研究生算法课课堂笔记

上课日期： 2016 年 10 月 27 日　　第(2)节课

组长学号及姓名：1601214533 庄月清

组员学号及姓名：1601111316 张紫薇

1601214534 朱瑞

内容概要

本节课主要讲授二维动态规划加遍历的典型---最优二叉搜索树(POJ-786 Huffman's Greed)，具体包括如下：

1. 最优二叉搜索树的问题描述
2. 最优二叉搜索树的算法设计
3. 最优二叉搜索树的算法分析
4. 问题集合：
   1) 本题 POJ-786 (Huffman's Greed)与算法导论 P226 最优二叉搜索树之间的区别。
   2) 贪心法是否能够解决最优二叉搜索树？
   3) 分治法和动态规划都能分解成子问题，它们之间有什么区别？在最优二叉搜索树问题中，重复子问题如何体现？

详细内容

1. 问题描述

有这样一行已排好序的字符串$K = < k_1, k_2, \ldots, k_n >$（$k_1 < k_2 < \cdots < k_n$），其中，每个关键字$k_i$对应一个搜索频率$p_i$，同时可能存在一些不属于 K 的搜索值，将其定义为伪关键字$d_0, d_1, d_2, \ldots, d_n$，每个伪关键字$d_i$也对应一个搜索频率$q_i$，对于$\forall i \in [1, n-1]$，$d_i$的取值范围为$k_i < d_i < k_{i+1}$，特别的，当 i=0 时，$d_0$表示所有小于$k_1$的值；i=n 时，$d_n$表示所有大于$k_n$的值。

图 1 表示了 n=5 时构造的一棵二叉搜索树，关键字k_i对应了内部节点，伪关键字d_n对应了叶节点。

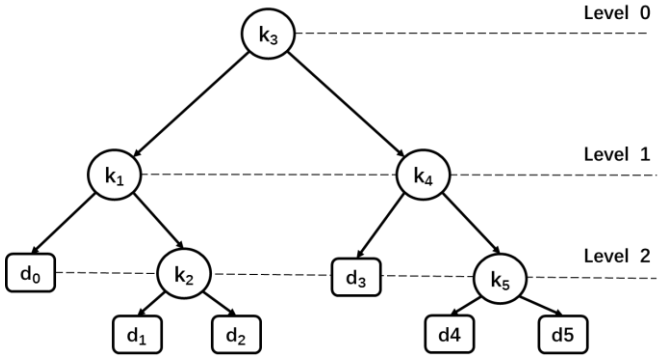


图 1 当 n=5 时构造的二叉搜索树

表 1 节点的搜索概率

	d_0	k_1	d_1	k_2	……	k_n	d_n
命中概率		p_1		p_2	……	p_n	
Miss 概率	q_0		q_1		……		q_n

输入： 命中词典$\langle k1,k2,k3\ldots\ldots kn\rangle$的概率： $p_1,p_2,p_3,p_4\ldots\ldots p_n$

Miss 词典的概率： $q_0,q_1,q_2,q_3\ldots\ldots q_n$

目标函数： $cost=\sum_{i=1}^{n}p_i*(1+depth(k_i))+\sum_{i=1}^{n}q_i*depth(d_i)$

注：该目标方程和算法导论 P226 所述的目标方程有所区别，本题搜索时，若目标值小于 k1，则搜索失败，叶节点 d0 和其根节点 k1 的代价相同。

约束条件： $\sum_{i=1}^{n}p_i+\sum_{i=0}^{n}q_i=1$

输出： **min** $cost$

2. 算法设计

1） 将原问题分解成为子问题

我们先考虑这个问题的子问题，考虑一棵二叉搜索树的子树t[i,j]，它包含了字符串$k_i,\ldots,k_j,(1\le i\le j\le n)$和其叶节点$d_{i-1},\ldots,d_j$。如果一棵最优二叉搜索树 T 有一棵子树t[i,j]，那么必然是包含关键字$k_i,\ldots,k_j$和伪关键字$d_{i-1},\ldots,d_j$的子问题的最优解，如图 2 所示

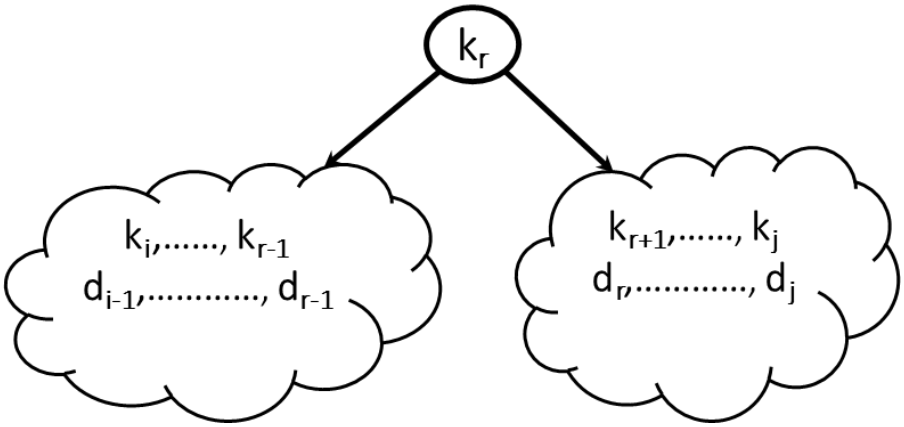


图 2 子问题分解示意图

因此我们可以寻找子问题的最优解从而求得原问题的最优解。在关键字序列$k_i,\ldots,k_j$中找到一个$k_r(i\le r\le j)$，作为最优子树的根节点，其左子树包含关键字$k_i,\ldots,k_{r-1}$和伪关键字$d_{i-1},\ldots,d_{r-1}$，其右子树包含关键字$k_{r+1},\ldots,k_j$和伪关键字$d_r,\ldots,d_j$，可以从 i 到 j 遍历所有可能的k_r，求得使期望代价最小的二叉搜索树，即可找到问题的最优解。

2） 确定状态转移条件

设 f[i,j]为对包含$k_i,\ldots,k_j$和$d_{i-1},\ldots,d_j$的最优二叉搜索树进行一次搜索的最优期望代价值，则它由两部分之和构成：

1） 对其左子树和右子树均进行一次搜索的期望代价值之和$f[i,r-1]$和 $f[r+1,j]$

2） 将左右子树和根节点相连时，期望代价的增加值$s[i,j]$

由于将一棵子树与根节点相连时，每个节点的深度增加 1，因此对于任意节点k_l来说，它的代价增加了p_l的大小，因此整个左子树增加的期望代价为

$$s[i,r-1]=\sum_{l=i}^{r-1}p_l+\sum_{l=i-1}^{r-1}q_l$$

右子树增加的期望代价为

$$\mathrm{s[r+1,j]}=\sum_{l=r+1}^{j}p_l+\sum_{l=r}^{j}q_l$$

因此

$$\mathrm{s[i,j]}=\mathrm{s[i,r-1]}+\mathrm{s[r+1,j]}+p_r=\sum_{l=i}^{j}p_l+\sum_{l=i-1}^{j}q_l$$

（p_r为根节点的搜索代价）。

综上，该最优二叉搜索树的代价递推方程为:

$$f[i,j]=\min_{i\le r\le j}\{f[i,r-1]+f[r+1,j]+s[i,j]\},\quad (1\le \mathrm{i}\le \mathrm{j}\le \mathrm{n})$$

注意，这里不会出现无限次递归的情况，因为无论 r 取 i 或 j，求解$f[i,j]$的时候都不会调用它本身。

3） 确定初始条件

根据状态转移方程$f[i,j]=\min_{i\le r\le j}\{f[i,r-1]+f[r+1,j]+s[i,j]\}$，$(1\le \mathrm{i}\le \mathrm{j}\le \mathrm{n})$

当$\mathrm{j}=\mathrm{i}$时，需要知道$\mathrm{f[i,i-1]}$和$\mathrm{f[i+1,i]}$、$s[i,i]$，因此其初始的条件为

$$\mathrm{f[i,i-1]},\ (1\le \mathrm{i}\le \mathrm{n})$$

但若$\mathrm{j}=\mathrm{i}-1$，意味着该树只包含伪关键字d_{i-1}，则$\mathrm{f[i,i-1]}=0\times q_{i-1}$=0

（注意！与《算法导论》不同，书上该式=q_{i-1}）。

验证如下：

$$\mathrm{f[i,i]}=q_{i-1}+p_i+q_i$$
$$\mathrm{s[i,i]}=q_{i-1}+p_i+q_i$$

又因为$f[i,i]=\min_{i\le r\le i}\{f[i,i-1]+f[i+1,i]+s[i,i]\}$

因此$\mathbf{f[i,i-1]=0}$边界条件成立

4） 确定输出值

根据定义可知$\mathrm{f[1,n]}$是最优搜索树的代价，因此返回值为$\mathrm{f[1,n]}$

算法优化

由于每次计算$f[i,j]$时都需重新计算$\mathrm{s[i,j]}$，因此建立$\mathrm{s[i,j]}$和上一个式子的递推方程为

$$\mathrm{s[i,j]}=\mathrm{s[i,j-1]}+p_j+\ q_j$$

对$\theta(n^2)$个$\mathrm{s[i,j]}$，每个的计算时间变成了$\theta(1)$，提高了算法效率。

综上可得最终$f[i,j]$递归方程为：

$$f[i,j]=\begin{cases}0 & (\mathrm{j}=\mathrm{i}-1)\\ \min\limits_{i\le r\le j}\{f[i,r-1]+f[r+1,j]+s[i,j]\} & (1\le \mathrm{i}\le \mathrm{j}\le \mathrm{n})\end{cases}$$

$s[i,j]$递归方程为：

$$s[i,j]=\begin{cases}q_{i-1} & (\mathrm{j}=\mathrm{i}-1)\\ \mathrm{s[i,j-1]}+p_j+q_j & (1\le \mathrm{i}\le \mathrm{j}\le \mathrm{n})\end{cases}$$

3. 算法分析

1） 空间复杂度$\boldsymbol{\theta(n^2)}$，能否采用滚动数组进行优化？

根据状态初始条件，得到状态初始状态如图 3 所示，其状态递推过程如图 4 所示，为了存储状态空间，需要空间复杂度为$\boldsymbol{\theta\left(\frac{n(n+1)}{2}+n+1\right)=\theta(n^2)}$。

那能否采用滚动数组进行优化空间复杂度？答案是否定的，根据状态转移方程

$$f[i,j]=\min_{i\le r\le j}\{f[i,r-1]+f[r+1,j]+s[i,j]\}$$

可知，其状态值$\mathrm{f}[i,j]$依赖的子问题对分别是$\{\mathrm{f}[i,i-1],f[i+1,j]\}$，$\{\mathrm{f}[i,i],f[i+2,j]\}$，…… $\{\mathrm{f}[i,j-1],f[j+1,j]\}$，也就是说，$\mathrm{f}[i,j]$依赖于之前在区间$[i,j]$内的所有由小到大的子问题，如图 5 和图 6，为了求解$\mathrm{f}[1,3]$**(图 5 所示)**和$\mathrm{f}[2,4]$**(图 6 所示)**，其依赖的状态为图中的子问题对$\boldsymbol{r=i,i+1\ldots\ldots j}$，由于需要保存之前所有的子问题，因此不能采用滚动数组优化空间复杂度，因此其空间复杂度为：

$$\boldsymbol{\theta\left(\frac{n(n+1)}{2}+n+1\right)=\theta(n^2)}。$$

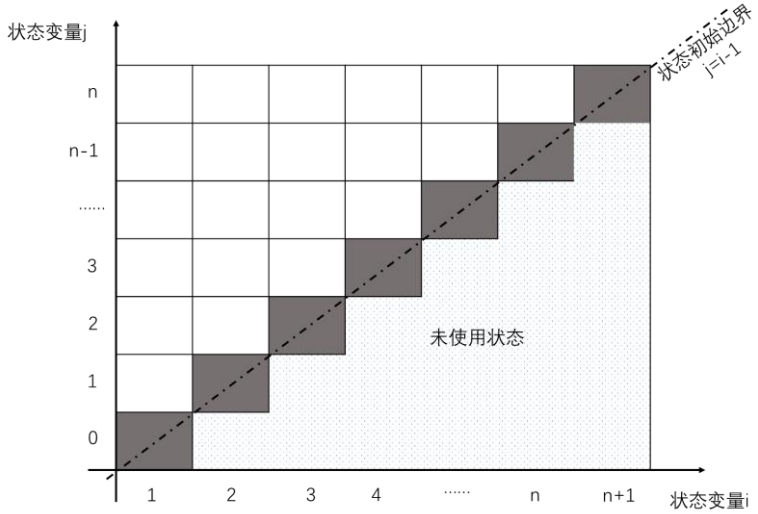


图 3 状态空间初始化示意图

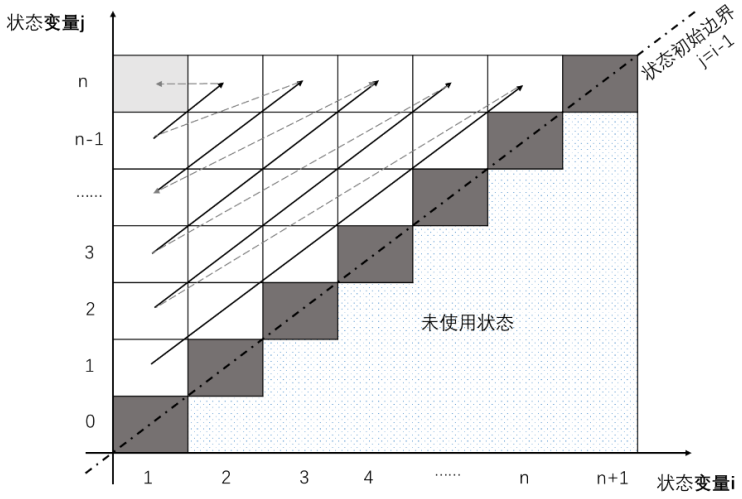


图 4 状态空间递推过程示意图

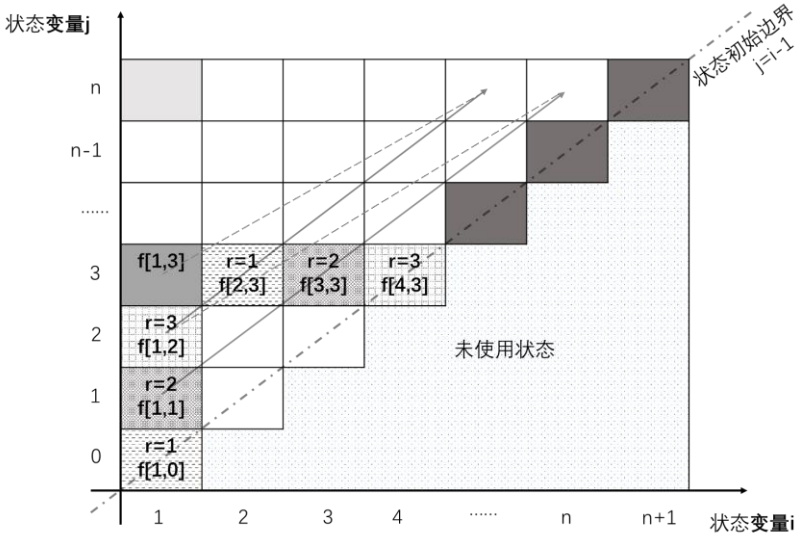


图 5 f[1，3]状态所依赖的子问题状态

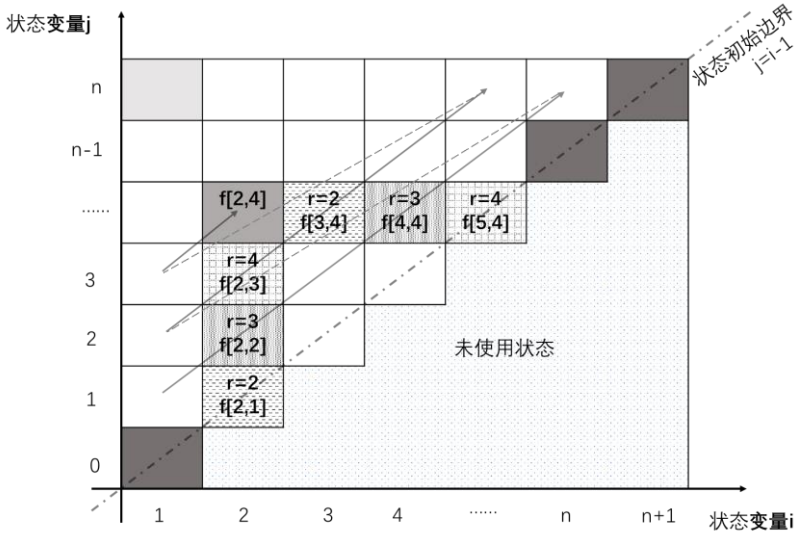


图 6 f[2,4]状态所依赖的子问题的状态

2） 时间复杂度$O(n^3)$：

初步推导： 1）子问题长度 L 从 1 到 n 所需要的次数为 n

2）每种长度 L 的起点从 1 到 n-L+1 所需要的次数为 n-L+1

3）遍历在长度 L 的子问题中，以各个节点为根结点的次数为 L

由于其包含了三重 for 循环，而每层的下标最大值为 n，因此可以知道其运行时间为$O(n^3)$

详细公式推导,运行时间T(n)公式如下：

$$T(n)=\sum_{L=1}^{n}(n-L+1)*L=(n+1)\sum_{L=1}^{n}L-\sum_{L=1}^{n}L*L$$

$$=(n+1)*\frac{n(n+1)}{2}-\frac{n(n+1)(2n+1)}{6}$$

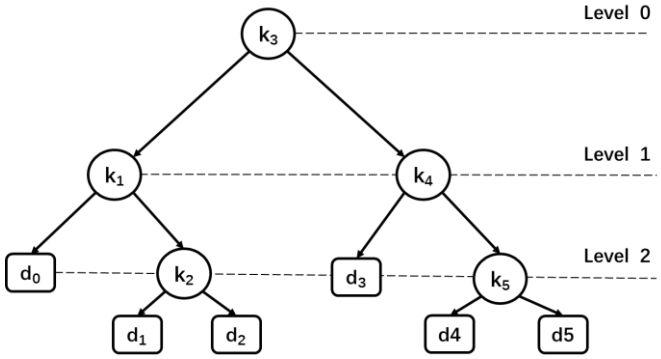
$$= (\boldsymbol{n+1}) * \frac{\boldsymbol{n(n+1)}}{\boldsymbol{2}} - \frac{\boldsymbol{n(n+1)(2n+1)}}{\boldsymbol{6}}$$

$$= \frac{\boldsymbol{n^3 + 3n^2 + 2n}}{\boldsymbol{6}} = \ \boldsymbol{O(n^3)}$$

采用四边形不等式优化加速可以获得$O(n^2)$的时间复杂度，本课程不做要求。

4. 问题集合

1) 本题 POJ-786 (Huffman's Greed)与算法导论 P226 最优二叉搜索树之间的区别。



主要的区别在于查找字典失败时的搜索路径的**深度/层次（depth/level）**的定义是不一样的，不同如下：

- 算法导论 P226 页中其定义为，叶子节点的查找次数也就是叶子节点所处的深度加 1：

  即： 次数 = depth(d) + 1

- POJ786(Huffman's Greed)：查找失败的深度/层次（depth/level）等于查找父节点的次数：

  即：次数 = depth(d)

这就导致优化的目标函数的不同：

- 算法导论 P226 页：

$$cost = \sum_{i=1}^{n} p_i * (1 + depth(k_i)) + \sum_{i=1}^{n} q_i * (depth(d_i) + 1)$$

- POJ786(Huffman's Greed)：

$$cost = \sum_{i=1}^{n} p_i * (1 + depth(k_i)) + \sum_{i=1}^{n} q_i * depth(d_i)$$

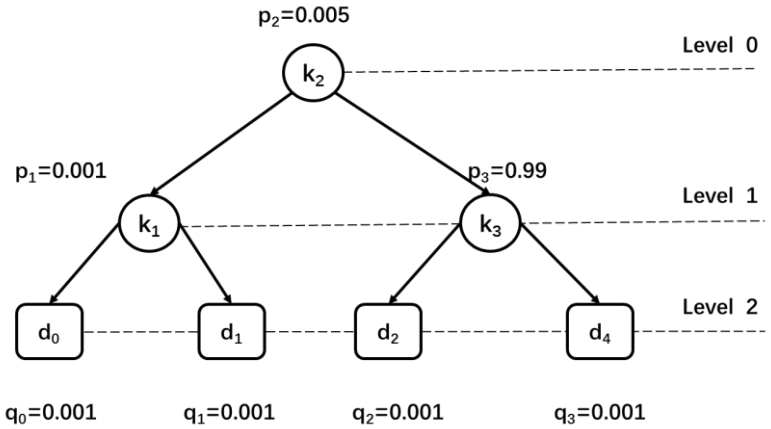
2) 贪心法是否能够解决最优二叉搜索树？

贪心法是不能解决最优二叉搜索树的，举例如下所示：

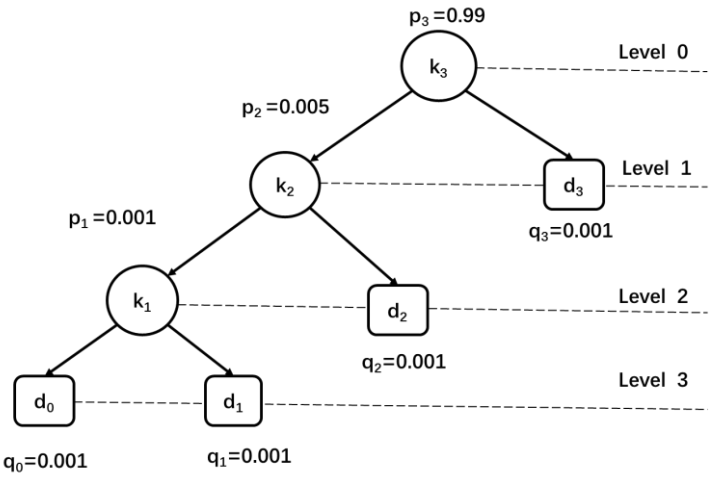
(1)第一种贪心策略：让二叉搜索树的高度尽可能的低，其等价于选择越靠近中心的节点作为树的根节点，当叶子节点和内部节点的概率均匀分布的时候，此时该策略得出的答案是最优的。

但是这种思想在节点概率分布很不均匀的情况下树的高度低的不一定是最优解：

比如图 7 所示



$$Cost = 0.005 + 0.991 * 2 + 0.001 * 2 = 1.994$$



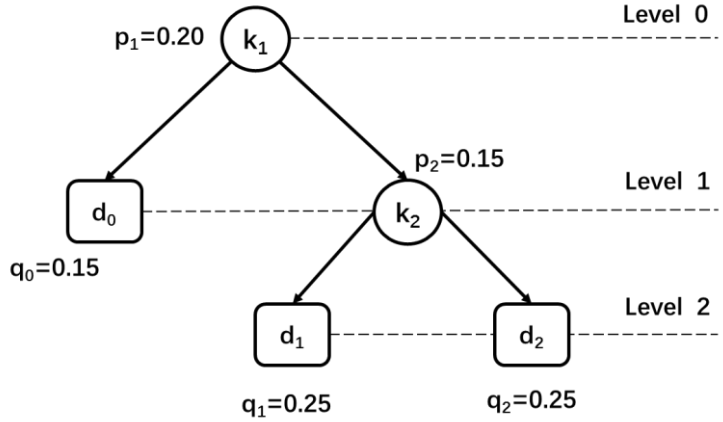
$$Cost = 0.99 + 0.005 * 2 + 0.001 * 3 + 0.002 * 3 + 0.001 * 2 + 0.001 = 1.0012$$

图 7 高度最矮≠最优二叉搜索树

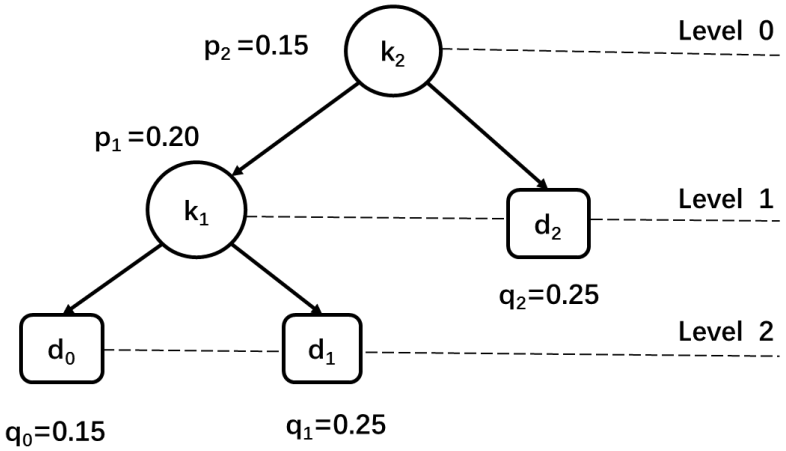
结论：高度为3的树的代价小于高度为2的树的代价，贪心方法不可以找出最优二叉树。这是由于采用贪心算法建立的高度最矮的二叉搜索树其根节点一定是处在序列$\langle k1, k2, k3 \ldots\ldots kn\rangle$的中间$k_{mid}$的位置，此时$k_{mid}$所对应的概率并不是最大的，所以构建出来的最优的二叉搜索树并不是最优的。

（2）第二种贪心策略：先找出当前子树中概率最大的节点作为根节点，序号小于该节点的在左边，序号大于该节点的在右边，再选择概率最大的作为其子树的根节点，依次递归生成一颗搜索树。但是该搜索策略并不是最优的，反例如图 8，因此选择

举例如下：



$$\text{Cost} = 0.20 + 0.15 * 2 + 0.15 * 1 + 0.25 * 2 + 0.25 * 2 = 1.65$$



$$\text{Cost} = 0.15 + 0.20 * 2 + 0.15 * 2 + 0.25 * 2 + 0.25 * 1 = 1.60$$

图 8 选概率最大的节点作为子节点≠最优二叉搜索树

综上所述，按照贪心法选择高度最矮或者选择概率最大的节点作为根结点，都不一定能获得最优二叉搜索树,因此不能使用贪心法进行求解，这与 POJ5941 Huffman 编码树有所区别。

3） **分治法和动态规划都能分解成子问题，它们之间有什么区别？在最优二叉搜索树问题中，重复子问题如何体现？**

- 分治法的子问题之间是**没有重叠的**，子问题之间是相互独立的。
- 动态规划的子问题之间是**有重叠**的，如果采用暴力算法计算，则其计算的复杂度是指数级别,采用动态规划的算法则将重叠的子问题计算单次,而不是多次。例如 fibonacci 数列中，其递归方程为

$$\text{fib(n)} = \text{fib(n} - 1) + \text{fib(n} - 2)$$

计算$\text{fib(n} - 1) = \text{fib(n} - 2) + \text{fib(n} - 3)$时候，又重新计算了$\text{fib(n} - 2)$，此时也就存在重叠的子问题。

在本题中，重复子问题的由来是因为不知道根节点所在的位置，因此需要遍历所有的节点来寻找最优的根节点，如下图所示，以 k3 为根节点的搜索树和以 k4 为根结点的搜索树，都需要求解由 k1,k2 构成的最优的搜索树，因此其存在着重复的子问题。

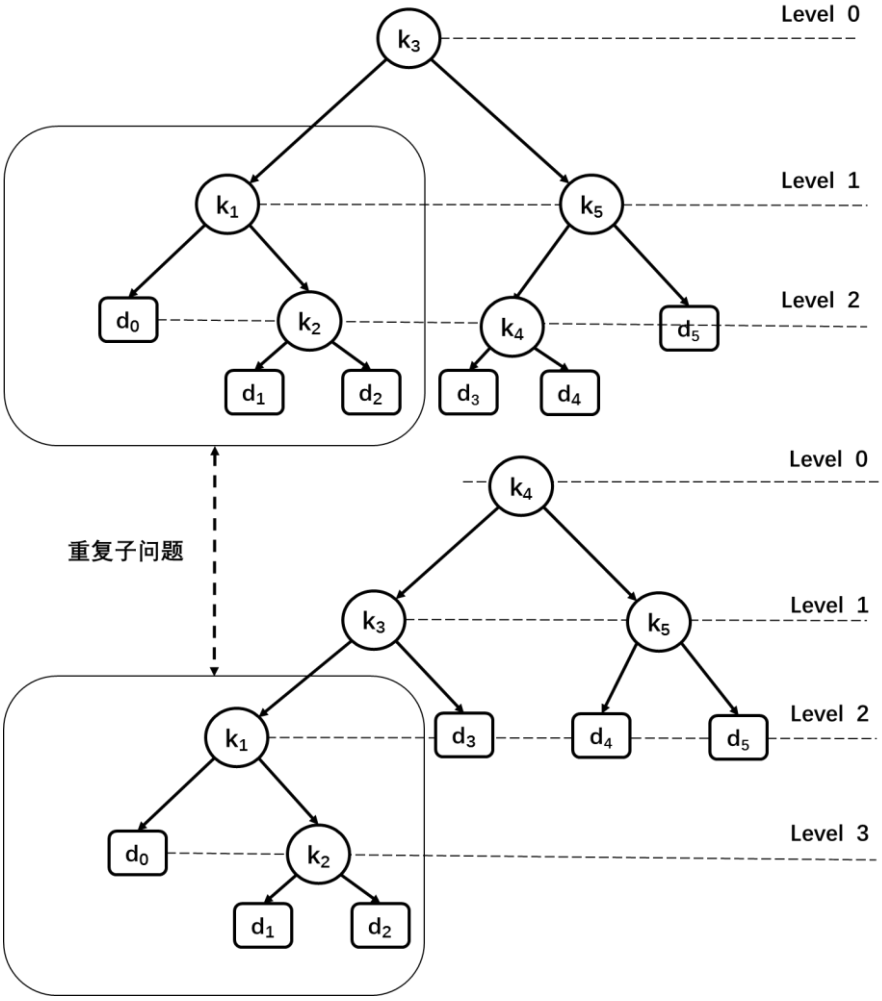


图 9 最优二叉搜索树重叠子问题示意图