最优二叉搜索树

Optimal Binary Search Trees

1 二叉搜索树

2 最优二叉搜索树

3 最优二叉搜索树问题描述

4 最优子结构性质

5 递归计算最优值

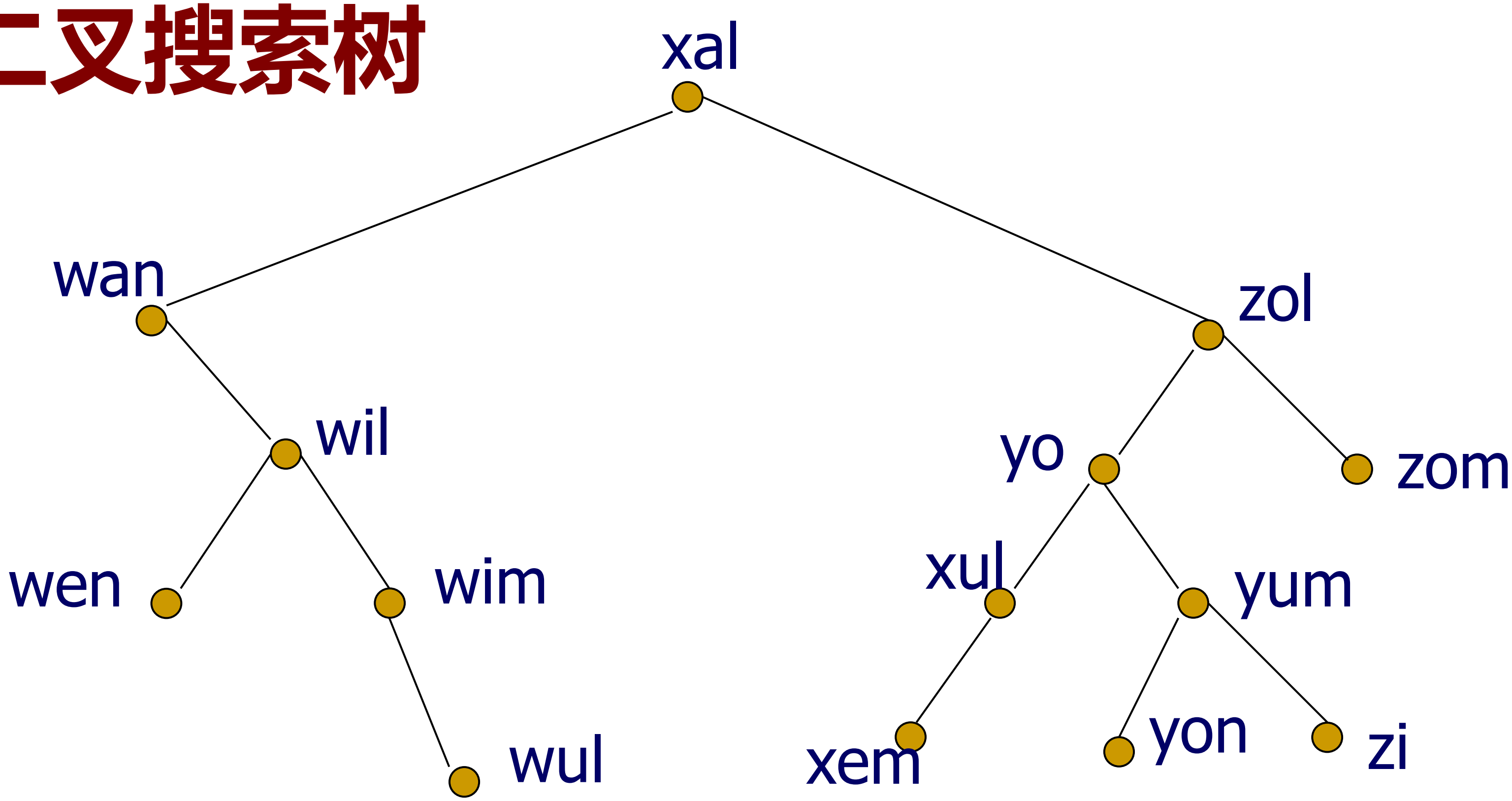
6 算法

1 二叉搜索树

一棵空树或者满足以下的性质：

- 每个结点作为搜索对象，它的关键字是互不相同的。
- 对于树上的所有结点，如果它有左子树，那么左子树上所有结点的关键字都小于该结点的关键字。
- 对于树上的所有结点，如果它有右子树，那么右子树上所有结点的关键字都大于该结点的关键字。

1 二叉搜索树



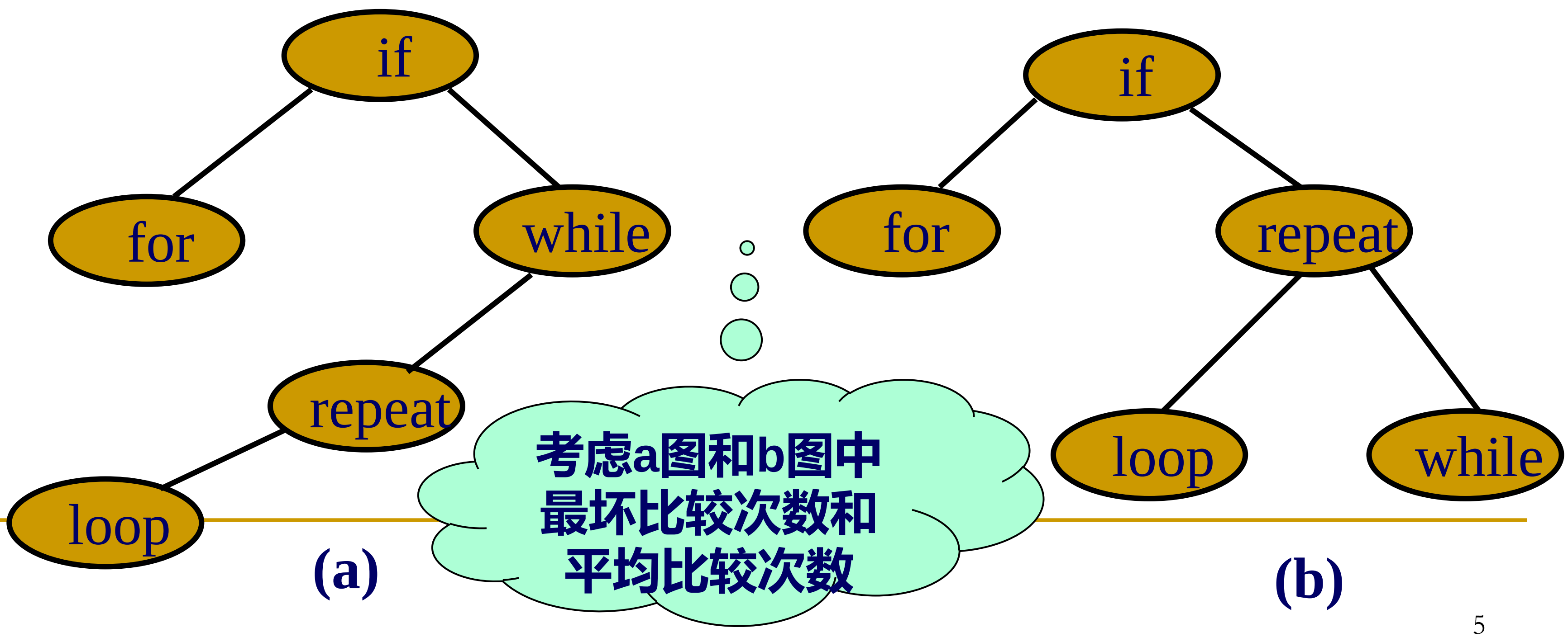
搜索过程：从根结点开始，如果根为空，则搜索不成功；否则使用待搜索值与根结点比较，如果待搜索值等于根结点关键字，则搜索成功返回，如果小于根结点，则向左子树搜索；如果大于根结点，则向右子树搜索。

1 二叉搜索树

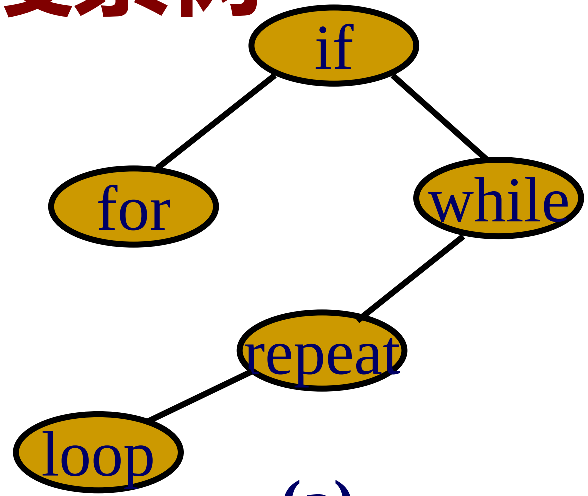
对于一个给定的关键字集合，可能有若干不同的二叉搜索树如：

Name：	**1**	**2**	**3**	**4**	**5**
	for	**if**	**loop**	**repeat**	**while**

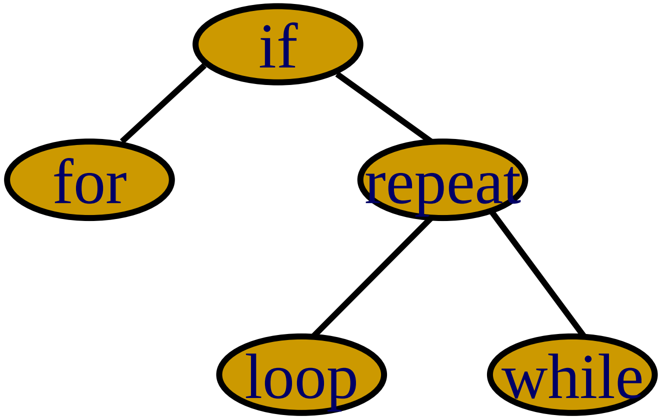
的两棵二叉搜索树为



1 二叉搜索树



(a)



(b)

- 构造不同的二叉搜索树就有不同的性能特征。
- 二叉搜索树a在最坏情况下找一个标识符需要4次比较，而b表示的二叉搜索树最坏情况下只需3次比较。
- 假设只作成功的检索并且检索每个标识符的概率相同，则两棵二叉搜索树在平均情况下各需要12/5和11/5次比较。

2 最优二叉搜索树

存在的两个问题

1 在实际中也会遇到不成功检索的情况。

2 在实际中，不同标识符会有不同的检索概率。

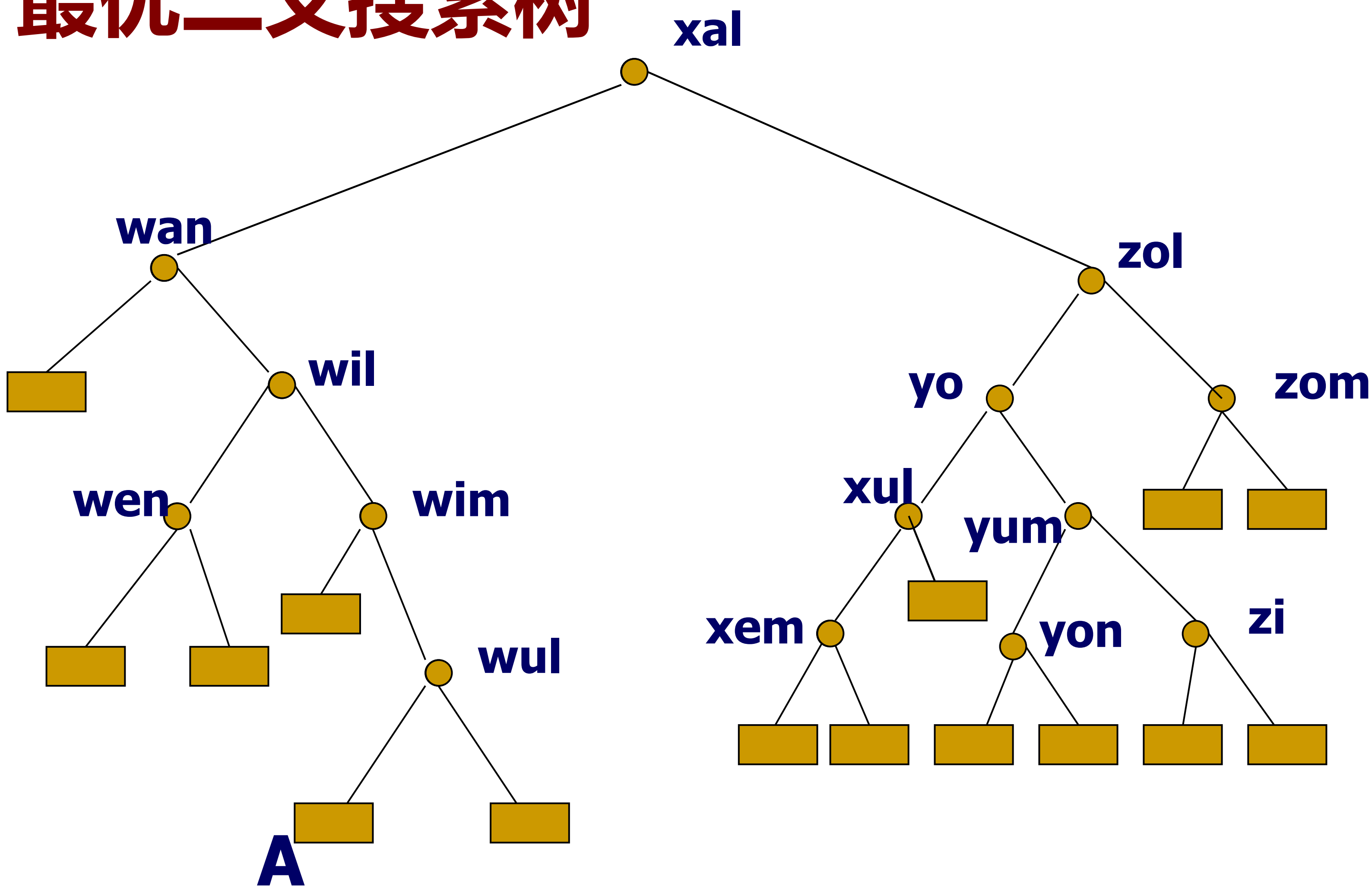
对给定的标识符集合，希望给出构造二叉搜索树的方法，使得所构造的二叉搜索树具有最优的性能。

2 最优二叉搜索树

在实际中也会遇到不成功检索的情况

- 扩充二叉树：当二叉树里出现空的子树时，就增加新的、特殊的结点——空树叶。对于原来二叉树里度数为1的分支结点，在它下面增加一个空树叶；对于原来二叉树的树叶，在它下面增加两个空树叶。

2 最优二叉搜索树



- A代表其值处于wim和wul之间的可能关键字集合

2 最优二叉搜索树

在二叉搜索树中搜索一个元素x

设 $S=\{x_1, x_2, \cdots, x_n\}$ 是一个有序集合，且$x_1, x_2, \cdots, x_n$ 表示有序集合的二叉搜索树利用二叉树的顶点存储有序集中的元素，而且具有性质：

- 存储于每个顶点中的元素x 大于其左子树中任一个顶点中存储的元素，小于其右子树中任意顶点中存储的元素。二叉树中的叶顶点是形如(x_i, x_{i+1}) 的开区间。

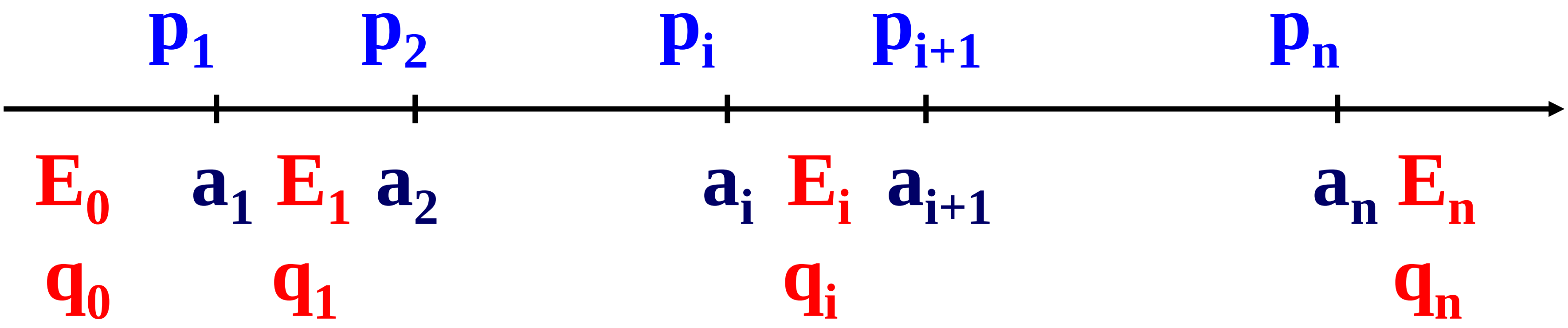
(1) 在二叉树的内部顶点处找到： $x = x_i$

(2) 在二叉树的叶顶点中确定： $x \in (x_i, x_{i+1})$

2 最优二叉搜索树

在实际中，不同标识符会有不同的搜索概率。

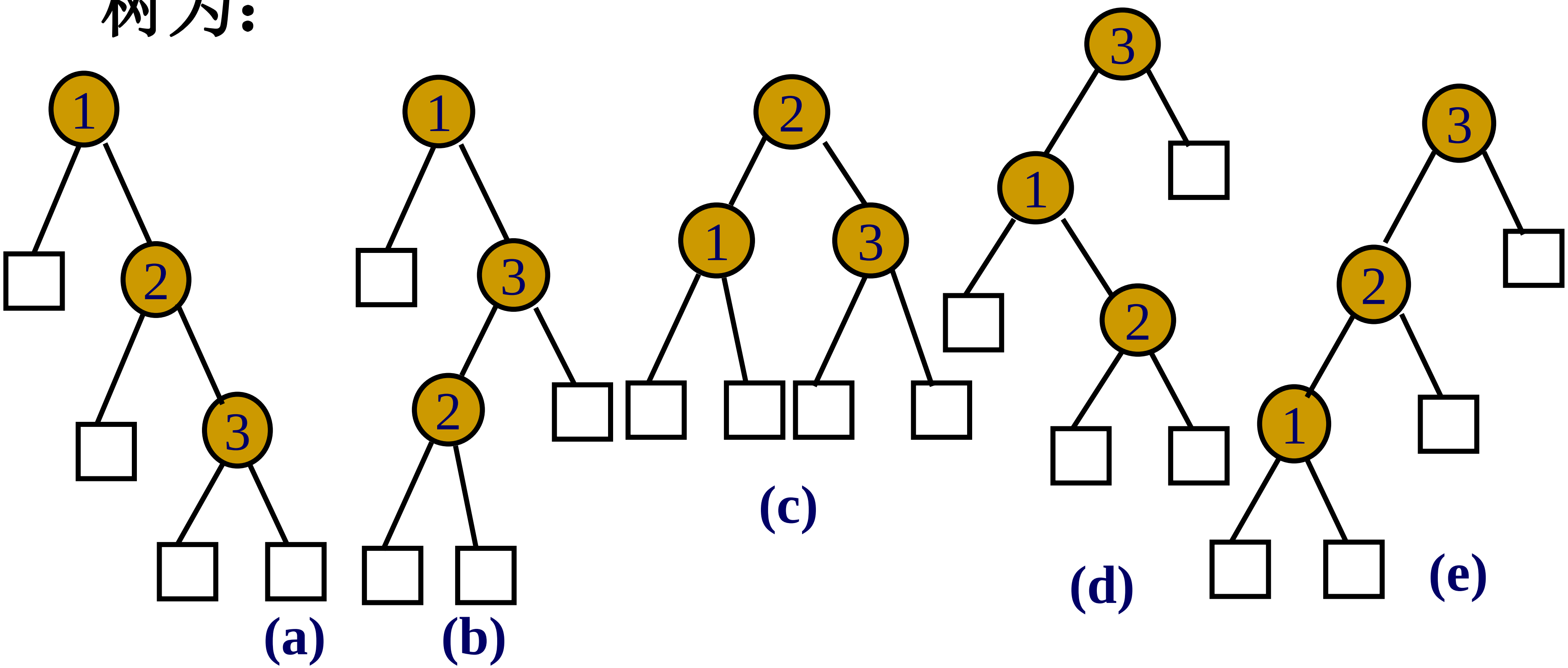
- 设p_i是对a_i检索的概率。
- 设q_i是对满足$a_i<X<a_{i+1}, 0\leq i \leq n$的标识符X检索的概率，（假定$a_0=-\infty$且$a_{n+1}=+\infty$）。



2 最优二叉搜索树

利用动态规划构造对标识符集合 **{a1, a2, …, an}**的最优二叉搜索树算法（包括成功检索和不成功检索）。

例 标识符集{1, 2, 3}={do, if, stop}可能的二叉搜索树为：



设每个内、外结点检索的概率相同：$p_i=q_i=1/7$，求每棵树的平均比较次数（成本）。

若$P_1=0.5$, $P_2=0.1$, $P_3=0.05$,

$q_0=0.15$, $q_1=0.1$, $q_2=0.05$, $q_3=0.05$，求每棵树的平均比较次数（成本）。

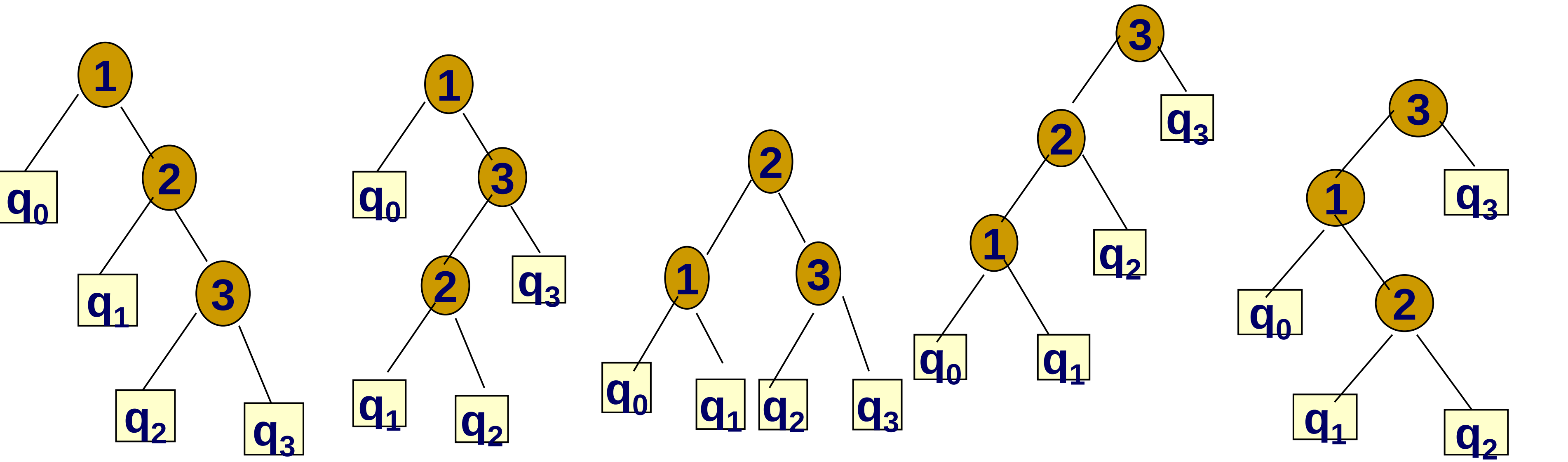
2 最优二叉搜索树

在搜索过程中，每进行一次比较，就进入下面一层，

- 对于成功的搜索，比较的次数就是所在的层数加**1**。
- 对于不成功的搜索，被搜索的关键字属于哪个外部结点代表的可能关键字集合，比较次数就等于此外部结点的层数。

2 最优二叉搜索树

例：$P_1=0.5, P_2=0.1, P_3=0.05, q_0=0.15, q_1=0.1, q_2=0.05, q_3=0.05$



(a) (b) (c) (d) (e)

考虑平均搜索次数，也叫做平均路长

$P_a(n)=1\times p_1+2\times p_2+3\times p_3+1\times q_0+2\times q_1+3\times(q_2+q_3)$

$=1\times 0.5+2\times 0.1+3\times 0.05+1\times 0.05+2\times 0.1+3\times(0.05+0.05)$

$=1.5$

2 最优二叉搜索树

分析

对于图的内结点而言，第0层需要比较操作次数为1，第1层需要比较2次，第2层需要3次

- $P_b(n)=1\times p_1+2\times p_3+3\times p_2+1\times q_0+2\times q_3+3\times(q_1+q_2)$
  $=1\times 0.5+2\times 0.05+3\times 0.1+1\times 0.15+2\times 0.05+3\times(0.1+0.05)$
  $=1.6$
- $P_c(n)=1\times p_2+2\times(p_1+p_3)+2\times(q_0+q_1+q_2+q_3)$
  $=1\times 0.1+2\times(0.5+0.05)+2\times(0.15+0.1+0.05+0.05)$
  $=1.9$
- $P_d(n)=1\times p_3+2\times p_2+3\times p_1+1\times q_3+2\times q_2+3\times(q_1+q_0)$
  $=1\times 0.05+2\times 0.1+3\times 0.5+1\times 0.05+2\times 0.05+3\times(0.1+0.15)$
  $=2.65$
- $P_e(n)=1\times p_3+2\times p_1+3\times p_2+1\times q_3+2\times q_0+3\times(q_1+q_2)$
  $=1\times 0.05+2\times 0.5+3\times 0.1+1\times 0.05+2\times 0.15+3\times(0.1+0.05)$
  $=2.15$

2 最优二叉搜索树

- 找到元素$x = x_i$的概率为b_i；确定$x \in (x_i, x_{i+1})$的概率为a_i。其中约定$x_0 = -\infty$, $x_{n+1} = +\infty$,有

$$a_i \geq 0, 0 \leq i \leq n; b_j \geq 0, 1 \leq j \leq n; \sum_{i=0}^{n} a_i + \sum_{j=1}^{n} b_j = 1$$

称集合$\{a_0, b_1, a_1, \cdots, b_n, a_n\}$为有序集 S 的存取概率分布。

2 最优二叉搜索树

- 在一个表示S的二叉树T中，设存储元素x_i的结点深度为c_i；叶结点（x_j，x_{j+1}）的结点深度为d_j 。

$$p=\sum_{i=1}^{n} b_i(1+c_i)+\sum_{j=0}^{n} a_j d_j$$

表示在二叉搜索树T中作一次搜索所需的平均比较次数。p又称为二叉搜索树T的平均路长，在一般情况下，不同的二叉搜索树的平均路长是不同的。

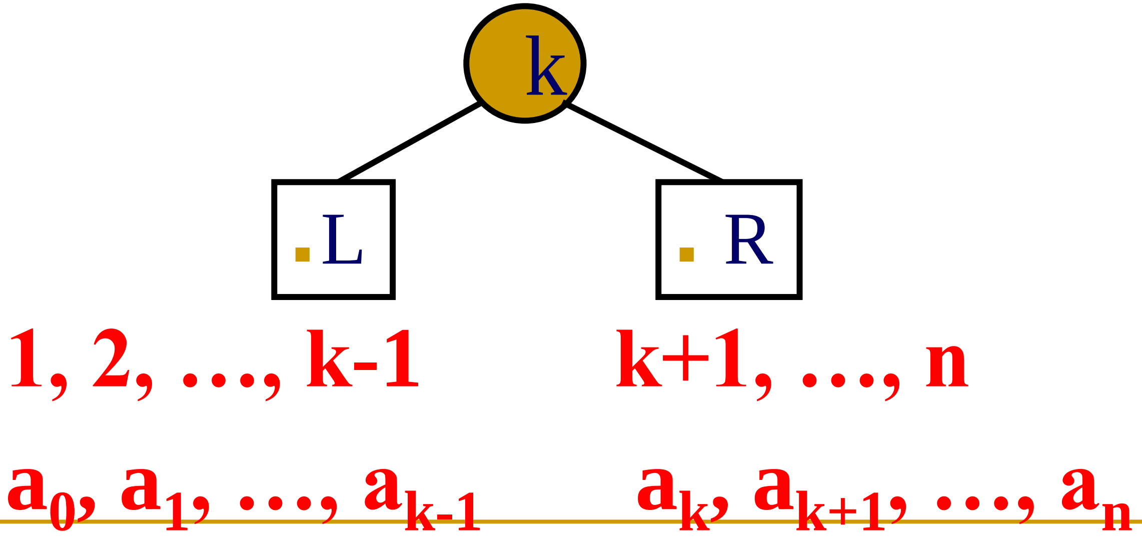
3 最优二叉搜索树问题描述

- 对于有序集S及其存取概率分布（$a_0, b_1, a_1, \cdots, b_n, a_n$），在所有表示有序集S的二叉搜索树中找出一棵具有最小平均路长的二叉搜索树。

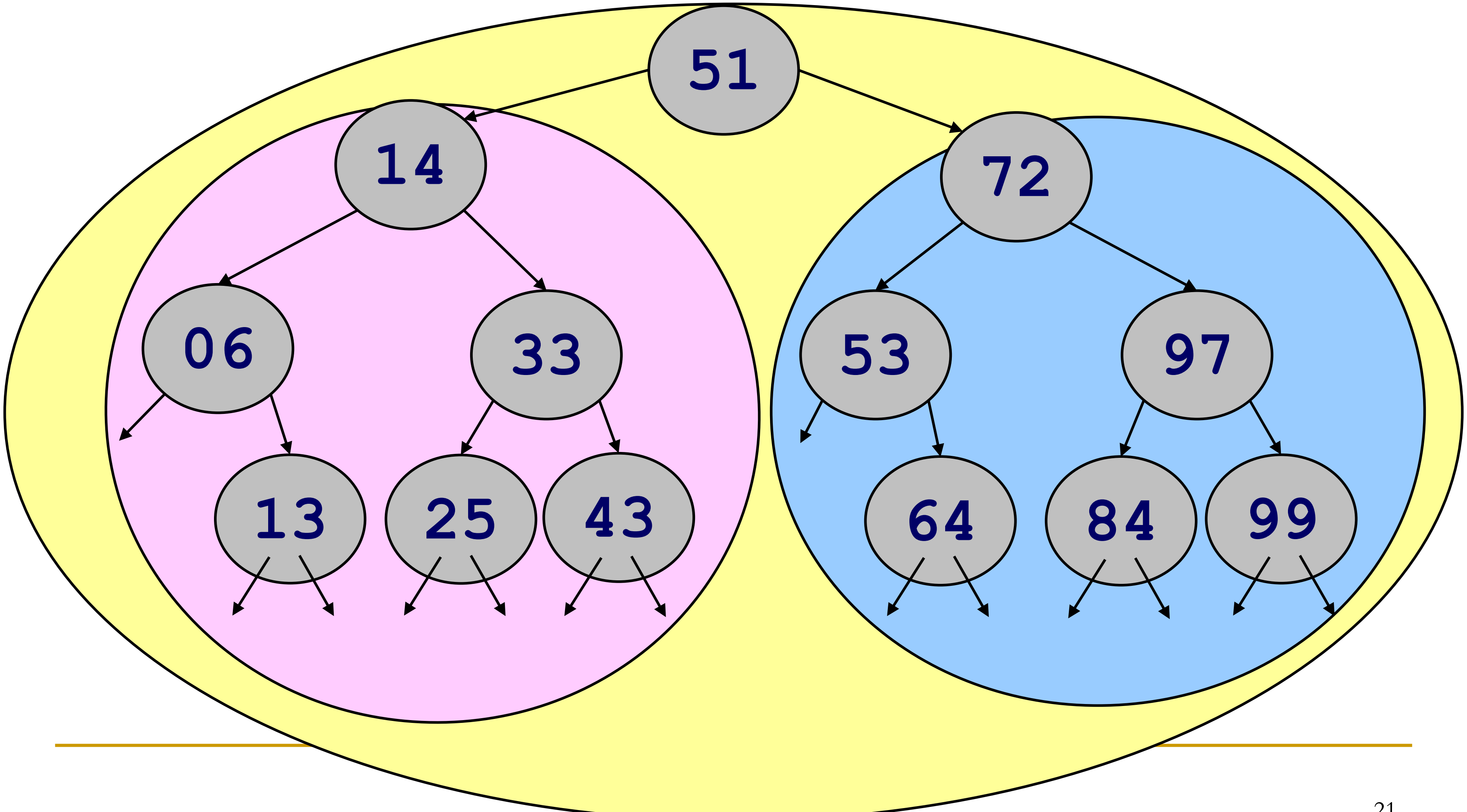
结点在二叉搜索树中的层次越深，需要比较的次数就越多，因此要构造一棵最小二叉树，一般尽量把搜索概率较高的结点放在较高的层次。

4、最优子结构性质

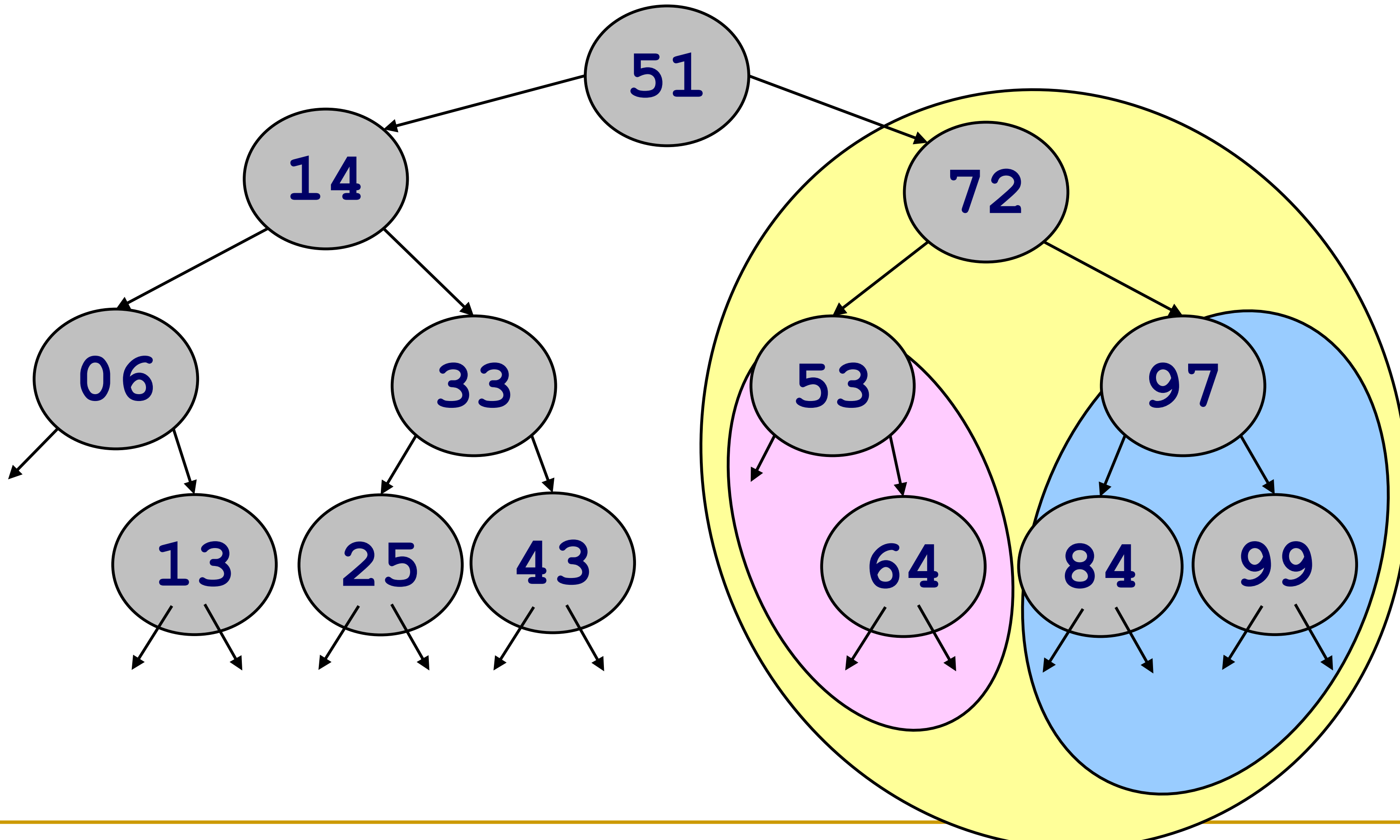
假设选择 k为树根，则 1, 2, …, k-1 和a_0, a_1, …, a_{k-1} 都将位于左子树 L 上，其余结点 (k+1, …, n 和 a_k, a_{k+1}, …, a_n)位于右子树 R 上。



4 最优子结构性质



4 最优子结构性质



4 最优子结构性质

设COST(L) 和COST(R) 分别是二叉搜索树T的左子树和右子树的成本。

则检索树T的成本是：

P(k)+ COST(L) + COST(R) + ……

若 T 是最优的，则上式及 COST(L) 和COST(R)必定都取最小值。

4 最优子结构性质

最优子结构性质证明

- 二叉搜索树T 的一棵含有顶点$x_i, \cdots, x_j$和叶顶点 $(x_{i-1}, x_i), \cdots, (x_j, x_{j+1})$的子树可以看作是有序集$\{x_i, \cdots, x_j\}$关于全集为$\{x_{i-1}, x_{j+1}\}$的一棵二叉搜索树（T 自身可以看作是有序集)。

根据*S* 的存取分布概率，在子树的顶点处被搜索到的概率是：

$$w_{ij} = \sum_{i-1 \le k \le j} a_k + \sum_{i \le k \le j} b_k$$

4 最优子结构性质

- $\{x_i, \cdots, xj\}$的存储概率分布为$\{\bar{a}_{i-1}, \bar{b}_i, \ldots, \bar{b}_j, \bar{a}_j\}$，其中，$a_h$，$b_k$分别是下面的条件概率：

$$\bar{b}_k = b_k / w_{ij}, \quad i \le k \le j; \qquad \bar{a}_h = a_h / w_{ij}, \quad i-1 \le h \le j$$

- 设Tij是有序集$\{x_i, \cdots, xj\}$关于存储概率分布为$\{a_{i-1}, b_i, \ldots, b_j, a_j\}$的一棵最优二叉搜索树，其平均路长为$p_{ij}$，$T_{ij}$的根顶点存储的元素$x_m$，其左子树$T_l$和右子树$T_r$的平均路长分别为$p_l$和$p_r$。由于$T_l$和$T_r$中顶点深度是它们在$T_{ij}$中的深度减1，所以得到

左子树的搜索概率

右子树的搜索概率

$$w_{ij} p_{ij} = w_{i,m-1}(p_l + 1) + w_{m,m} + w_{m+1,j}(p_r + 1)$$

$$= w_{ij} + w_{i,m-1} p_l + w_{m+1,j} p_r \qquad i \le j$$

4 最优子结构性质

由于T_l是有序集$\{x_i,\cdots,x_{m-1}\}$的一棵二叉搜索树，故$p_l \geq p_{i,m-1}$.若$p_l > p_{i,m-1}$，则$T_{i,m-1}$替换T_l可得到平均路长比T_{ij}更小的二叉搜索树。这与T_{ij}是最优二叉搜索树矛盾。同理可证，T_r也是一棵最优二叉搜索树。因此，最优二叉搜索树问题具有最优子结构性质。

构造最优二叉搜索树时，可以选择先构造其左右子树，使其左右子树最优，然后构造整棵树。

5 递归计算最优值

- 最优二叉搜索树T_{ij}的平均路长为p_{ij}，则所求的最优值为$p_{1,n}$。由二叉树的花费公式

$$w_{ij}p_{ij} = w_{i,k-1}(p_{i,k-1}+1) + w_{k,k} + w_{k+1,j}(p_{k+1,j}+1)$$
$$= w_{ij} + w_{i,k-1}p_{i,k-1} + w_{k+1,j}p_{k+1,j} \qquad i \le j$$

- 根据最优二叉搜索树问题的最优子结构性质可建立计算p_{ij}的递归式如下

$$w_{ij}p_{ij} = w_{ij} + \min_{i \le k \le j}\{w_{i,k-1}p_{i,k-1} + w_{k+1,j}p_{k+1,j}\}, \quad i \le j$$

初始时 $p_{i,i-1} = 0, \quad 1 \le i \le n$

5 递归计算最优值

- 记 $w_{i,j}p_{i,j}$ 为 $m(i,j)$

$m(1,n)=w_{1,n}p_{1,n}=p_{1,n}$ 为所求最优值。

记 $w_{ij}p_{ij}$ 为 $m(i,j)$ ，得关于 $m(i,j)$ 的递推公式

$$m(i,j)=w_{ij}+\min_{i\le k\le j}\{m(i,k-1)+m(k+1,j)\},\quad i\le j$$

$$m(i,i-1)=0,\quad i=1,2,\cdots,n$$

$$m(i,j)=w_{ij}+\min_{i\le k\le j}\{m(i,k-1)+m(k+1,j)\},\quad i\le j$$

$$m(i,i-1)=0,\quad i=1,2,\cdots,n$$

根据该公式，计算树**T[i][j]**的花费只用到了**T[i][k-1]**，**T[k+1][j]**, 可得到具体求解过程如下：

1）构造只有**1**个内部结点的最优二叉搜索树 **T[1][1],T[2][2]…, T[n][n]**，可以求得**m[i][i]** 同时可以用一个数组存做根结点元素为：

s[1][1]=1, s[2][2]=2…s[n][n]=n

2) 构造具有**2**个内部结点的最优二叉搜索树

5 递归计算最优值

例

给出标识符集{1, 2, 3}＝{do, if, stop}存取概率

若 $P_1=0.5$, $P_2=0.1$, $P_3=0.05$,

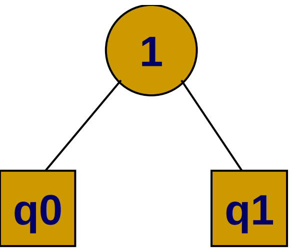
$q_0=0.15$, $q_1=0.1$, $q_2=0.05$, $q_3=0.05$

构造一棵最优二叉搜索树

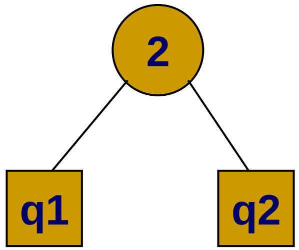
q0=0.15, P1=0.5, q1=0.1, P2=0.1, q2=0.05, P3=0.05, q3=0.05

$$m(i,j) = w_{ij} + \min_{i \le k \le j} \{m(i,k-1) + m(k+1,j)\}, \quad i \le j$$

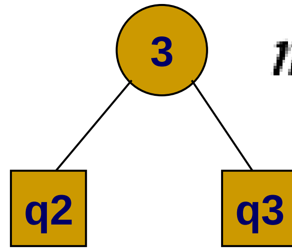
$$m(i,i-1) = 0, \quad i = 1,2,\cdots,n$$



T[1][1]
w[1][1]=0.75
m[1][1]=0.75

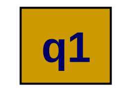


T[2][2]
w[2][2]=0.25
m[2][2]=0.25

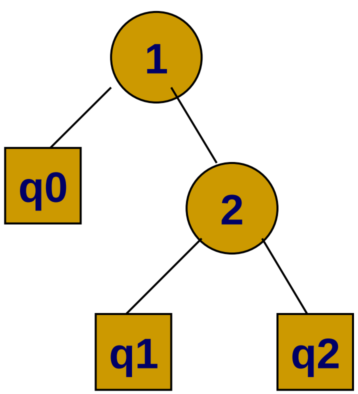


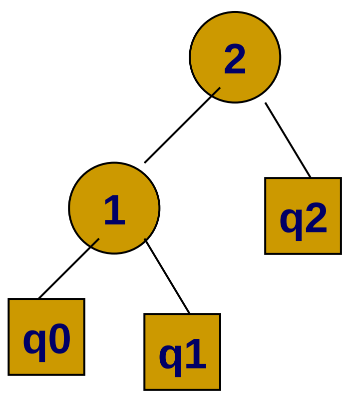
T[3][3]
w[3][3]=0.15
m[3][3]=0.15

T[1][0]
w[1][0]=0.15
m[1][0]=0



T[2][1]
w[2][1]=0.1
m[2][1]=0





T[3][2]
w[3][2]=0.05
m[3][2]=0

T[4][3]
w[4][3]=0.05
m[4][3]=0

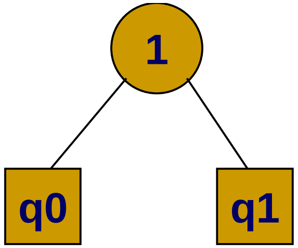
T[1][2]

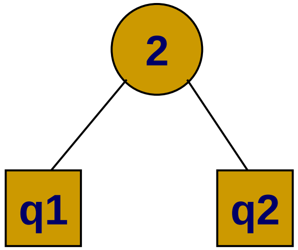
w[1][2]=0.9
m[1][2]=0.9+
m[1][0]+m[2][2]
=1.15

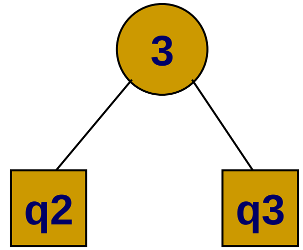
w[1][2]=0.9
m[1][2]=0.9+
m[1][1]+m[3][2]
=1.65

q0=0.15, P1=0.5, q1=0.1, P2=0.1, q2=0.05, P3=0.05, q3=0.05

$$m(i,j) = w_{ij} + \min_{i \le k \le j} \{m(i,k-1) + m(k+1,j)\}, \quad i \le j$$



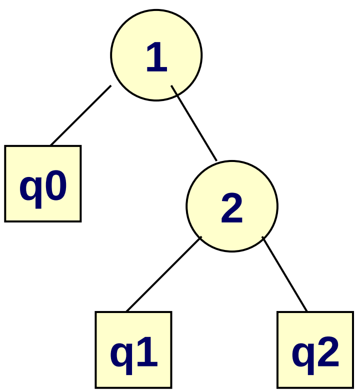


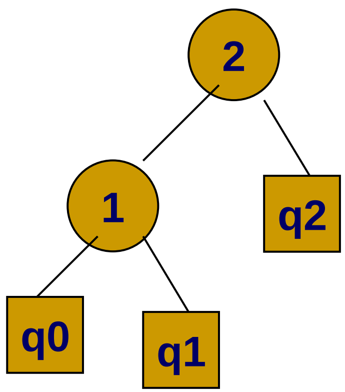


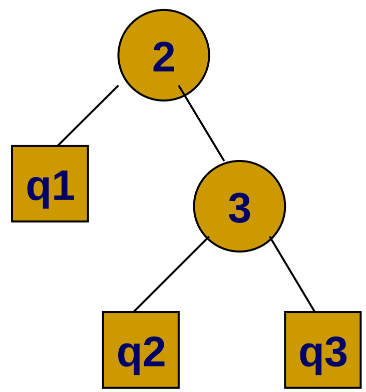
T[1][1]
w[1][1]=0.75
m[1][1]=0.75

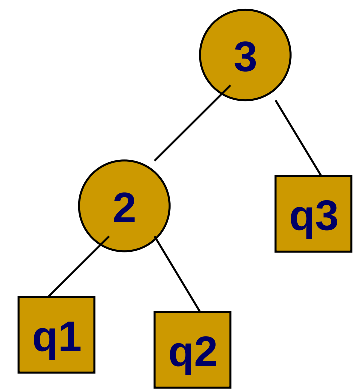
T[2][2]
w[2][2]=0.25
m[2][2]=0.25

T[3][3]
w[3][3]=0.15
m[3][3]=0.15









T[1][2]

w[1][2]=0.9
m[1][2]=0.9+
m[1][0]+m[2][2]
=1.15

w[1][2]=0.9
m[1][2]=0.9+
m[1][1]+m[3][2]
=1.65

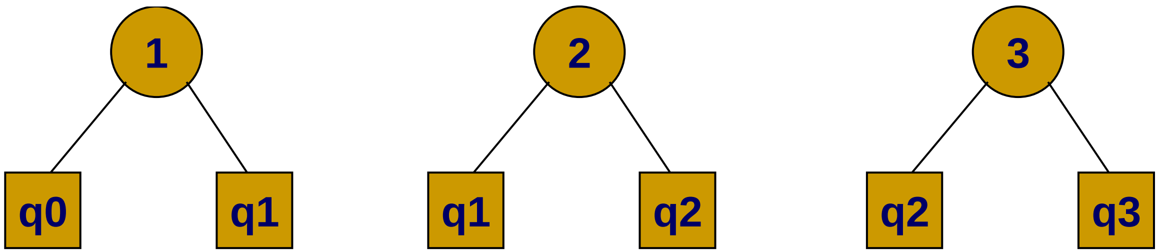
T[2][3]
w[2][3]=0.35

m[2][3]=0.5

m[2][3]=0.6

q0=0.15, P1=0.5, q1=0.1, P2=0.1, q2=0.05, P3=0.05, q3=0.05

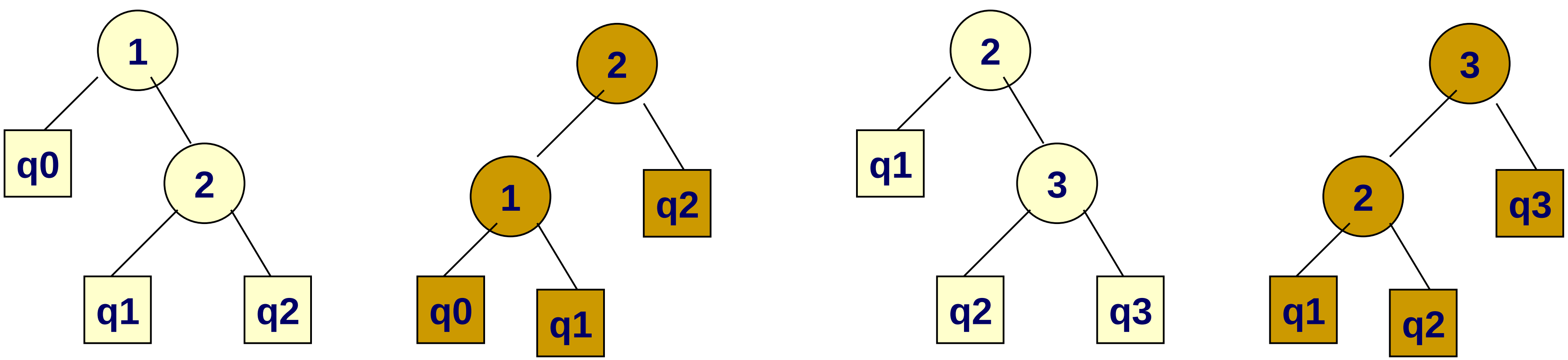
$$m(i,j) = w_{ij} + \min_{i \le k \le j} \{m(i,k-1) + m(k+1,j)\}, \quad i \le j$$



T[1][1]
w[1][1]=0.75
m[1][1]=0.75

T[2][2]
w[2][2]=0.25
m[2][2]=0.25

T[3][3]
w[3][3]=0.15
m[3][3]=0.15



w[1][2]=0.9
m[1][2]=0.9+
m[1][0]+m[2][2]
=1.15

T[1][2]

w[1][2]=0.9
m[1][2]=0.9+
m[1][1]+m[3][2]
=1.65

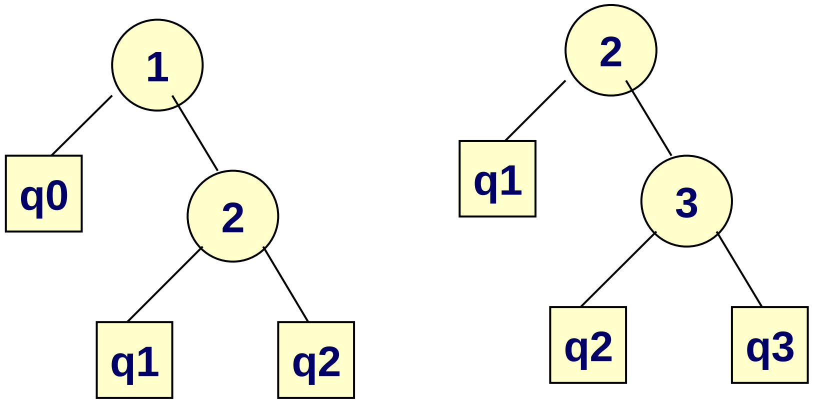
T[2][3]
w[2][3]=0. 35

m[2][3]=0.5

m[2][3]=0.6

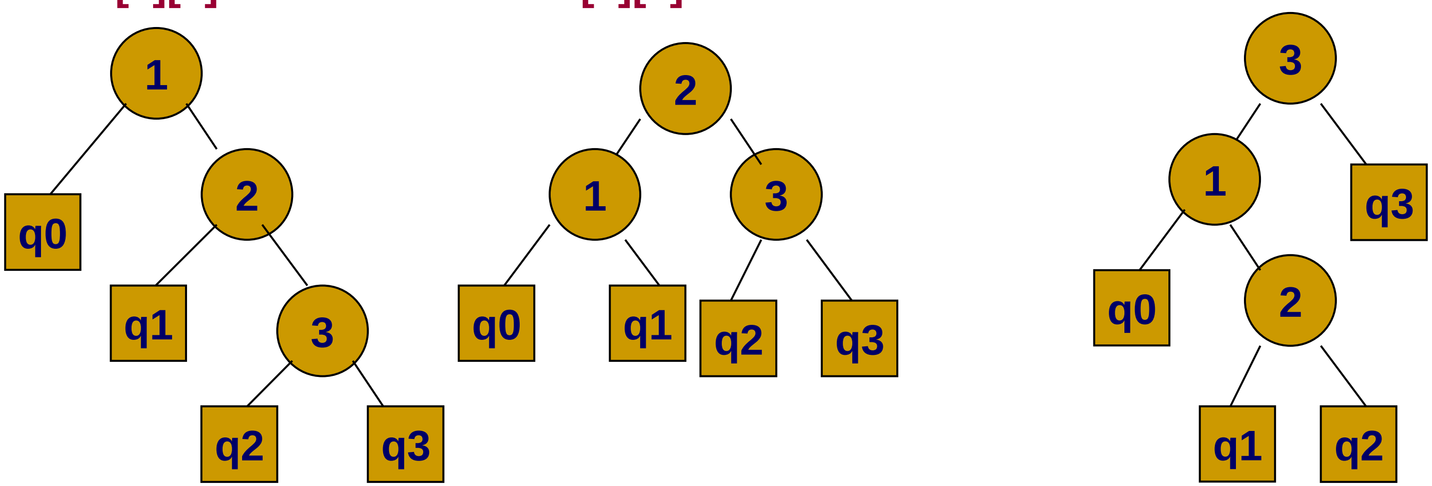
q0=0.15, P1=0.5, q1=0.1, P2=0.1, q2=0.05, P3=0.05, q3=0.05

$$m(i,j) = w_{ij} + \min_{i \le k \le j} \{m(i,k-1) + m(k+1,j)\}, \quad i \le j$$



T[1][2]
m[1][2]=1.15

T[2][3]
m[2][3]=0.5



T[1][3]
W[1][3]=1

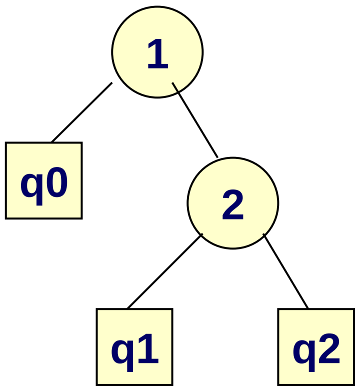
m[1][3]=1.5

m[1][3]=1.9

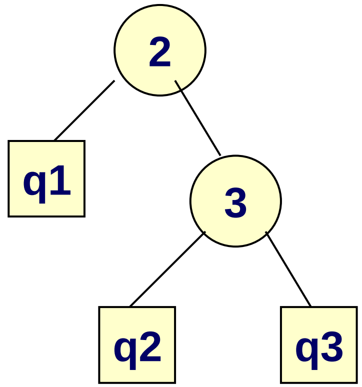
m[1][3]=2.15

q0=0.15, P1=0.5, q1=0.1, P2=0.1, q2=0.05, P3=0.05, q3=0.05

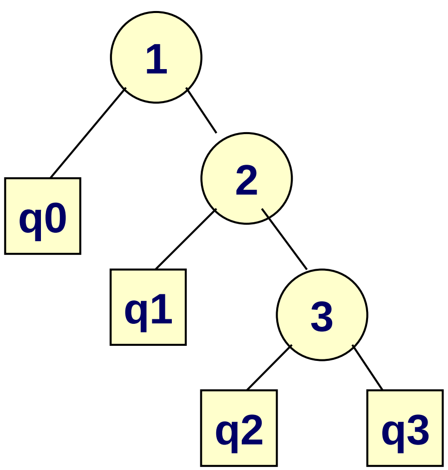
$$m(i,j) = w_{ij} + \min_{i \le k \le j} \{m(i,k-1) + m(k+1,j)\}, \quad i \le j$$

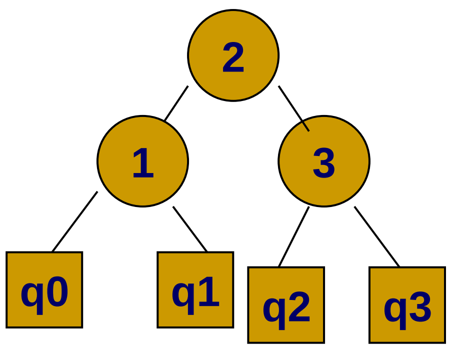


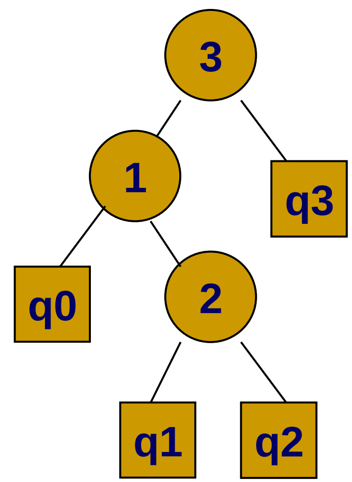
T[1][2]
m[1][2]=1.15



T[2][3]
m[2][3]=0.5







T[1][3]
W[1][3]=1

m[1][3]=1.5

m[1][3]=1.9

m[1][3]=2.15

q0=0.15, P1=0.5, q1=0.1, P2=0.1, q2=0.05, P3=0.05, q3=0.05

$$m(i,j) = w_{ij} + \min_{i \le k \le j} \{m(i,k-1) + m(k+1,j)\}, \quad i \le j$$

	0	1	2	3
1	0.15	0.75	0.9	1
2		0.1	0.25	0.35
3			0.05	0.15
4				0.05

W(i, j)

	0	1	2	3
1	0	0.75	1.15	1.5
2		0	0.25	0. 5
3			0	0.15
4				0

m(i, j)

	0	1	2	3
1	0	1	1	1
2		0	2	2
3			0	3
4				0

s(i, j)

5 递归计算最优值

具体求解过程

1) 递归出口，没有内部节点时，构造T[1][0] T[2][1],T[3][2]……，T[n+1][n]

2) 构造只有1个内部结点的最优二叉搜索树T[1][1],T[2][2]…, T[n][n]，可以求得m[i][i] 同时可以用一个数组存做根结点元素为：s[1][1]=1, s[2][2]=2…s[n][n]=n

3) 构造具有2个、3个、……、n个内部结点的最优二叉搜索树

r （ 起止下标的差）

0 T[1][1], T[2][2] , …，T[n][n],

1 T[1][2], T[2][3], …，T[n-1][n],

2 T[1][3], T[2][4], …，T[n-2][n],

⋮

r T[1][r+1], T[2][r+2], …，T[i][i+r]，…，T[n-r][n]

⋮

n-1 T[1][n]

```
void OBST ( int *a, int *b,int n, int **m, int **s, int **w)
{
    for( int i=0; i<=n; i++)
    {
        w[i+1][i]=a[i];
        m[i+1][i]=0;
    }//初始化，构造没有内部节点时的情况
    for(int r=0; r<n; r++)
        for(int i =1; i<=n-r; i++)
        {
            int j= i+r;
            构造T[i][j]，填写w[i][j],m[i][j],s[i][j]
        }
}
```

5 递归计算最优值

构造T[i][j]

T[i][j]表示用第i到第j个内部节点构造的树，做根的结点可以是第i，i+1,…,j中任意一个。

1) 首选i作为根，其左子树空，右子树为结点i+1,i+2…j构成即T[i+1][j]。

m[i][j]=w[i][j]+0+m[i+1][j]

s[i][j]= i

2) 不选i做根，设k为其根，则k= i+1,…,j,左子树为结点i, i+1,…,k-1, 右子树为k+1, k+2, …,j

```
t= w[i][j]+m[i][k-1]+m[k+1][j]
if(t < m[i][j])
  { m[i][j]=t;
    s[i][j]=k;
  }
```

3)k= k+1, 跳回2

5 递归计算最优值

```
void OptimalBinarySearchTree( int *a, int *b,int n, int **m, int **s, int **w)
{
  for( int i=0; i<=n; i++){
    w[i+1][i]=a[i];
    m[i+1][i]=0;
    }
  for(int r=0; r<n; r++)
    for(int i =1; i<= n - r; i++){
      int j= j + r;
      w[i][j]=w[i][j-1]+a[j]+b[j];
      m[i][j]=m[i+1][j];
    s[i][j] = i;
for(int k = i+1; k<=j; k++){
     int t=m[i][k-1]+m[k+1][j];
     if (t<m[i][j]){
       m[i][j]=t;
        s[i][j]=k;}
          }
     m[i][j]+=w[i][j];
     }
}
```

初始化
对角线赋值

i为起始元素下标

j为终止元素下标

如第i个结点作根的值

加第j个结点后，权值w改变

取第k个结点作根

6 构造最优解

算法 OptimalBinarySearchTree 中用 s[I][j]保存最优子树的根顶点中元素。当 s[1][n] = k 时，x_k 为所求二叉搜索树的根顶点元素。其左子树为 T(1,k-1)，因此 I = s[1][k-1]即表示 T(1,k-1)的根顶点为 x_i。依次类推，容易由 s 记录的信息在 O(n)时间内构造出所求的最优二叉搜索树。

7 计算复杂性

算法中用到三个数组 m、s、w，故所需的空间为 O(n^2).算法的主要计算量在于计算 $w_{ij}+\min\limits_{i\le k\le j}\{m(i,k-1)+m(k+1,j)\}$。对于固定的差 r=j-I，需要计算时间 $O(j-i+1)=O(r+1)$。因此算法所耗费的总时间为 $\sum\limits_{r=0}^{n-1}\sum\limits_{i=1}^{n-r}O(r+1)=O(n^3)$.

最优二叉搜索树习题

- 给定7个关键字及其存取概率分布，确定其最优二叉搜索树及其代价
- 关键字集{1，2，3，4，5，6，7}
- 存取概率：

i：	0	1	2	3	4	5	6	7
pi:		0.04	0.06	0.08	0.02	0.10	0.12	0.14
qi:	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05