算法导论　第十周作业　11 月 19 日　周四

PB18151866 龚小航

6.1 我们对钢条切割问题进行一点修改，除了切割下的钢条段具有不同价格 p_i 外，每次切割还要付出固定的成本 c。这样，切割方案的收益就等于钢条段的价格之和减去切割的成本。设计一个动态规划算法解决修改后的钢条切割问题。

解：长度为 n 英寸的钢条有 2^{n-1} 种切割方案，因为在距离钢条左端 $i\,(i = 1,2,\dots,n-1)$ 英寸处，我们总是可以选择切割或不切割。但是在实际求解过程中，可以不用遍历这种切割方案，而可以将该问题分解为规模更小的子问题，以下是求解问题的方法：

将钢条从左端切下长度为 i 的一段，其中 $i = 1,2,\dots,n$，有 n 种切法，对这一段不再进行切割，该段的销售收益为 p_i；而右端剩下的长度为 $n-i$，对这一段再进行切割，这是一个规模更小的子问题，其销售收益为 r_{n-i}。即该问题满足最优子结构。可以采用自顶向下或是自底向上方法求解。

每次切割还要付出固定的成本 c 时，该问题仍然满足最优子结构。只需要对算法稍作修改，在计算每种方案的收益时减去切割成本即可（不切割的方案不会有切割成本）：

$$r_n = \max\left\{p_n \max_{1\le i<n}(p_i + r_{n-i} - c)\right\}$$

以下为具体实现伪代码，利用自底向上方法实现：【餐参考课本 208 页代码】

```
//n:钢条总长度
//p:价格表
//c:单次切割成本
//return值:长度为n的钢条的最大收益
BUTTON-UP-CUT-ROD(p,n,c):
 let r[0...n] be a new array
  r[0] = 0;
 for i = 1 to n
     q = p[i];
     for j = i-1 to 1
         q = max(q,p[j]+r[i-j]-c);
     r[i] = q;
 return r[n];
```

6.2．给定一个有向无环图 $G = (V,E)$，边权重为实数，给定图中两个顶点 s 和 t。设计动态规划算法，求从 s 到 t 的最长加权简单路径.

解：该问题不能够用贪心求解，假设从 k 出发，每一步取得 weight 最大的边，按这样的路径，并不能够保证能走到终点 t。所以只能考虑动态规划算法。先将顶点做拓扑排序，取点集 $a[1\dots n]$，其中 $a[1] = s$，$a[n] = t$，这表示路径上的点都是从 s 到 t 路径上的前驱节点。记这个问题的子问题为 LongthestPath(a[1..n])。则动态规划代码如下：

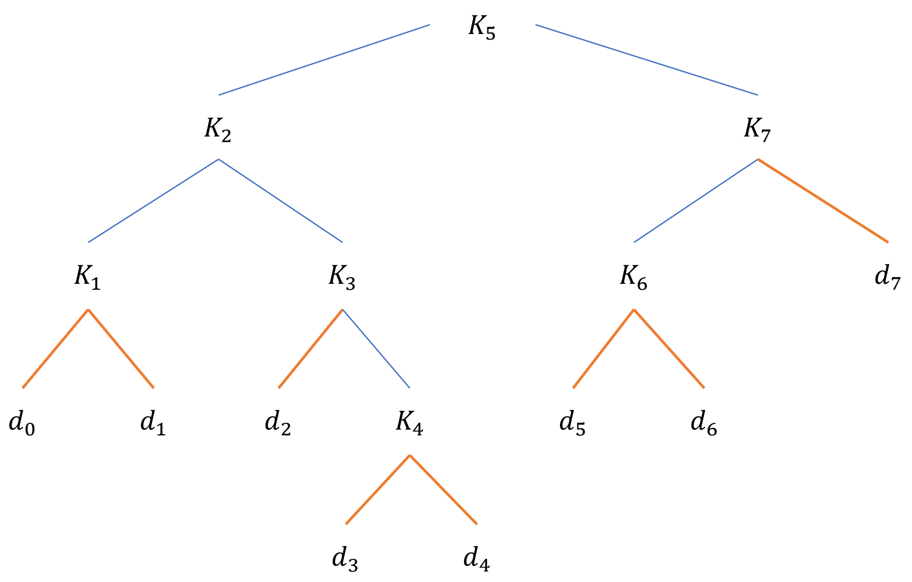
```
LongestPath(s,t):
  if(s==t)
    LWP(s,t)=0;
  else if(s!=t;x∈adj(v))
    LWP(s,t)=max(LWP(s,x)+W(x,t));
```

从点 s 到点 t 的最长简单路径是从点 s 到点 t 前驱节点x的最长简单路径加上xt权重的和的最大值。

6.3 在最优二叉搜索树问题中，若给定 7 个关键字的概率如下所示，求其最优二叉搜索树的结构和代价.(p,q 定义和课本相同).

i	0	1	2	3	4	5	6	7
p_i		0.04	0.06	0.08	0.02	0.10	0.12	0.14
q_i	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05

解：作出其最优二叉搜索树，其结构如下图所示：



结点	深度	概率	贡献
K_1	2	0.04	0.12
K_2	1	0.06	0.12
K_3	2	0.08	0.24
K_4	3	0.02	0.08
K_5	0	0.10	0.10
K_6	2	0.12	0.36
K_7	1	0.14	0.28
d_0	3	0.06	0.24
d_1	3	0.06	0.24
d_2	3	0.06	0.24
d_3	4	0.06	0.30
d_4	4	0.05	0.25
d_5	3	0.05	0.20
d_6	3	0.05	0.20
d_7	2	0.05	0.15
合计			3.12

6.4 一位公司主席正在向 Stewart 教授咨询公司聚会方案。公司的内部结构关系是层次化的，即员工按主管-下属关系构成一棵树，根结点为公司主席。人事部按“宴会交际能力”为每个员工打分，分值为实数。为了使所有参加聚会的员工都感到愉快，主席不希望员工及其直接主管同时出席。公司主席向 Stewart 教授提供公司结构树，采用左孩子右兄弟表示法（参见课本 10.4 节）描述。每个节点除了保存指针外，还保存员工的名字和宴会交际评分。设计算法，求宴会交际评分之和最大的宾客名单。分析算法复杂度。

解：由于某个员工出席则其直接主管以及其下属都不会出席，考虑每个节点 T，以及以 T 为根的子树，对这个节点来说只有参与和不参与两种选择，将 $A(T)$ 记为 T 参与宴会时，以 T 为根的整棵子树的最大分数值，$NOA(T)$ 记为 T 不参与宴会时整棵子树分数的最大值。利用递归算法，能简单的求出宴会交际评分之和的最大值，以及选择哪些员工参加。因为对每个树节点处理时，只需处理它本身以及它的孩子。因此这个算法执行的时间复杂度为 $O(n^2)$.

6.5 设计一个高效的算法，对实数线上给定的一个点集 $\{x_1, x_2, \dots x_n\}$，求一个单位长度闭区间的集合，包含所有给定的点，并要求此集合最小。证明你的算法是正确的。

解：采用贪心算法，将点集 $\{x_1, x_2, \dots x_n\}$ 按升序排序，记得到的新点集记为 $\{y_1, y_2, \dots y_n\}$。贪心算法描述如下：

① 从 y_1 开始，将第一个区间左端点选为 y_1，并将第一个区间内所有的点做标记。

② 选择未被标记的最小点，例如 y_i，以它为新区间的起始点，并将这个新区间内所有的点做标记。

③ 重复上一步，直到 y_i 中所有的点都已经被标记。

为说明这样的贪心算法得到的就是最优解，可以简单地递归说明：

每一步选择区间为了覆盖尽可能多的点，考虑最小值点（或最大值点），必须有某个区间覆盖它，而它左边不存在任何需要覆盖的点，因此把 y_1（或 $y(n)$）选作区间端点能让这个区间尽可能的覆盖更多点，而恰好被这个区间覆盖的点就无需考虑了。每一步的操作也如同第一步一样，只需从未被标记的点集中考虑即可。显然每一步操作都符合最优性质，因为每一次最左端点都必须由一个区间覆盖，而这个区间的左端点如果不在最左点，那么显然不如上述算法能覆盖的未标记点多，因为此时左侧已经没有未标记点需要覆盖。

6.6 考虑用最少的硬币找 n 美分零钱的问题。假定每种硬币的面额都是整数。设计贪心算法求解找零问题，假定有 25 美分、10 美分、5 美分和 1 美分四种面额的硬币。证明你的算法能找到最优解。

解：使用贪心算法，记四种面额的硬币为 c_1, c_2, c_3, c_4，设总共找回 x 枚硬币，找零系列为 $\{X_1, X_2, X_3 \dots X_x\}$，则在第 i 次选取找零的币种时满足 $X_i = \max\{c_i \mid c_i \le n - \sum_{j=1}^{i-1} X_j\}$，即每一个 X_i 都选取当时能选择的最大面额。

再证明这样的选取就能得到最优解：

设最优解总共找回 y 枚硬币，找零系列为 $\{Y_1, Y_2, Y_3, \dots Y_y\}$ 将其按找零面额从大到小排序，即保证 $Y_1 \ge Y_2 \ge Y_3 \ge \dots \ge Y_y$；而贪心算法的性质保证了 $X_1 \ge X_2 \ge X_3 \ge \dots \ge X_x$。假设这两个系列前 i 个元素是一样的，第 $i+1$ 个元素不同：

显然出现这样的情况，只有最优解在 $i+1$ 步时取了一个比贪心更小的面额，记第 i 步后还需要找齐 m 美分，即令 $m = n - \sum_{j=1}^{i} X_j$，由贪心算法的性质，可知 $m \ge X_{i+1}$；再考虑最优解的性质，结合面额考虑，最优解中 1 美分最多有 4 个，因为再多就可以用一个 5 美分硬币替换其中 5 个 1 美分硬币；5 美分硬币最多有 1 个，因为再多就可以用 10 美分硬币替换其中两个 5 美分硬币；10 美分硬币最多有两个，因为多于两个时就可以用一个 25 美分硬币和一个 5 美分硬币替换其中三个 10 美分硬币。结合这个性质，可以发现最优解中若只使用 1 美分硬币，能表示的面额最大为 4 美分；若使用 1,5 两种面额的硬币，最多能表示 9 美分；使用 1,5,10三种硬币，最多能表示 24 美分。（按上面给出的最大硬币数量计算得出）。因此可以对 m 分类讨论：

① $m \ge 25$：此时 $X_{i+1} = 25$，若 $Y_{i+1} \ne 25$，仅凭1,5,10三种硬币表示 $m \ge 25$ 必然会出现上述三种可合并情况之一，因此必不是最优解；

② $10 \le m < 25$：此时 $X_{i+1} = 10$，若 $Y_{i+1} \ne 10$，仅凭 1,5 两种硬币表示 m 必然会出现上述三种可合并情况之一，因此必不是最优解；

③ $5 \le m < 10$：此时 $X_{i+1} = 5$，若 $Y_{i+1} \ne 5$，仅凭 1 美分硬币表示 m 必然会出现能用 5 美分合并 1 美分的情况，因此必不是最优解；

④ $0 \le m < 5$：此时 $X_{i+1} = 1$，若 Y_{i+1} 必须为 1，否则不能完成总计 n 美分的任务；

综上，不可能存在一种情况，使得最优解与贪心算法得到的解在前 i 个元素一样，而 $i+1$ 个元素不同。因此，i 必须等于 x or y，贪心算法得到的解就是最优解。