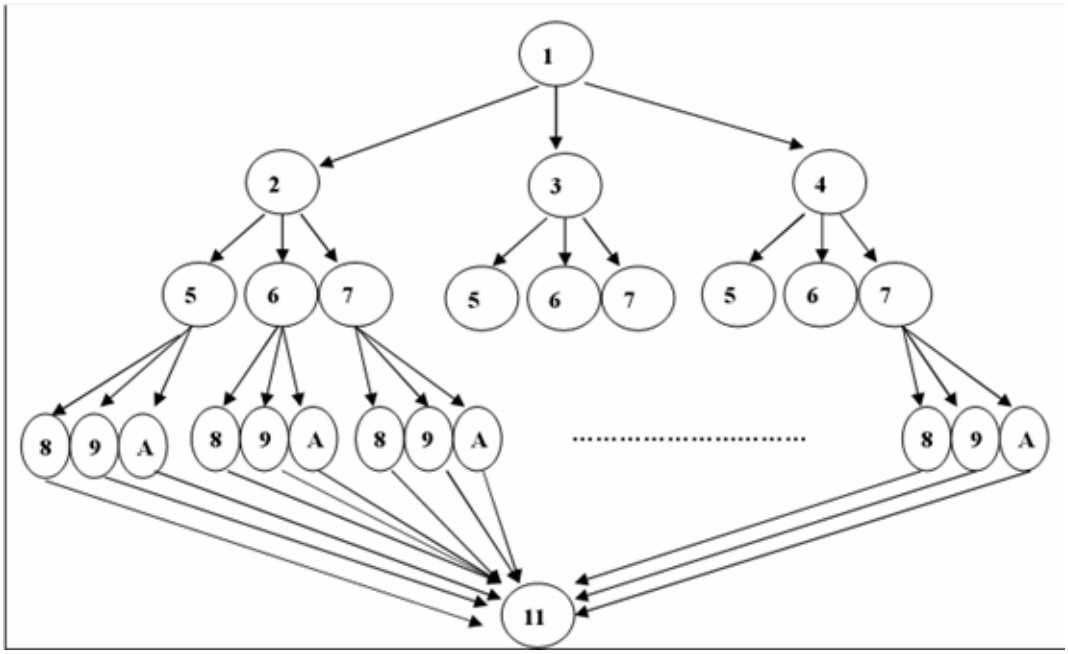

图2

显然，上面的搜索有大量重复性工作。比如节点8、9、10到11的最短路分别被调用了9次，从节点5、6、7到节点11也分别搜索了3次。 如果先算出节点8、9、10到11的最短路，由于它与前面的点无关，因此最优值确定下来，再用它们求定节点5、6、7到节点11的最短路径。同理，再用节点5、6、7的最优值，来求节点2、3、4优值。最后从节点2、3、4推出1到11的最优值，显然复杂度大为降低。

当然，如果本题把简单搜索改为搜索＋记忆化的方法，则就是得能动态规划的原理，本质上就是动态规划，只是实现的方法不同与传统的表格操作法。

贪心算法则不同，它不是建立在枚举方案的基础上的。它从前向后，根据当前情况，“贪心地”决定出下一步，从而一步一步直接走下去，最终得到解。假如本例子中，我们定下这样的贪心策略：节点号k%3= =1。显然，它只访问了节点1、4、7、

10、11，工作量最少，效率最高。当然，对本题来说 路径。 从图3中可以看出，贪心算法是一种比动态规划更高效的算法。只是要保证得到最优解是贪心算法的关键。

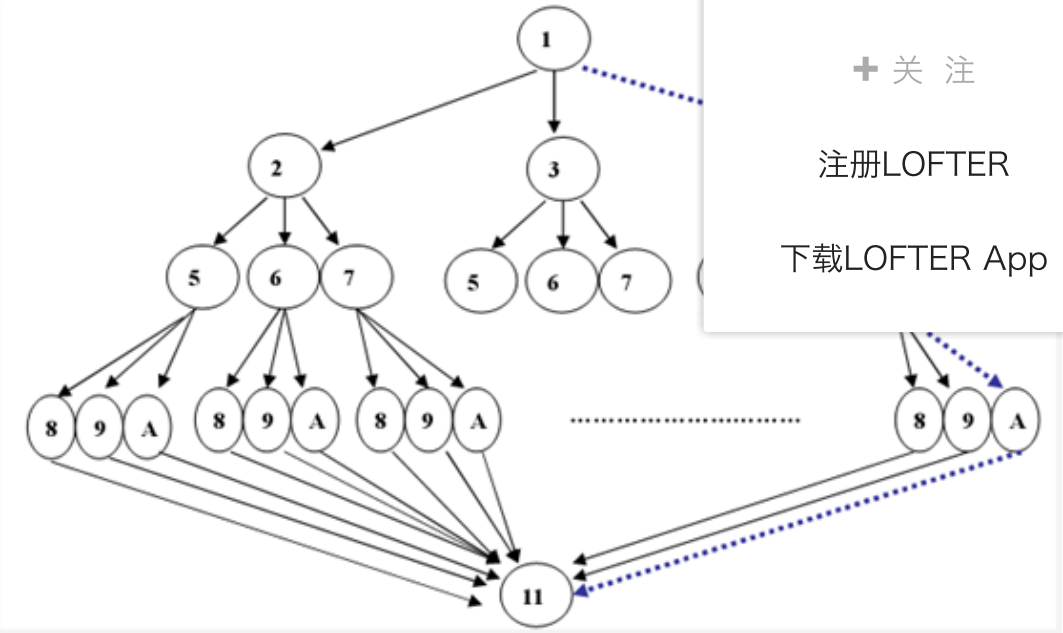

图3

2、背包问题和0-1背包问题

1）0-1背包问题:

给定n种物品和一个背包。物品i的重量是Wi，其价值为Vi，背包的容量为C。应如何选择装入背包的物品，使得装入背包中物品的总价值最大?

在选择装入背包的物品时，对每种物品i只有2种选择，即装入背包或不装入背包。不能将物品i装入背包多次，也不能只装入部分的物品i。

2）背包问题:

与0-1背包问题类似，所不同的是在选择物品i装入背包时，可以选择物品i的一部分，而不一定要全部装入背包，1≤i≤n。

这2类问题都具有最优子结构性质，极为相似，但背包问题可以用贪心算法求解，而0-1背包问题却不能用贪心算法求解。

首先计算每种物品单位重量的价值Vi/Wi，然后，依贪心选择策略，将尽可能多的单位重量价值最高的物品装入背包。若将这种物品全部装入背包后，背包内的物品总重量未超过C，则选择单位重量价值次高的物品并尽可能多地装入背包。依此策略一直地进行下去，直到背包装满为止。

对于0-1背包问题，贪心选择之所以不能得到最优解是[illegible]最终能将背包装满，部分闲置的背包空间使每公斤背包空间的价值降低了。事实上，在考虑0-1背包问题时，应比较选择该物品和不选择该物品[illegible]作出最好选择。由此就导出许多互相重叠的子问题。这正[illegible]求解的另一重要特征。

三、贪心算法的一般框架

经由贪心算法处理的问题需要经过排序。即把“最[illegible]最前面，一直到“最不贪心的”。这是处理问题的第一步[illegible]最终结果。归纳起来，贪心算法处理问题需要下面四步：

1、 读入一个问题

2、 进行贪心排序

3、 处理问题

4、 综合结果并输出

“进行贪心排序”和“处理问题”这两步，是贪心算法的核心部分，甚至能有子问题的重叠和多个贪心问题的重叠。

一般的，经过快速排序产生子问题结果序列的时间复杂度为O(nlogn)。

最小生成树---Kruskla算法

● 贪心

评论