1-3

最优二叉查找树

一、二叉查找树

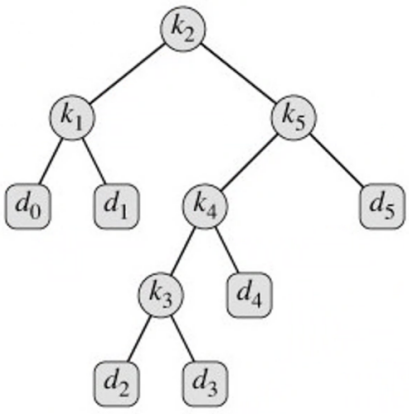
二、最优二叉查找树

任何一个查找算法的优劣都取决于两个因素

1. 问题初始的规模（状态）（类似于初始有序/无序）
2. 建立在取每一个数的概率相同的平均的假设之上

最优二叉查找树

问题背景：访问每一个元素概率需要不同



在一棵二叉树中，访问深度为k处的一个元素所需要比较的代价是pi(1+k)
——即概率*比较次数（深度+1），所以对于整个树来说我们就是要Σpi(1+ki)极小化
——就标志着搜索整棵树的代价最小。

理论基础

> 动态规划：一种数学优化方法，也是一种变成思想。

数学优化——最优子结构

——规定的是子问题与原问题的关系

通过一定的数学方法对各个子问题的最优解进行组合得出的最优结果。

编程思想——重叠子问题

例：运用递归求斐波那契数列（并不属于严格动态规划的例子）

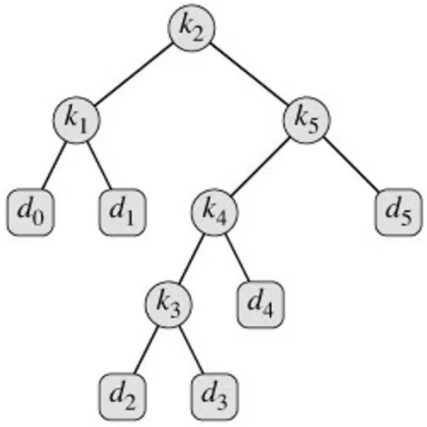
分析问题

1. 是否由最优子结构构成最优解法
2. 是否有重叠子问题

Q:最优子结构一定可以构成原问题最优解？考虑的子结构考虑了整体？

解决问题

1. 求递归子问题的最优化解的统一计算式



1.查找成功（设其查找概率为pi）
2.查找失败，也就是找到的是我们人为添加的伪关键字，这些伪关键字在整棵二叉树的最底端，是我们新添加叶子节点di（设其查找概率为qi）

`E[树中搜索一次的代价]=(ki+1)*pi+(di+1)*pi`

E[树中一次搜索的代价]=(ki+1)*pi+(di+1)*qi

E[整棵树中搜索要访问的节点数]=Σ[(ki+1)*pi+(di+1)*qi]

Σpi+Σqi=1

E[T中的搜索代价]=1+Σki*pi+Σdi*qi

一棵树的搜索期望=左子树的搜索期望问题+根的搜索期望问题+右子树的搜索期望问题

E[i,j]=p_r+(e[i,r-1]+w(i,r-1))+(e[r+1,j]+w(r+1,j))

总的搜索期望=左子树搜索期望+右子树搜索期望+根节点期望+变化得到的搜索期望代价的增加量

w(i,j)=w(i,r-1)+p_r+w(r+1,j)

即整棵子树的概率之和为选中的r根节点的左子树概率之和+根节点本身概率+其右子树概率之和

$$e[i,j]=\begin{cases} q_{i-1} & \text{如果 } j=i-1 \\ \min\limits_{i\leqslant r\leqslant i}\{e[i,r-1]+e[r+1,j]+\omega(i,j)\} & \text{如果 } i\leqslant j \end{cases}$$

2.计算一个最优二叉查找树的搜索代价期望——以自底向上的方式计算出各个子问题、原问题的最优值，并避免子问题的重复计算；
3．根据计算最优值时得到的信息，构造最优解。