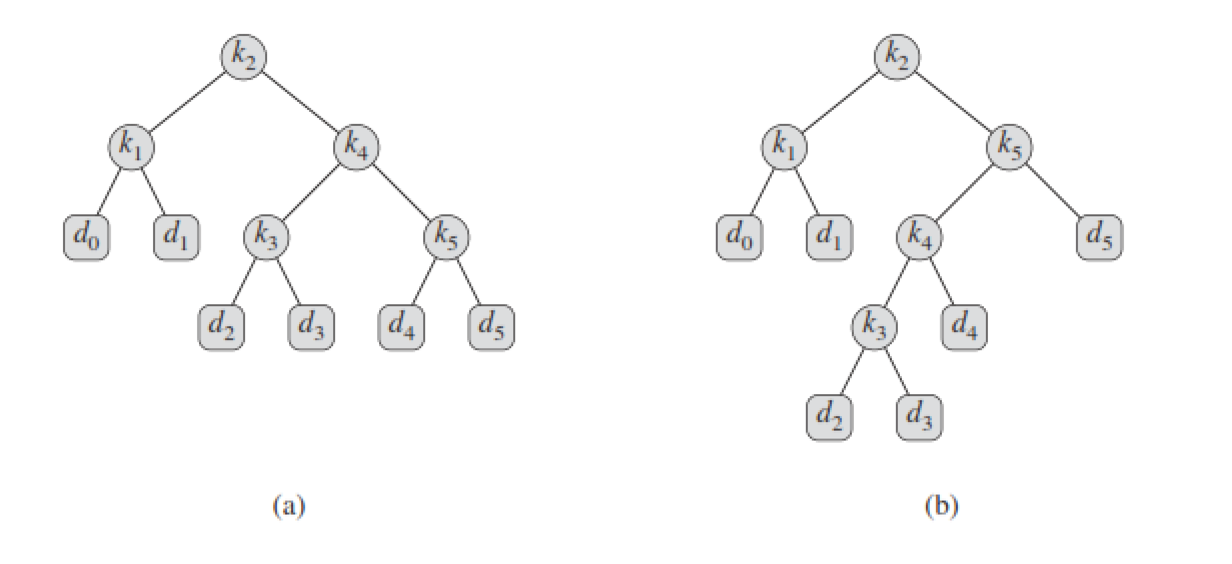
OBST

What we need is known as an ***optimal binary search tree***. Formally, we are given a sequence $K = \langle k_1, k_2, \ldots, k_n \rangle$ of n distinct keys in sorted order (so that $k_1 < k_2 < \cdots < k_n$), and we wish to build a binary search tree from these keys. For each key k_i, we have a probability p_i that a search will be for k_i. Some searches may be for values not in K, and so we also have $n + 1$ "dummy keys"

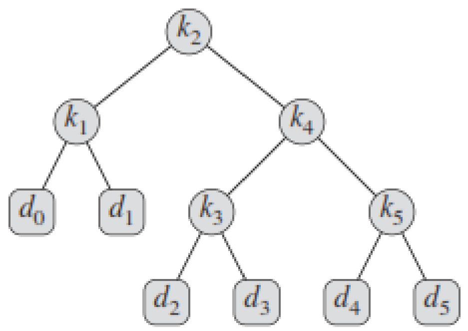
$d_0, d_1, d_2, \ldots, d_n$ representing values not in K.



i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10

$$\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1 .$$

$$\begin{aligned} \mathrm{E}[\text{search cost in } T] &= \sum_{i=1}^{n} (\mathrm{depth}_T(k_i) + 1) \cdot p_i + \sum_{i=0}^{n} (\mathrm{depth}_T(d_i) + 1) \cdot q_i \\ &= 1 + \sum_{i=1}^{n} \mathrm{depth}_T(k_i) \cdot p_i + \sum_{i=0}^{n} \mathrm{depth}_T(d_i) \cdot q_i . \end{aligned} \tag{15.11}$$



node	depth	probability	contribution
k_1	1	0.15	0.30
k_2	0	0.10	0.10
k_3	2	0.05	0.15
k_4	1	0.10	0.20
k_5	2	0.20	0.60
d_0	2	0.05	0.15
d_1	2	0.10	0.30
d_2	3	0.05	0.20
d_3	3	0.05	0.20
d_4	3	0.05	0.20
d_5	3	0.10	0.40
Total			2.80

i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10

with an observation about subtrees. Consider any subtree of a binary search tree. It must contain keys in a contiguous range $k_i, \ldots, k_j$, for some $1 \leq i \leq j \leq n$. In addition, a subtree that contains keys $k_i, \ldots, k_j$ must also have as its leaves the dummy keys $d_{i-1}, \ldots, d_j$.

The easy case occurs when $j = i - 1$. Then we have just the dummy key d_{i-1}. The expected search cost is $e[i, i-1] = q_{i-1}$.

When $j \geq i$, we need to select a root k_r from among $k_i, \ldots, k_j$ and then make an optimal binary search tree with keys $k_i, \ldots, k_{r-1}$ as its left subtree and an optimal binary search tree with keys $k_{r+1}, \ldots, k_j$ as its right subtree. What happens to the

$$w(i, j) = \sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l . \tag{15.12}$$

Thus, if k_r is the root of an optimal subtree containing keys $k_i, \ldots, k_j$, we have

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

Noting that

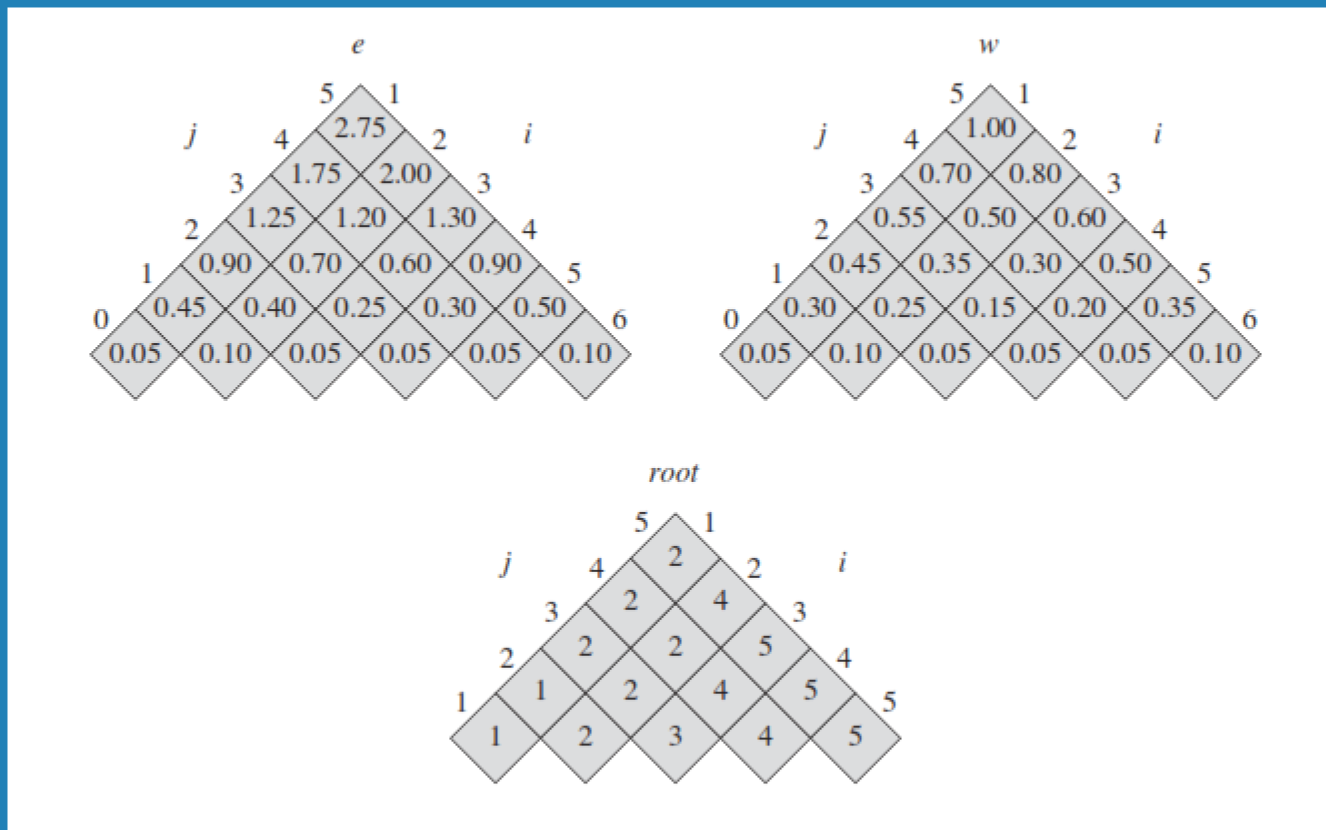
$$w(i, j) = w(i, r-1) + p_r + w(r+1, j) ,$$

we rewrite $e[i, j]$ as

$$e[i, j] = e[i, r-1] + e[r+1, j] + w(i, j) . \tag{15.13}$$

The recursive equation (15.13) assumes that we know which node k_r to use as the root. We choose the root that gives the lowest expected search cost, giving us our final recursive formulation:

$$e[i, j] = \begin{cases} q_{i-1} & \text{if } j = i-1 , \\ \min\limits_{i \le r \le j} \{e[i, r-1] + e[r+1, j] + w(i, j)\} & \text{if } i \le j . \end{cases} \tag{15.14}$$


e
2.75
1.75 2.00
1.25 1.20 1.30
0.90 0.70 0.60 0.90
0.45 0.40 0.25 0.30 0.50
0.05 0.10 0.05 0.05 0.05 0.10
w
1.00
0.70 0.80
0.55 0.50 0.60
0.45 0.35 0.30 0.50
0.30 0.25 0.15 0.20 0.35
0.05 0.10 0.05 0.05 0.05 0.10
root
2
2 4
2 2 5
1 2 4 5
1 2 3 4 5
j
i

OPTIMAL-BST(p, q, n)

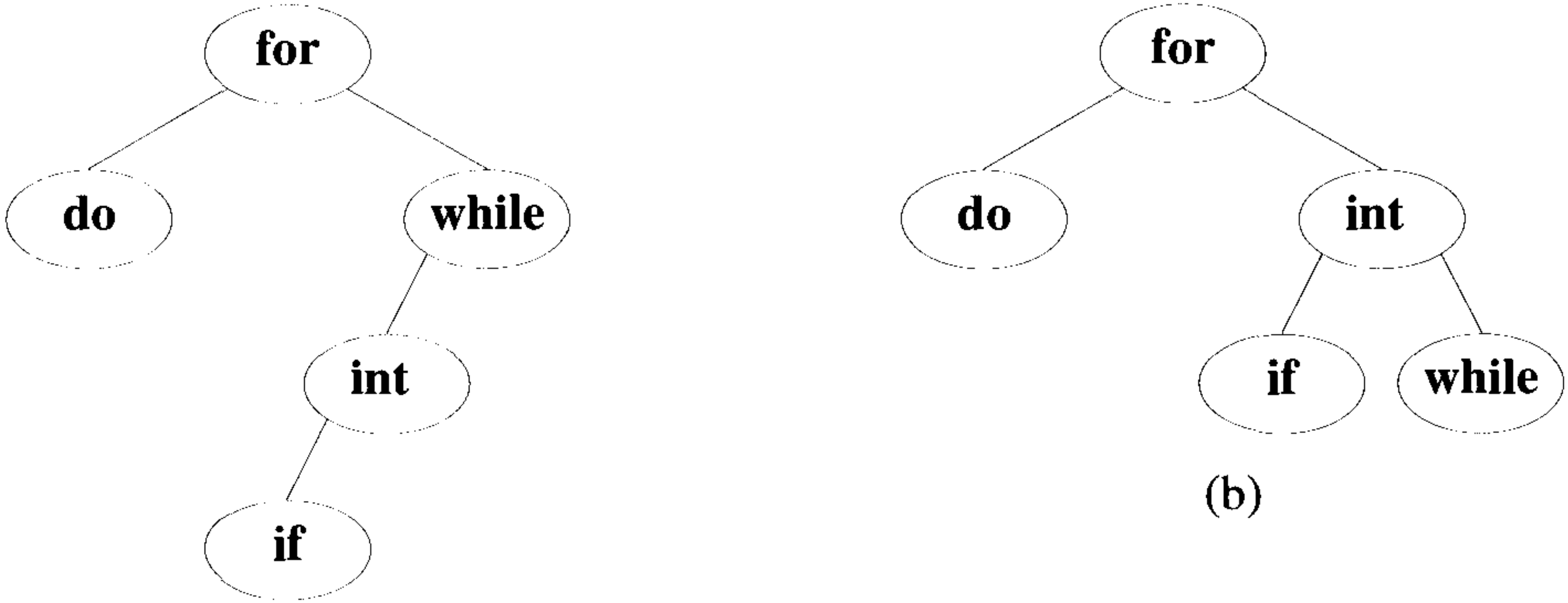
```
let e[1..n+1, 0..n], w[1..n+1, 0..n],
        and root[1..n, 1..n] be new tables
for i = 1 to n + 1
    e[i, i-1] = q_{i-1}
    w[i, i-1] = q_{i-1}
for l = 1 to n
    for i = 1 to n - l + 1
        j = i + l - 1
        e[i, j] = ∞
        w[i, j] = w[i, j-1] + p_j + q_j
        for r = i to j
            t = e[i, r-1] + e[r+1, j] + w[i, j]
            if t < e[i, j]
                e[i, j] = t
                root[i, j] = r
return e and root
```

base case, we compute $w[i, i-1] = q_{i-1}$ for $1 \le i \le n+1$. For $j \ge i$, we compute

$$w[i, j] = w[i, j-1] + p_j + q_j \ . \tag{15.15}$$

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

k_2 is the root
k_1 is the left child of k_2
d_0 is the left child of k_1
d_1 is the right child of k_1
k_5 is the right child of k_2
k_4 is the left child of k_5
k_3 is the left child of k_4
d_2 is the left child of k_3
d_3 is the right child of k_3
d_4 is the right child of k_4
d_5 is the right child of k_5

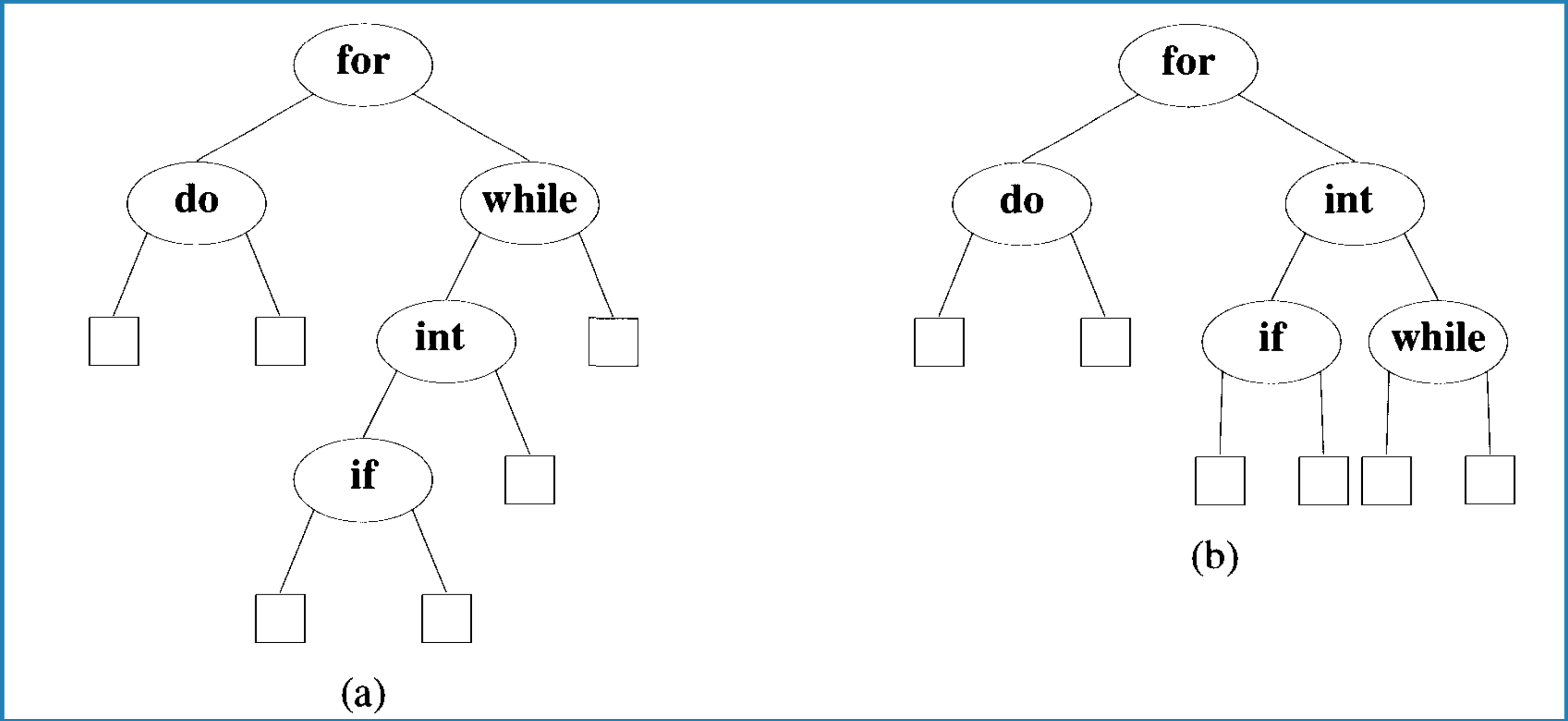


(a)

(b)

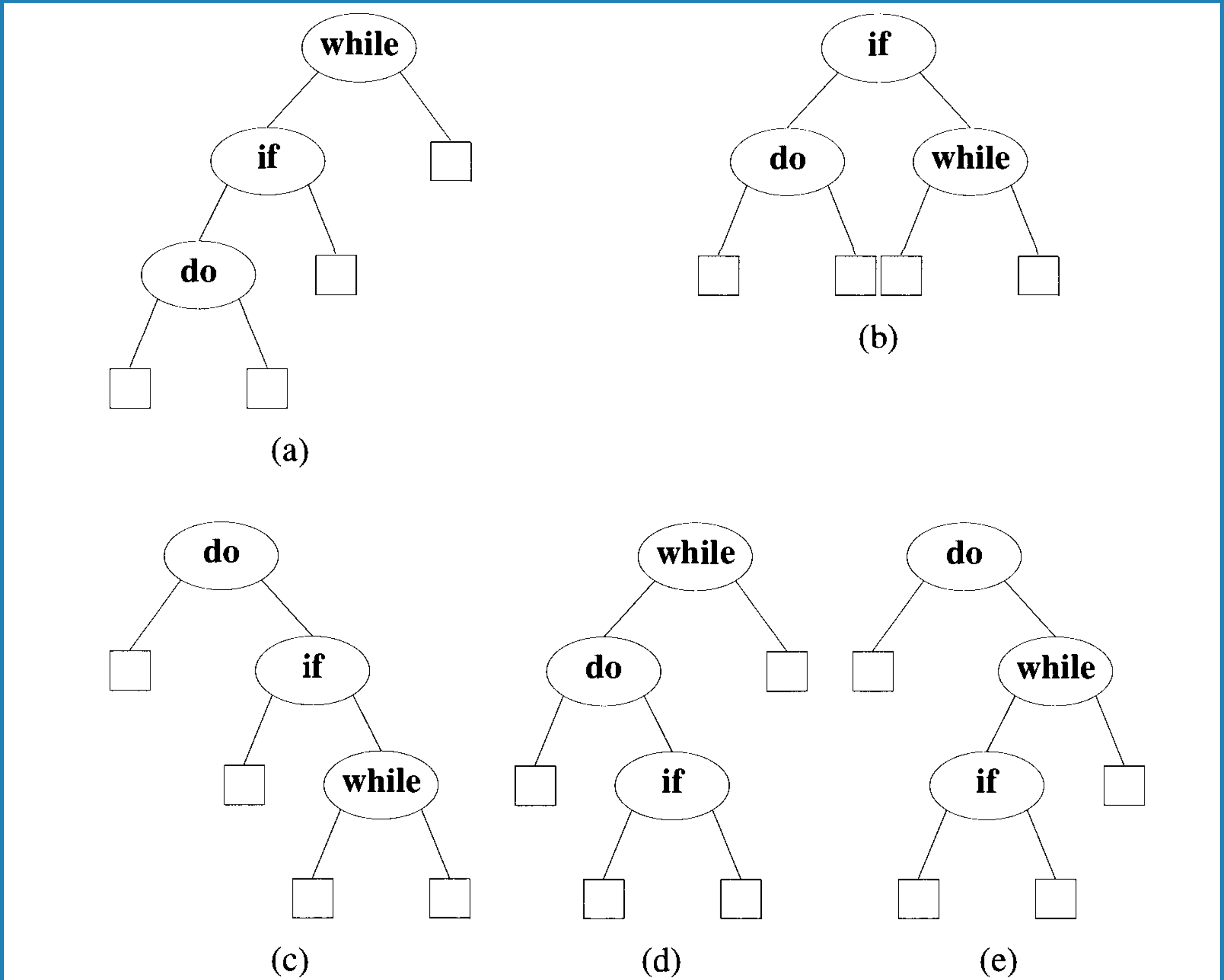
for with different frequencies (or probabilities). In addition, we can expect unsuccessful searches also to be made. Let us assume that the given set of identifiers is $\{a_1, a_2, \ldots, a_n\}$ with $a_1 < a_2 < \cdots < a_n$. Let $p(i)$ be the probability with which we search for a_i. Let $q(i)$ be the probability that the identifier x being searched for is such that $a_i < x < a_{i+1}$, $0 \le i \le n$ (assume $a_0 = -\infty$ and $a_{n+1} = +\infty$). Then, $\sum_{0 \le i \le n} q(i)$ is the probability of an unsuccessful search.

Clearly, $\sum_{1 \le i \le n} p(i) + \sum_{0 \le i \le n} q(i) = 1$.


for
do
while
int
if
for
do
int
if
while
(a)
(b)

class E_0 contains all identifiers x such that $x < a_1$. The class E_i contains all identifiers x such that $a_i < x < a_{i+1}$, $1 \leq i < n$. The class E_n contains all identifiers x, $x > a_n$. It is easy to see that for all identifiers in the same class E_i, the search terminates at the same external node. For identifiers in different E_i the search terminates at different external nodes. If the failure

$$\sum_{1 \leq i \leq n} p(i) * level(a_i) + \sum_{0 \leq i \leq n} q(i) * (level(E_i) - 1)$$



while
if
do
(a)
if
do
while
(b)
do
if
while
(c)
while
do
if
(d)
do
while
if
(e)

Example 5.17 The possible binary search trees for the identifier set $(a_1, a_2, a_3) =$ (**do, if, while**) are given if Figure 5.14. With equal probabilities $p(i) = q(i) = 1/7$ for all i, we have

$$\begin{aligned} cost(\text{tree a}) &= 15/7 \quad & cost(\text{tree b}) &= 13/7 \\ cost(\text{tree c}) &= 15/7 \quad & cost(\text{tree d}) &= 15/7 \\ cost(\text{tree e}) &= 15/7 \end{aligned}$$

tree b is optimal.

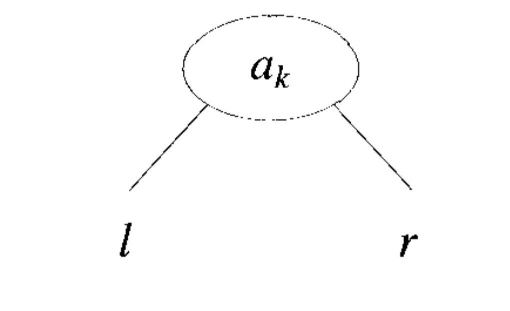
As expected, tree b is optimal. With $p(1) = .5$, $p(2) = .1$, $p(3) = .05$, $q(0) = .15$, $q(1) = .1$, $q(2) = .05$ and $q(3) = .05$ we have

$$\begin{aligned} cost(\text{tree a}) &= 2.65 \quad & cost(\text{tree b}) &= 1.9 \\ cost(\text{tree c}) &= 1.5 \quad & cost(\text{tree d}) &= 2.05 \\ cost(\text{tree e}) &= 1.6 \end{aligned}$$

a_i's should be assigned to the root node of the tree. If we choose a_k, then it is clear that the internal nodes for $a_1, a_2, \ldots, a_{k-1}$ as well as the external nodes for the classes $E_0, E_1, \ldots, E_{k-1}$ will lie in the left subtree l of the root. The remaining nodes will be in the right subtree r. Define

$$cost(l) = \sum_{1 \leq i < k} p(i) * \text{level}(a_i) + \sum_{0 \leq i < k} q(i) * (\text{level}(E_i) - 1)$$

$$cost(r) = \sum_{k < i \leq n} p(i) * \text{level}(a_i) + \sum_{k < i \leq n} q(i) * (\text{level}(E_i) - 1)$$



Using $w(i, j)$ to represent the sum $q(i) + \sum_{l=i+1}^{j}(q(l) + p(l))$, we obtain the following as the expected cost of the search tree (Figure 5.15):

$$p(k) + cost(l) + cost(r) + w(0, k-1) + w(k, n) \tag{5.10}$$

If the tree is optimal, then (5.10) must be minimum. Hence, $cost(l)$ must be minimum over all binary search trees containing $a_1, a_2, \ldots, a_{k-1}$ and $E_0, E_1, \ldots, E_{k-1}$. Similarly $cost(r)$ must be minimum. If we use $c(i, j)$ to represent the cost of an optimal binary search tree t_{ij} containing $a_{i+1}, \ldots, a_j$ and $E_i, \ldots, E_j$, then for the tree to be optimal, we must have $cost(l) = c(0, k-1)$ and $cost(r) = c(k, n)$. In addition, k must be chosen such that

$$p(k) + c(0, k-1) + c(k, n) + w(0, k-1) + w(k, n)$$

is minimum. Hence, for $c(0, n)$ we obtain

$$c(0,n) = \min_{1 \le k \le n} \{c(0,k-1) + c(k,n) + p(k) + w(0,k-1) + w(k,n)\} \quad (5.11)$$

We can generalize (5.11) to obtain for any $c(i,j)$

$$c(i,j) \quad = \quad \min_{i<k\le j} \{c(i,k-1) + c(k,j) + p(k) + w(i,k-1) + w(k,j)\}$$

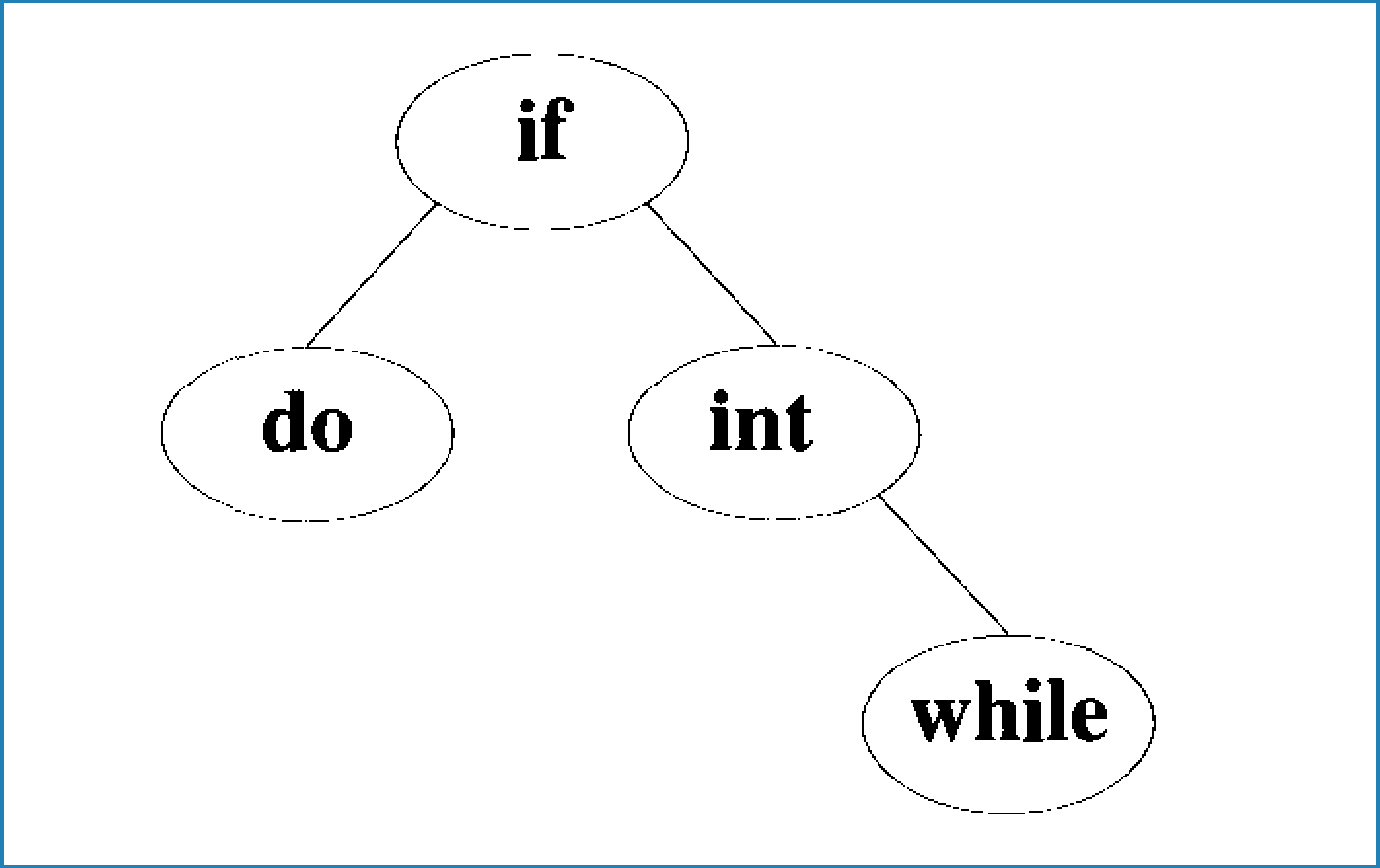
$$c(i,j) \quad = \quad \min_{i<k\le j} \{c(i,k-1) + c(k,j)\} + w(i,j)$$

that $j-i=1$ (note $c(i,i)=0$ and $w(i,i)=q(i)$, $0 \le i \le n$). Next we can compute all $c(i,j)$ such that $j-i=2$, then all $c(i,j)$ with $j-i=3$, and so on. If during this computation we record the root $r(i,j)$ of each tree t_{ij}, then an optimal binary search tree can be constructed from these $r(i,j)$. Note that $r(i,j)$ is the value of k that minimizes (5.12).

Example 5.18 Let $n=4$ and $(a_1,a_2,a_3,a_4) =$ (**do, if, int, while**). Let $p(1:4)=(3,3,1,1)$ and $q(0:4)=(2,3,1,1,1)$. The p's and q's have been multiplied by 16 for convenience. Initially, we have $w(i,i)=q(i), c(i,i)=0$ and $r(i,i)=0$, $0 \le i \le 4$. Using Equation 5.12 and the observation $w(i,j)=p(j)+q(j)+w(i,j-1)$, we get

$$
\begin{aligned}
w(0,1) &= p(1)+q(1)+w(0,0)=8\\
c(0,1) &= w(0,1)+\min\{c(0,0)+c(1,1)\} \quad = \quad 8\\
r(0,1) &= 1\\
w(1,2) &= p(2)+q(2)+w(1,1) \quad = \quad 7\\
c(1,2) &= w(1,2)+\min\ \{c(1,1)+c(2,2)\} \quad = \quad 7\\
r(0,2) &= 2\\
w(2,3) &= p(3)+q(3)+w(2,2) \quad = \quad 3\\
c(2,3) &= w(2,3)+\min\ \{c(2,2)+c(3,3)\} \quad = \quad 3\\
r(2,3) &= 3\\
w(3,4) &= p(4)+q(4)+w(3,3) \quad = \quad 3\\
c(3,4) &= w(3,4)+\min\ \{c(3,3)+c(4,4)\} \quad = \quad 3\\
r(3,4) &= 4
\end{aligned}
$$

	0	1	2	3	4
0	$w_{00} = 2$ $c_{00} = 0$ $r_{00} = 0$	$w_{11} = 3$ $c_{11} = 0$ $r_{11} = 0$	$w_{22} = 1$ $c_{22} = 0$ $r_{22} = 0$	$w_{33} = 1$ $c_{33} = 0$ $r_{33} = 0$	$w_{44} = 1$ $c_{44} = 0$ $r_{44} = 0$
1	$w_{01} = 8$ $c_{01} = 8$ $r_{01} = 1$	$w_{12} = 7$ $c_{12} = 7$ $r_{12} = 2$	$w_{23} = 3$ $c_{23} = 3$ $r_{23} = 3$	$w_{34} = 3$ $c_{34} = 3$ $r_{34} = 4$	
2	$w_{02} = 12$ $c_{02} = 19$ $r_{02} = 1$	$w_{13} = 9$ $c_{13} = 12$ $r_{13} = 2$	$w_{24} = 5$ $c_{24} = 8$ $r_{24} = 3$		
3	$w_{03} = 14$ $c_{03} = 25$ $r_{03} = 2$	$w_{14} = 11$ $c_{14} = 19$ $r_{14} = 2$			
4	$w_{04} = 16$ $c_{04} = 32$ $r_{04} = 2$				



if
do
int
while